

DEVELOPMENT OF A FACE-MASK DETECTION SOFTWARE USING ARTIFICIAL INTELLIGENCE (AI) IN PYTHON FOR COVID-19 PROTECTION

Ahmed Imran Kabir, United International University
Sriman Mitra, United International University
Jakowan, United International University
Soumya Suhreed Das, Stamford University

ABSTRACT

Infectious diseases like Covid-19 transmits and contaminates via dispersal through aerial medium has become a serious issue nowadays that stimulated increasing the facemask usage. However, in densely populated countries it is very hard for law enforcing organizations to monitor the number of people wearing mask and those who are not wearing any in public. For this reason, a facemask detection model can be developed for security cameras in public places and surveil on people. In this research, researchers developed a prototype of facemask detector using python codes and artificial intelligence. Since Machine cannot understand human language, developers use different types of programming languages for training machine. Here, Python was used with the help of its packages to train models and two separate python files were developed to complete this research; the first one is to develop a mask detector and the other one to combine mask and face detector, combined together to develop facemask detector in real time. Data was collected, analyzed and visualized with python programming language, model was trained using ConvNet (Convolutional Neural Network) and finally an output was received from a raw input, which can detect mask in human face.

Keywords: Artificial intelligence; Machine learning; Python programming language; Covid-19.

INTRODUCTION

Science has established that daily works can be effectively done, even accelerated by many degrees by the use of AI (Artificial intelligence). A recent study has established that AI can be used for facial mask presence detection during the Covid-19 pandemic situation. Since wearing a mask is a matter of utmost importance for people to curb the rate of infection by the SARS Cov2 virus, the outcome of the study will definitely produce some positive impact in the global movement of reinforcing wearing a mask in public. This research is based on Artificial Intelligence (AI) with taking the help of deep learning method and machine learning. To conduct the research, the machine needed to be trained to use its own intelligence. However, since machine cannot understand any human languages and emotions, researchers needed to use a programming language. Here, 'Python Programming Language' was chosen for completing this research.

The main objectives of the study are:

1. To make a facemask detector prototype.

2. To monitor people in public places that they are following prevention protocol.
3. To make the facemask detection process from manual to automatic.
4. To lower costs and human effort.
5. To maximize the effectiveness of the process.
6. To develop knowledge and experience on machine learning and artificial intelligence

REVIEW OF THE LITERATURE

There is huge amount of previous studies available online that discussed about the Artificial Intelligence and related fields. Most of them showed the factors related to AI like deep learning, machine learning, CNN, ANN, MobileNet, programming languages and so on. As well as The importance of mask usage has been emphasized by entrepreunering universities in time of Covid-19, according to Salamzadeh and Dana (2020), and others as they conducted qualitative research by interviewing twenty-five experts from different countries in the Middle East, including Iran, Turkey, Iraq, United Arab Emirates, and the respondents were engaged in five online focus group sessions while the findings were coded. While researchers like Altınbaş et al. (2021), investigated if the Patients with COVID-19 under the Risk of Cardiovascular Events, and sclerosis (Ghajarzadeh et al., 2020). In this research, all the factors have been defined related to the thesis that other literature discussed about. While the disruption of various startup businesses due to COVID-19 were discussed in (Salamzadeh & Dana, 2020).

Python Programming Language

Sanner et al. (1999) mentioned that python is an interpreted, object-oriented and interactive as well as simple yet powerful general purpose programming language. He also told that various type of high-level data types are provided by python. Though there are some other objects can be found in python also. In addition, python has various kind of statements that are simple in nature (van Rossum & de Boer, 1991).

Artificial Intelligence

Researchers are always trying to make machine think by itself with the help of AI. AI is used for robotics, which is one of the sub areas of itself. AI can be used in medical fields as well, since Charniak (1985) mentioned that AI is the study of cognitive faculties using computational models. AI refers to provide intelligence to the machine so that it can act like human, solve problems using its own intelligence.

