A Project Report

On

Face Mask Detection Using Deep Learning

Submitted in partial fulfillment of the
requirement for the award of the degree of

Bachelor of Computer Application

GALGOTIAS UNIVERSITY
(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of
DR. SEEMA RANI
Assistant Professor

Submitted By

DEEPANSHU
BANSAL
(19SCSE1040015)

SANDEEP SINGH
(19SCSE1040027)

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING GALGOTIAS UNIVERSITY, GREATER
NOIDA, INDIA

MAY, 2022

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the project, entitled **" Face Mask Detection using Deep learning "** in partial fulfillment of the requirements for the award of the **BACHELOR OF COMPUTER APPLICATION** submitted in the **School of Computing Science and Engineering** of Galgotias University, Greater Noida, is an original work carried out during the period of **JANUARY -2022 to JULY -2022**, under the supervision of **Dr. SEEMA RANI , Assistant Professor**, **Department of Computer Science and Engineering** of School of Computing Science and Engineering , Galgotias University, Greater Noida.

The matter presented in the project has not been submitted by me/us for the award of any other degree of this or any other places.

19SCSE1040027 - SANDEEP SINGH
19SCSE1040015 - DEEPANSHU BANSAL

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor

(Dr. **SEEMA RANI,**
Assistant Professor)

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of **19SCSE1040015 – DEEPANSHU BANSAL, 19SCSE10400127– SANDEEP SINGH** has been held on ________and his/her work is recommended for the award of **BACHELOR OF COMPUTER APPLICATION**

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date: Place:

ACKNOWLEDGEMENT

Apart from the efforts of our, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We would like to show my greatest appreciation to DR. SEEMA RANI . We can’t say thank you enough for his tremendous support and help. We feel motivated and encouraged every time, We attend his meeting. Without his encouragement and guidance this project would not have materialized.

The guidance and support received from all the members who contributed and who are contributing to this project, was vital for the success of the project. We are grateful for their constant support and help.

Table of Contents

Abstract

The present scenario of COVID-19 demands an efficient face mask detection application. The main goal of the project is to implement this system at entrances of colleges, airports, hospitals, and offices where chances of spread of COVID-19 through contagion are relatively higher. Reports indicate that wearing face masks while at work clearly reduces the risk of transmission. It helps raise awareness and prioritizes health andsafety in public places where wearing a mouth and nose covering is mandatory. Using image recognition and machine-learning technology, it monitors whether individuals are correctly wearing their face masks in public spaces by adding a visual overlay. A hybrid model using deep and classical machine learning for detecting face mask will be presented. A data set is used to build this face mask detector using Python, OpenCV, and TensorFlow and Keras. While entering the place everyone should scan their face and then enter ensuring they have a mask with them. If anyone is found to be without a face mask, beep alert will be generated. As all the workplaces are opening. The number of cases of COVID-19 are still getting registered throughout the country. If everyone follows the safety measures, then it can come to an end. A deep learning-based approach for detecting masks over faces in public placesto curtail the community spread of Corona virus is present

Chapter 1

Introduction

In this work, a deep learning-based approach for detecting masks over faces in public places to curtail the community spread of Coronavirus is presented. The proposed technique efficiently handles occlusions in dense situations by making use of an ensemble of single and two-stage detectors at the pre-processing level.The ensemble approach not only helps in achieving high accuracy but also improves detection speed considerably. Face mask detection refers to detect whether a person is wearing a mask or not. In fact, the problem is reverse engineering of face detection where the face is detected using different machine learning algorithms for the purpose of security, authentication and surveillance. Face detection is a key area in the field of Computer Vision and Pattern Recognition.

Although numerous researchers have committed efforts in designing efficient algorithms for face detection and recognition but there exists an essential difference between 'detection of the face under mask' and 'detection of mask over face'. As per available literature, very little body of research is attempted to detect mask over face.

A. TensorFlow

TensorFlow, an interface for expressing machine learning algorithms, is utilized for implementing ML systems into fabrication over a bunch of areas of computer science, including sentiment analysis, voice recognition, geographic information extraction, computer vision, text summarization, information retrieval, computational drug discovery and flaw detection to pursue research . In the proposed model, the whole Sequential CNN architecture (consists of several layers) uses TensorFlow at backend. Itis also used to reshape the data (image) in the data processing.

B. Keras

Keras gives fundamental reflections and building units for creation and transportation ofML arrangements with high iteration velocity. It takes full advantage of the scalability and cross-platform capabilities of TensorFlow. The core data structures of Keras are layers and models . All the layers used in the CNN model are implemented using Keras. Along with the conversion of the class vector to the binary class matrix in data processing, it helps to compile the overall model.

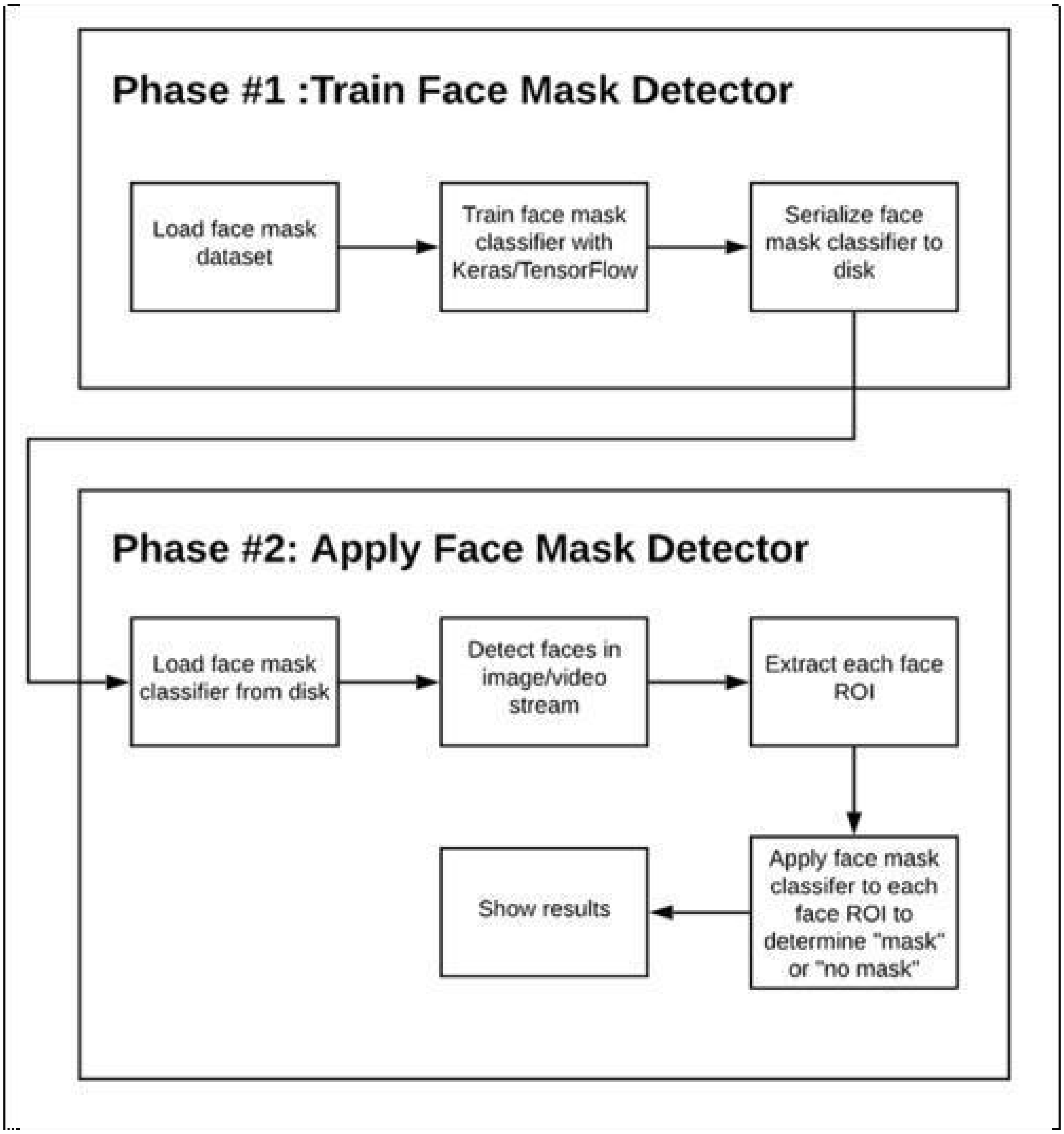
C. OpenCV

OpenCV (Open Source Computer Vision Library), an open-source computer vision andML software library, is utilized to differentiate and recognize faces, recognize objects, group movements in recordings, trace progressive modules, follow eye gesture, track camera actions, expel red eyes from pictures taken utilizing flash, find comparative pictures from an image database, perceive landscape and set up markers to overlay itwith increased reality and so forth. The proposed method makes use of these features of OpenCV in resizing and color conversion of data image

Two Face For Face Mask Detection:-

1. **Training:** Here we'll focus on loading our face mask detection dataset from disk, training a model (using Keras/TensorFlow) on this dataset, and then serializing the face mask detector to disk**:** Once the face mask detector is trained, we can then move on to loading the mask detector, performing face detection, and then classifying each face as With mask and without mask

2. **Deployment:** Once the face mask detector is trained, we can then move on to loading the mask detector, performing face detection, and then classifyingeach face as With mask and without mask


Phase #1 :Train 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 classifier from disk
Detect faces in image/video stream
Extract each face ROI
Apply face mask classifer to each face ROI to determine "mask" or "no mask"
Show results

Applications of face mask detection

In our study the public services including hospitals, stations, supermarkets, banks, stores, or other public places. So, our research proposed an application of a mask detector to prevent Covid-19 in the public services area.