Machine Learning

Machine learning is a method of AI to choose for computer vision, controlling robot, and speech and face recognition and so on. Many AI developers consider that training a system by showing it examples is easier than doing it manually. There is a broad range of machine learning in the field of computer science (Jordan & Mitchell, 2015).

Deep Learning

Deep learning algorithms are a subset of machine learning algorithms. Computer vision, transfer learning, natural language processing etc. are the approaches of deep learning method (Guo et al., 2016). The conventional machine learning techniques had limitations, which Deep learning methods overcame. It advances in solving high-level problems, discovering intricate structures in high-level data, image and speech recognition and many more (LeCun et al., 2015). Deep learning can be successfully applied to analyze image and recognize target. For this, the non-uniformity of the shape, position and size of welding defects have impacts. Before that, it was a complicated task to analyze and evaluate the acquired welding defects images manually (Pan et al., 2020).

Face Detection

To develop a facemask detector, developing two individual detectors is necessary. At first, it is needed to develop a mask detector model individually and then a face detector model. Then both of them needed to be combined in a separate file, which will finally create a facemask detection prototype. Same spatial configuration, large components of non-rigidity and textural differences among faces make face recognition a difficult task, so, it is mandatory to train machine a lot by giving them more and more examples. It has also potential applications in human-computer interfaces and surveillance systems (Sung & Poggio, 1998).

Convolutional Neural Networks

CNN or ConvNet is a class of deep learning or deep neural network. It is used for analyzing visual imagery shown in Figure 1.

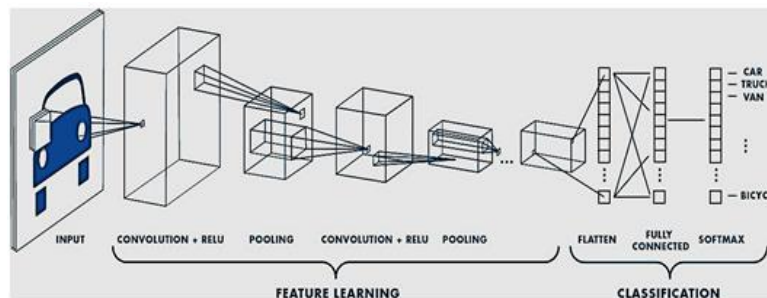


FIGURE 1

CONVOLUTIONAL NEURAL NETWORK

CNN is a simplified way of ANN. CNN analogous to traditional ANN in that they are comprised of neurons that self-optimize through learning (O'Shea & Nash, 2015). Convolutional neural network (CNN) is a part of deep neural network (DNN). In fact, it is one of the most popular among the DNNs. There are no parameters in pooling and non-linearity layers. However, parameters are present in convolutional and fully connected layers (Albawi et al., 2017). The datasets were collected from the GitHub account of 'Balaji Srinivas' which has been appreciated

in the reference (Srinivas, 2020). There are total 1915 images in the folder named 'with_mask'. Moreover, the 'without_mask' folder contains 1918 images.

RESEARCH METHODS

This research has been completed with the help of machine learning, precisely the deep learning methods. Here convolutional neural network has been used with a slight change.

Data Analysis Plan

As previously mentioned, the datasets contain different types images divided in two sub-folders. With the help of the data analyzed by Python, at first the machine was trained and a mask detector model was developed. A face detector was collected from online (Srinivas, 2020). After that, both the detectors were combined to develop facemask detector (Figures 2 & 3). After the development is completed, it was implemented into real time using laptop camera.