In this research, we developed an application to solve the problem for break the distribution chain of COVID-19 in the public service area, because when someoneis outside the house there will be lots of threats of virus transmission. So the whole community needs to use masks. Then, we aim to develop applications and investigate mask detectors to prevent Covid-19 because the use of masks canprevent the transmission of sparks that can be used to protect others, and help contaminate the environment due to this spark, to minimize the spread of COVID- 19 due to the use of masks which has been monitored in public areas

The COVID-19 epidemic has quickly impacted our daily lives by disrupting trade and the movement of the earth. Wearing a protective face mask has become a new trend. In the near future, many government service providers will be asking clients to wear a mask properly to maximize their services. Therefore, the discovery of a face mask has become an important task of helping the international community. This paper introduces a simplified way to achieve this goal using basic machine learning packages such as TensorFlow, Keras. OpenCV and Scikit-Learn. The proposed method detects the surface in the image correctly and indicates whether it has a mask on it or not. As a security guard, it can also detect faces and moving masks. The method gets high accuracy up to 95.77% and 94.58% respectively on two separate databases.

We evaluated improved parameter values using the Sequential Convolutional Neural Network model to determine the presence of the mask correctly without causing excessive alignment. Section 1 Introduction According to the official World Health Organization (WHO) Situation Report - 205, coronavirus 2019 (COVID-19) has infected more than 20 million people worldwide and caused more than 0.7 million deaths. People with COVID-19 have had a wide range of reported symptoms ranging from mild appearance to serious illness. Respiratory problems such as shortness of breath or difficulty breathing are one of them. Adults with pneumonia may be more prone to

COVID-19 complications as they appear to be at higher risk. Other common human coronaviruses that infect humans worldwide are 229E, HKUI, OC43, and NL63. Before harming individuals. viruses such as 2019-nCoV, SARSCOV, and MERS- COV, infect animals and mutate into human coronaviruses. People with respiratory problems can expose anyone (close to them) to contagious beads. Circumcision of an unclean person can cause human infections as droplets that carry the virus

CHAPTER-2

Literature Survey

Face detection is defined as the procedure that has many applications like face tracking, pose estimation or compression. Face detection is a two-class problem where we have to decide if there is a face or not in a picture. This approach can be seen as a simplified face recognition problem.

Pattern learning and object recognition are the inherent tasks that a computer vision (CV) technique must deal with. Object recognition encompasses both image classification and object detection . The task of recognizing the mask over the face in the pubic area can be achieved by deploying an efficient object recognition algorithm through surveillance devices. The object recognition pipeline consists of generating the region proposals followed by classification of each proposal into related class.

On the way to find the face, the face is found in a picture that has a few features in it. According to, a facial recognition study requires attention to speech, facial expressions, and posture. Given the picture alone, the challenge is to see the face in the picture. Face detection is a difficult task because the face changes size, shape, color, etc. and does not change. It becomes a difficult task of blurred image by being blocked by something other than the camera, and so on. The authors of think that the detection of a blurred face comes with two major challenges the unavailability of a very strong database containing both the covered or uncovered surface, and 2) the extraction without the appearance of the face in the covered area. Using a locally linear embedding (LLE) algorithm and dictionaries trained in a large pool of covered face, a cohesive surface of the world, a few misconceptions can be repeated and the height of facial expressions can be greatly reduced. According to the work reported in convolutional neural network (CNNs) in computer vision comes with a strict limit on the size of the input image.

A common practice is to rearrange images before uploading them to the network to bypass the block Here's a great job challenge to find the face in the photo correctly and indicate if you have a mask on it or not. In order to perform surveillance tasks, the proposed route should also see a face and a moving mask Section 3 Dataset Two datasets have been used for experimenting the current method. Dataset 1 consists of 1376 images in which 690 images with people wearing face masks and the rest 686 images with people who do not wear face mask mostly contains front face pose with single face in the frame and with same type of mask having white color only. Dataset 2 from Kaggle consists of 853 images and its countenances are clarified either with a mask or without a mask. In some face collections are head. This project uses OpenCV Caffe-based face detector, Keras, TensorFlow and MobileNetV2 to detect facial masks in humans. The database used contains 3835 images of which 1916 images have masked characters and 1919 masked characters. The first is a basic model. This is done using Keras and MobileNetV2. First a basic model is produced and a head model is made on top of that. The header model consists of a 128 horizontal network, a "Relu" activation function and a 0.5 drop followed by another 2 horizontal network and a "softmax" opening function. All three layers are combined, they will provide a model to be trained. The model produced into a training database with a label by dividing it into two parts. One section contains 75 percent of the images and is used for training. The remaining part contains 25 percent of the remaining images and is used to test model accuracy.

After the model has been trained, it can be used for facemask detection on a person's face. A trained model is uploaded and an image containing a face mask with or without a mask or continuous video streaming with people is provided as included. The image or video frame, if the input is a video stream, is first sent to the default face detector module to find the person's face. This is done by resizing the image or video frame first, then finding the blob in it. This detected bridge is sent to a face detector model that only removes the cut-off face of a person with no back. This face is given as part of the model we have trained before. This weather output is a mask or not. Another model is trained with human faces. Photos used for model training are provided with that person's name and email address as labels for those images.

CHAPTER-3

Working

This project makes the use of OpenCV, Caffe-based face detector, Keras, TensorFlow and MobileNetV2 for the detection of face mask on humans. The dataset which is beingused contains 3835 images out of which 1916 images have people with masks in them and 1919 people without masks in them.

First a base model is generated. This is done my using Keras and MobileNetV2. First abase model is generated and a head model is generated on top of that. The head model consists of a network with 128 layers, an activation function of "Relu" and a dropout of 0.5 followed by another network with 2 layers and an activation function "softmax". Allthese three layers combined, will give out model which will be trained.

The generated model is then trained with the labeled dataset by splitting it into two portions. One portion contains 75 percent images and it is used for training. The remaining portion contains the remaining 25 percent of images and is used for testing the model accuracy. After the model is trained, it can be used for detection of facemaskon human faces.

The trained model is loaded and image which contains human faces with or without masks or a continuous video stream with humans is given as input. The image or a frame of the video, in case the input is a video stream, is first sent to the default face detector module for the detection of human faces. This is done by resizing the image Or   the video frame first, followed by detecting the blob in it. This detected blob is sent to the face detector model which outputs only the cropped face of a person without the background. This face is given as the input to the model which we trained earlier. Thisoutputs weather there is a mask or not.