FIGURE 2

WITH MASK AND WITHOUT MASK DATASET

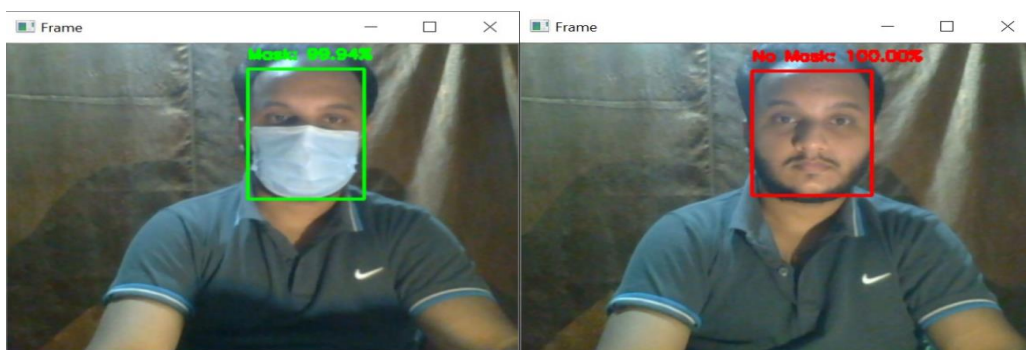
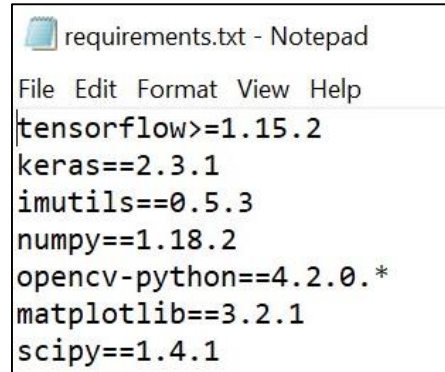


FIGURE 3

FACE MASK DETECTOR

RESEARCH ANALYSIS AND FINDINGS

To develop model and use the model in the real time camera, the following dependencies or python packages needed to be installed first shown in Figure 4.

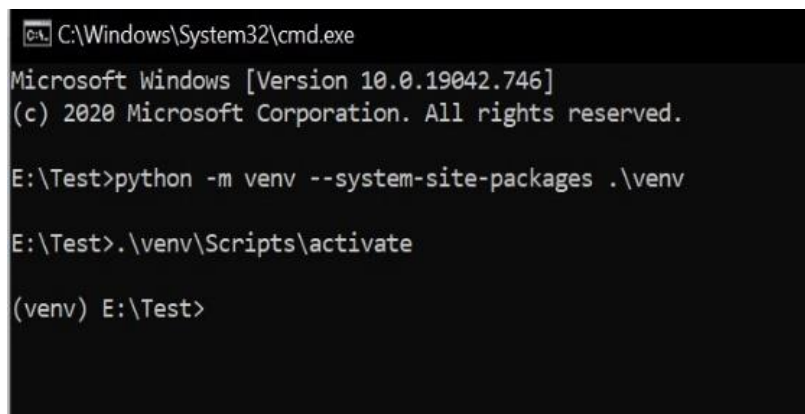


```
requirements.txt - Notepad
File Edit Format View Help
tensorflow>=1.15.2
keras==2.3.1
imutils==0.5.3
numpy==1.18.2
opencv-python==4.2.0.*
matplotlib==3.2.1
scipy==1.4.1
```

FIGURE 4

REQUIREMENTS

There are two ways to install this package. For installation purpose, a virtual environment needs to be created and activated to the working folder as the following Figure 5.



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19042.746]
(c) 2020 Microsoft Corporation. All rights reserved.

E:\Test>python -m venv --system-site-packages .\venv

E:\Test>.\venv\Scripts\activate

(venv) E:\Test>
```

FIGURE 5

CREATING AND ACTIVATING VIRTUAL ENVIRONMENT

Mask Detector Model

At first all the necessary packages were needed to be imported into the python file. Packages needed are shown in Figure 6 and Figure 7.