Another model is trained with the faces of humans. The images used for the training of the model are provided with the name and email address of that person as the labels of those images. This is done by using Open CV. When an input image is given to the CV model, it detects the face of a person and asks the user to provide the name and email address of that person which will be stored in the database. The output of the first modelis given as the input to this model. This face will be compared with the persons present in the database. And if his face matches, then a bounding box will be drawn over his face with his name on it and an email and Sms will be sent to him that he is not wearinga mask. Else, only the words “Mask” will be present below the bounding box if the person is wearing a mask and “No Mask” if the person is not wearing one.

Two datasets have been used for experimenting the current method. Dataset 1 consists of 1376 images in which 690 images with people wearing face masks and the rest 686 images with people who do not wear face masks. Fig. I mostly contains front face pose with single face in the frame and with same type of mask having white color only. Dataset 2 from Kaggle consists of 853 images and its countenances are clarified either with a mask or without a mask. In some face collections are head turn, tilt and slant with multiple faces in the frame and different types of masks having different colors as well. TensorFlow

MASK (DATA SETS)

The discovery of a face mask has become an important task of helping the international community. This paper introduces a simplified way to achieve this goal using basic machine learning packages such as TensorFlow, Keras. OpenCV and Scikit-Learn. The proposed method detects the surface in the image correctly and indicates whether it has a mask on it or not. As a security guard, it can also detect faces and moving masks. The method gets high accuracy up to 95.77% and 94.58% respectively on two separate databases.

The model is trained, certified and tested on two databases. According to database 1, the method achieves 95.77% accuracy shows how this precision precision reduces the cost of error. Database 2 is more flexible than Database I as it has more faces on the frame and different types of masks of different colors as well. Therefore, the model obtains 94 58% accuracy in database shows the difference between training and loss of validation associated with the database One of the main reasons for gaining this accuracy is in Max Pooling. It provides consistent translation on internal representation and a reduction in the number of parameters the model should study. This discretization-based sampling process enables input samples that comprise the image, by reducing their size. The number of neurons has a set value of 64 which is not very high. Too high a number of neurons and filters can lead to worse performance. Improved filter values and pool size_help to filter the main part (face) of the image to determine the presence of the mask correctly without causing it

SOURCE CODE

```python
# import the necessary packages
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
from imutils.video import VideoStream
import numpy as np
import imutils
import time
import cv2
import os

def detect_and_predict_mask(frame, faceNet, maskNet):
    # grab the dimensions of the frame and then construct a blob
    # from it
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),
        (104.0, 177.0, 123.0))

    # pass the blob through the network and obtain the face detections
    faceNet.setInput(blob)
    detections = faceNet.forward()
    print(detections.shape)

    # initialize our list of faces, their corresponding locations,
    # and the list of predictions from our face mask network
    faces = []
    locs = []
    preds = []

    # loop over the detections
    for i in range(0, detections.shape[2]):
        # extract the confidence (i.e., probability) associated with
        # the detection
        confidence = detections[0, 0, i, 2]

        # filter out weak detections by ensuring the confidence is
        # greater than the minimum confidence
        if confidence > 0.5:
            # compute the (x, y)-coordinates of the bounding box for
            # the object
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")
            # ensure the bounding boxes fall within the dimensions of
            # the frame
            (startX, startY) = (max(0, startX), max(0, startY))
            (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

            # extract the face ROI, convert it from BGR to RGB channel
            # ordering, resize it to 224x224, and preprocess it
            face = frame[startY:endY, startX:endX]
            face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
            face = cv2.resize(face, (224, 224))
            face = img_to_array(face)
            face = preprocess_input(face)

            # add the face and bounding boxes to their respective
            # lists
            faces.append(face)
            locs.append((startX, startY, endX, endY))

    # only make a predictions if at least one face was detected
    if len(faces) > 0:
        # for faster inference we'll make batch predictions on *all*
        # faces at the same time rather than one-by-one predictions
        # in the above `for` loop
        faces = np.array(faces, dtype="float32")
        preds = maskNet.predict(faces, batch_size=32)

    # return a 2-tuple of the face locations and their corresponding
    # locations
    return (locs, preds)

# load our serialized face detector model from disk
prototxtPath = r"face_detector\deploy.prototxt"
weightsPath = r"face_detector\res10_300x300_ssd_iter_140000.caffemodel"
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# load the face mask detector model from disk
maskNet = load_model("mask_detector.model")

# initialize the video stream
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()

# loop over the frames from the video stream
while True:
    # grab the frame from the threaded video stream and resize it
    # to have a maximum width of 400 pixels
    frame = vs.read()
    frame = imutils.resize(frame, width=400)

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

```python
    # face mask or not
    (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)

    # loop over the detected face locations and their corresponding
    # locations
    for (box, pred) in zip(locs, preds):
        # unpack the bounding box and predictions
        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred

        # determine the class label and color we'll use to draw
        # the bounding box and text
        label = "Mask" if mask > withoutMask else "No Mask"
        color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

        # include the probability in the label
        label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)

        # display the label and bounding box rectangle on the output
        # frame
        cv2.putText(frame, label, (startX, startY - 10),
            cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
        cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)

    # show the output frame
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    # if the `q` key was pressed, break from the loop
    if key == ord("q"):
        break

# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()
```

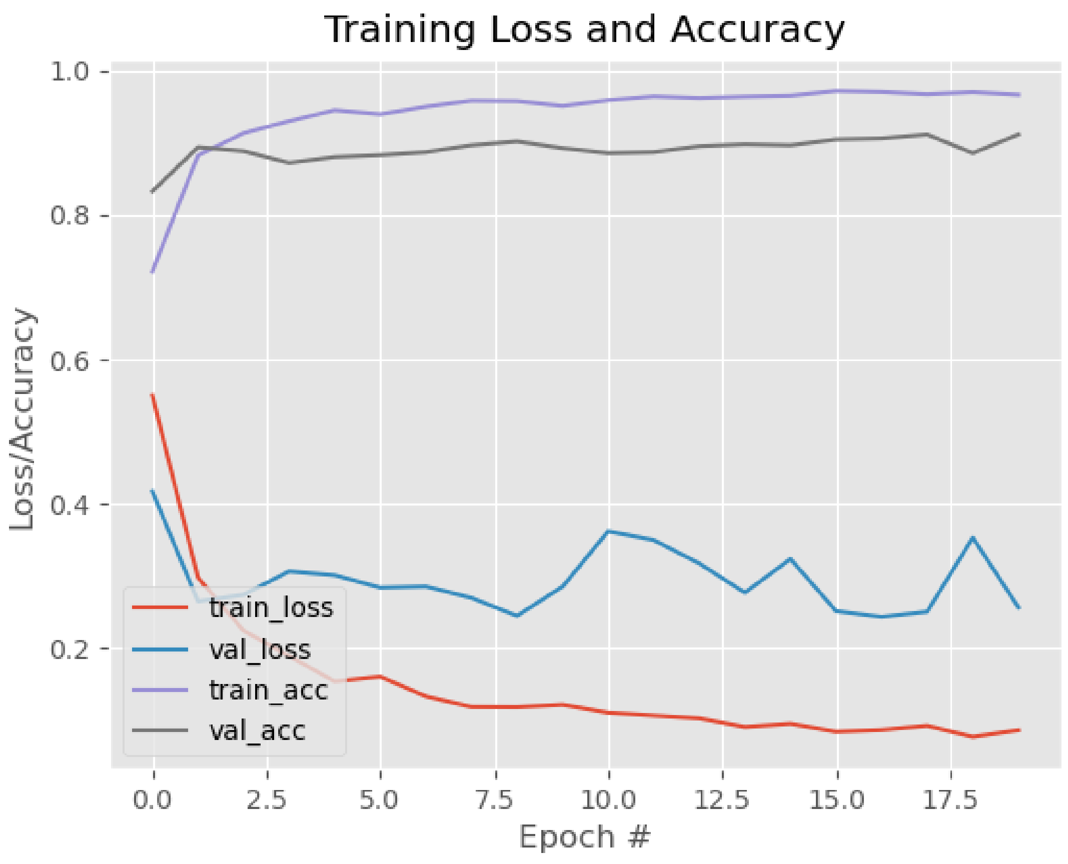
TRAINING