```

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

```

FIGURE 6

IMPORTING PACKAGES

```

DIRECTORY = r"D:\MIS Project\dataset"
CATEGORIES = ["with_mask", "without_mask"]

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)

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)

```

FIGURE 7

SETTING DIRECTORY & CATEGORIES, LOOPING THEM AND ENCODING LABELS

Here the directory was set and mentioned where the dataset folder is present. Therefore, machine can understand the dataset it need to use to run the program. Then categories were set where the values named have been mentioned as “*with_mask*” and “*without_mask*” which are the folders present in the DIRECTORY. After that, it has been looped through the CATEGORIES by using for command. By the command ‘*os.path.join*’ it was tried to loop through categories first. Then all the images were listed down into the particular directory by using ‘*listdir*’ command as

well as images were loaded using the preprocessing function `'load_img'` of the package `'keras'` was imported previously as well as gave the images a target size of (224, 224) which is the height and width of the images. Then all the images were converted into array by using another `'keras'` function called `'img_to_array'`. The `'mobilenet'` function of the package called `'tensorflow'` has been used. That is why researchers used `'preprocess_input'` function here. After preprocessing the images successfully, they were appended into the previously created `'data'` list and category into the `'labels'` list.

Then `'train_test_split'` was used to split the training and testing data. Here the test size is set as 0.20, which indicates 20% of the images had been given for testing purpose and the rest 80% for the training purpose. It is always good to give more data for training purpose to get a good test result. Stratification was used into labels which are nothing but classifying the labels and the `'random_state'` indicates the set of train and test split are being received. It actually does not matter what number is this and does not affect much on the split.

Training Model

To train the model used convolutional neural network have been used with a slight change. To understand the change, convolutional neural network (CNN) model needs to be understood first. Previously it was discussed briefly. MobileNet, a version of ConvNet has been used (Figure 8).

```
INIT_LR = 1e-4
EPOCHS = 20
BS = 32
```



FIGURE 8

MOBILENET NEURAL NETWORK AND PLOT (TRAINING ACCURACY AND LOSS)

Here, INIT_LR as $1e^{-4}$ has been provided which is the learning rate of this research. Keeping this rate less helps to calculate the loss properly, and by this getting a better accuracy is much easier. 20 EPOCHS was also been provided and batch size was given as 32. It was previously mentioned MobileNet was used which generated two models; one is the MobileNet model and the output of this model which created a regular model. After creating baseModel, the headModel was created using the output of baseModel. Then the pooling was created by using `'AveragePooling2D'` function. The size of pooling is 7/7 here. Flatten was added to the layer as

well as a Dense layer using 128 neurons, and the activation layer is mentioned here 'relu'. Relu is used for non-linear model like mine.

```

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")

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
print("[INFO] compiling model...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
    metrics=["acc"])
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)

```

FIGURE 9

DATA AUGMENTATION, CREATING BASE & HEAD MODEL, LOOPING BASE MODEL, COMPILING AND FITTING THEM

To avoid the overfitting Dropout has been used. Finally, the headModel has been created with two layers (with mask and without mask) and softmax as the activation value because softmax is the best fit for binary models. Now the model function was called. As mentioned previously, baseModel is the input model and headModel is the output model. The baseModel has been frozen by using 'for' loop, because it has been used instead of CNN and was to prevent it from running in the training. When both the models are ready, the models were needed to be compiled by giving them the initial learning rate, ADAM optimizer (a got to optimizer) and tracking the accuracy metrics. Then the model was needed to be fit. A function was created to plot the accuracy and loss of the model by using MATPLOTLIB. A model picture of the plotting is below showing the accuracy and loss of the model file. From the picture below it is quite evident that the accuracy level is quite good and it displays a constantly decreasing loss, which is also positive for the model. Therefore, the model can be said to be validated (Figure 9).

Using models in real time camera

Since the mask detector model have already been built a face detector model needed to be built to be used for both of the detectors to use in real time camera using mask detector model and the face detector a new python file needed to be created to use both the detectors in real time camera. Afterwards, a 'for' loop over the detection was created and the confidence level associated with the detection was extracted, which will filter out the weak detection by ensuring minimum confidence level properly. Also, the face ROI was extracted and had been converted it from BGR to RGB channel. Then the face was appended and the box was put into their respective lists.