```python
# import the necessary packages
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import os

# initialize the initial learning rate, number of epochs to train for,
# and batch size
INIT_LR = 1e-4
EPOCHS = 20
BS = 32

DIRECTORY = r"C:\Mask Detection\CODE\Face-Mask-Detection-master\dataset"
CATEGORIES = ["with_mask", "without_mask"]

# grab the list of images in our dataset directory, then initialize
# the list of data (i.e., images) and class images
print("[INFO] loading images...")

data = []
labels = []

for category in CATEGORIES:
    path = os.path.join(DIRECTORY, category)
    for img in os.listdir(path):
        img_path = os.path.join(path, img)
        image = load_img(img_path, target_size=(224, 224))
        image = img_to_array(image)
        image = preprocess_input(image)
        data.append(image)
        labels.append(category)

# perform one-hot encoding on the labels
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)

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

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

# construct the training image generator for data augmentation
aug = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.15,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.15,
    horizontal_flip=True,
    fill_mode="nearest")

# load the MobileNetV2 network, ensuring the head FC layer sets are
# left off
baseModel = MobileNetV2(weights="imagenet", include_top=False,
    input_tensor=Input(shape=(224, 224, 3)))

# construct the head of the model that will be placed on top of the
# the base model
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(128, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)

# place the head FC model on top of the base model (this will become
# the actual model we will train)
model = Model(inputs=baseModel.input, outputs=headModel)

# loop over all layers in the base model and freeze them so they will
# *not* be updated during the first training process
for layer in baseModel.layers:
    layer.trainable = False

# compile our model
print("[INFO] compiling model...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
```

```python
model.compile(loss="binary_crossentropy", optimizer=opt,
    metrics=["accuracy"])

# train the head of the network
print("[INFO] training head...")
H = model.fit(
    aug.flow(trainX, trainY, batch_size=BS),
    steps_per_epoch=len(trainX) // BS,
    validation_data=(testX, testY),
    validation_steps=len(testX) // BS,
    epochs=EPOCHS)

# make predictions on the testing set
print("[INFO] evaluating network...")
predIdxs = model.predict(testX, batch_size=BS)

# for each image in the testing set we need to find the index of the
# label with corresponding largest predicted probability
predIdxs = np.argmax(predIdxs, axis=1)

# show a nicely formatted classification report
print(classification_report(testY.argmax(axis=1), predIdxs,
    target_names=lb.classes_))

# serialize the model to disk
print("[INFO] saving mask detector model...")
model.save("mask_detector.model", save_format="h5")

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

REQUIREMENTS

tensorflow>=1.15.2
keras==2.3.1
imutils==0.5.3
numpy==1.18.2
opencv-python==4.2.0.*
matplotlib==3.2.1
scipy==1.4.1



CHAPTER 4

RESULT AND DISCUSSION

The model is trained, validated and tested upon two datasets. Corresponding to dataset 1,the method attains accuracy up to 95.77% . depicts how this optimized accuracy mitigates the cost of error. Dataset 2 is more versatile than dataset 1 as it has multiple faces in the frame and different types of masks having different colors as well. Therefore, the model attains an accuracy of 94.58% on dataset depicts the contrast between training and validation loss correspondingto dataset . One of the main reasons behind achieving this accuracy lies in *Max Pooling*. It provides rudimentary translation invariance to the internal representation along with the reduction in the number of parameters the model has to learn. This sample-based discretization process down-samples the input representation consisting of image, by reducing its dimensionality. Number of neurons has the optimized value of 64 which is not too high. A much higher number of neurons and filters can lead to worse performance. The optimized filter values and pool_size help tofilter out the main portion (face) of the image to detect the existence of mask correctly without causing over-fitting.

This is done using an Open CV. When a photo is provided with a CV model, it identifies the person's face and asks the user to provide that person's name and email address to be stored on the website. The output of the first model is provided as input to this model. This face will be compared to the people present on the website. And if her face is the same, then a binding box will be drawn with her name and email and sms will be sent to her that she is not wearing a mask. Alternatively, only "Mask" words will appear below the check box if the person is wearing a mask and "No Mask" if the person is not wearing it

CHAPTER 5

CONCLUSION

We briefly explained the motivation of the work at first. Then, we illustrated the learning and performance task of the model. Using basic ML tools and simplified techniques the method has achieved reasonably high accuracy. It can be used for a variety of applications. Wearing a mask may be obligatory in the near future, considering the Covid-19 crisis. Many public service providers will ask the customers to wear masks correctly to avail of their services. The deployed model will contribute immensely to the public health care system. In future it can be extended to detect if a person is wearing the mask properly or not. The model can be further improved to detect if the mask is virus prone or not i.e. the type of the mask is surgical, N95 or not.

REFERENCES

[1] P.B. Kanade, and P. Gumaste, Brain tumor detection using MRI images. vol. Vol. 3. Brain, 2015.

[2]. A. Ben Rabeh, F. Benzarti, and H. Amiri, "Segmentation of brain MRI using active contour model",Int. J. Imaging Syst. Technol., vol. 27, no. 1, pp. 3-11, 2017.[http://dx.doi.org/10.1002/ima.22205]

[3]. K. Kamnitsas, C. Ledig, V.F.J. Newcombe, J.P. Simpson, A.D. Kane, D.K. Menon, D. Rueckert, andB. Glocker, "Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation", Med. Image Anal., vol. 36, pp. 61-78, 2017.[http://dx.doi.org/10.1016/j.media.2016.10.004] [PMID: 27865153]

[4]. K.B. Vaishnave, and K. Amshakala, "An automated MRI brain image segmentation and tumordetection using SOM-clustering and proximal support vector machine classifier", IEEE Int.Conf.Engineering and Technology (ICETECH), 2015[http://dx.doi.org/10.1109/ICETECH.2015.7275030]

[5]. N.J. Tustison, B.B. Avants, P.A. Cook, Y. Zheng, A. Egan, P.A. Yushkevich, and J.C. Gee, "N4ITK:improved N3 bias correction", IEEE Trans. Med. Imaging, vol. 29, no. 6, pp. 1310-1320, 2010.[http://dx.doi.org/10.1109/TMI.2010.2046908] [PMID: 20378467]

[6]. K. Siddiqi, Y.B. Lauzière, A. Tannenbaum, and S.W. Zucker, "Area and length minimizing flows forshape segmentation", IEEE Trans. Image Process., vol. 7, no. 3,

pp. 433-443, 1998.[http://dx.doi.org/10.1109/83.661193] [PMID: 18276263]

[7]. J.G. Sled, A.P. Zijdenbos, and A.C. Evans, "A nonparametric method for automatic correction ofintensity nonuniformity in MRI data", IEEE Trans. Med. Imaging, vol. 17, no. 1, pp. 87-97, 1998.[http://dx.doi.org/10.1109/42.668698] [PMID: 9617910] 39.

[8]. A. Parveen Singh, "Detection of brain tumor in MRI images, using combination of fuzzy Cmeans andSVM", 2nd Int. Conf. Signal Processing and Integrated Networks (SPIN), 2015 pp. 98- 102[http://dx.doi.org/10.1109/SPIN.2015.7095308]

[9]. Bhandary, A. Deep-learning framework to detect lung abnormality–A study with chest X-Ray and lung CT scan images. Pattern Recogn. Lett. 2020, 129, 271–278.

[10]. https://braintumor.org/wp-content/assets/WHO-Central-Nervous-System-TumorClassification. [11].https://www.dreamstime.com/brain-cancer-different-types-primary-brain-tumors-brain-cancerdifferent-types-primary-brain-tumors-categorized-image158760967