```

def detect_and_predict_mask(frame, faceNet, maskNet):
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),
                                (104.0, 177.0, 123.0))

    faceNet.setInput(blob)
    detections = faceNet.forward()
    print(detections.shape)

    faces = []
    locs = []
    preds = []

    for i in range(0, detections.shape[2]):
        confidence = detections[0, 0, i, 2]
        if confidence > 0.5:
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")

```

FIGURE 10

DEFINING FRAME, FACE & MASK DETECTOR AND LOOPING OVER THE DETECTIONS

Here, it was ensured that the coordinates of the box would be into the frame and detect the face (Figures 10 & 11).

```

if len(faces) > 0:
    faces = np.array(faces, dtype="float32")
    preds = maskNet.predict(faces, batch_size=32)

    return (locs, preds)

prototxtPath = "D:\MIS_Project\face_detector\deploy.prototxt"
weightsPath =
"D:\MIS_Project\face_detector\res10_100x100_and_res14_0000.caffemodel"
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

maskNet = load_model("D:\MIS_Project\mask_detector.model")

print("[INFO] starting video stream...")
vs = VideoStream(cc=0).start()

while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=400)
    (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)

    for (box, pred) in zip(locs, preds):
        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred
        label = "Mask" if mask > withoutMask else "No Mask"
        color = (0, 255, 0) if label == "Mask" else (0, 0, 255)
        label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)
        cv2.putText(frame, label, (startX, startY - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 3)
        cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)

```

FIGURE 11

MAKING PREDICTIONS & LOADING THE FACE AND MASK DETECTOR MODEL

In addition to that the face detector files and the mask detector model have been loaded, which was previously been created using python coding. A path was given to them and they had been saved as the face detector in 'faceNet' variable and mask detector in 'maskNet' variable. The 'faceNet' variable was created using 'cv2.dnn.readNet' function, and the mask detector was loaded using 'load_model' function. Then "[INFO] starting video stream..." was printed to understand what is going on. Then a label was created for the prediction. If the person is wearing a mask it will show 'Mask' and if the person is not wearing mask it will show 'No Mask'. Here, the color of the rectangular box for mask was set as Green (0, 255, 0). For without mask, it will be Red (0, 0, 255). To understand the color, it should be known that machine only understand three basic color, called as BGR. Here, 0 defines completely negative and 255 defines completely positive. Therefore, for making green, Blue (van Rossum & de Boer, 1991) was set as 0, Green (G) as 255 and Red (R) as 0 and it produces completely green color. Same process has been followed for producing red color. Then the label was displayed using format string. It showed the maximum possible percentage for mask or without mask. Here maximum prediction for mask and No Mask will be above 90%, which will take upto 10% of error prediction. For not wearing mask properly,

it will show different values also. Here, the output of the frame has been shown and finally was breaking the loop. The 'q' button has been set to break the loop.

CONCLUSION

The focus of this research was to create a prototype of a Face-Mask Detector that might help to develop a real life research based on this research. Researchers tried to achieve highest accuracy possible in the model. By this research, we can understand things related to Machine Learning and Artificial Intelligence. Also, a lot can be learned about the python tools to develop such research. In addition, this research might help other researchers who might think of working on related research. In the end, it can be assumed that the goal of making this research has been fully utilized and achieved. Please see the Appendix for the source code of the AI model.

REFERENCES

- Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. *Proceedings of 2017 International Conference on Engineering and Technology (ICET)*, pp. 1-6.
- Altınbaş, Ö., Ertuş, İ. H., Mete, A. Ö., Demiryürek, Ş., Hafız, E., Saracaloğlu, A., Demiryürek, A. T. (2021). Comparison of the N-Terminal pro-brain natriuretic peptide levels, Neutrophil-to-Lymphocyte, Lymphocyte-to-Monocyte and Platelet-to-Lymphocyte ratios between the patients with COVID-19 and healthy subjects; are the patients with COVID-19 under the risk of cardiovascular events? *Authorea*, Preprints, 2021.
- Charniak, E. (1985). *Introduction to artificial intelligence*. Pearson Education India.
- Ghajarzadeh, M., Mirmosayyeb, O., Barzegar, M., Nehzat, N., Vaheb, S., Shaygannejad, V., Maghzi, A H. (2020). Favorable outcome after COVID-19 infection in a multiple sclerosis patient initiated on ocrelizumab during the pandemic. *Multiple Sclerosis and Related Disorders*, 43, 2020.
- Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., & Lew, M. S. (2016). Deep learning for visual understanding: A review. *Neurocomputing*, 187, 27-48.
- Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349, 255-260.
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521, 436-444.
- O'Shea, K., & Nash, R. (2015). *An introduction to convolutional neural networks*. arXiv:1511.08458v2.
- Pan, H., Pang, Z., Wang, Y., Wang, Y., & Chen, L. (2020). A new image recognition and classification method combining transfer learning algorithm and mobilenet model for welding defects. *IEEE Access*, 8, 119951-119960.
- Salamzadeh and L. P. Dana (2020). The coronavirus (COVID-19) pandemic: challenges among Iranian startups. *Journal of Small Business & Entrepreneurship*, 32, 1-24.
- Sanner, M. F. (1999). Python: a programming language for software integration and development. *Journal of molecular graphics & modelling*, 17(1), 57-61.
- Srinivas, B. (2020). *Face Mask Detection*. Retrieved from <https://github.com/balajisrinivas/Face-Mask-Detection>
- Sung, K.-K., & Poggio, T. (1998). Example-based learning for view-based human face detection. *IEEE Transactions on pattern analysis and machine intelligence*, 20, 39-51.
- van Rossum, G., & de Boer, J. (1991). Interactively testing remote servers using the Python programming language. *CWi Quarterly*, 4, 283-303.

APPENDIX

After completing all the EPOCHS something like the above picture will appear. Following are the codes used to develop this research.

Face Detector Model:

```
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

INIT_LR = 1e-4
EPOCHS = 20
BS = 32

DIRECTORY = r"D:\MIS_Project\dataset"
CATEGORIES = ["with_mask", "without_mask"]

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)

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)

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")

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

print("[INFO] compiling model...")
```

```
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
              metrics=["acc"])

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)

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_))

print("[INFO] saving mask detector model...")
model.save("mask_detector.model", save_format="h5")

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("plot.png")
```

Facemask Detector:

```
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):

    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),
                                (104.0, 177.0, 123.0))

    faceNet.setInput(blob)
    detections = faceNet.forward()
    print(detections.shape)

    faces = []
    locs = []
    preds = []

    for i in range(0, detections.shape[2]):
        confidence = detections[0, 0, i, 2]
        if confidence > 0.5:
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")
            (startX, startY) = (max(0, startX), max(0, startY))
            (endX, endY) = (min(w - 1, endX), min(h - 1, endY))
            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)
```



```

    faces.append(face)
    locs.append((startX, startY, endX, endY))

if len(faces) > 0:
    faces = np.array(faces, dtype="float32")
    preds = maskNet.predict(faces, batch_size=32)
    return (locs, preds)

prototxtPath = researchers"D:\MIS_Project\face_detector\deploy.prototxt"
weightsPath = researchers"D:\MIS_Project\face_detector\res10_300x300_ssd_iter_140000.caffemodel"
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)
maskNet = load_model("D:\MIS_Project\mask_detector.model")

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

while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=400)
    (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)

    for (box, pred) in zip(locs, preds):
        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred
        label = "Mask" if mask > withoutMask else "No Mask"
        color = (0, 255, 0) if label == "Mask" else (0, 0, 255)
        label = "[ ]: {:.2f}%".format(label, max(mask, withoutMask) * 100)
        cv2.putText(frame, label, (startX, startY - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
        cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    if key == ord("q"):
        break

cv2.destroyAllWindows()
vs.stop()

```