MULTIPLE FACE MASK DETECTION USING DEEP LEARNING

Bachelor of Technology
in
Computer Science and Engineering

Rajiv Gandhi University of Knowledge and Technologies
RK Valley Kadapa

Submitted By
Y HariPriya --R170498
K Saikeerthana --R170225

Under the Esteemed Guidance of
Mr.Sathyanandaram N
RGUKT RK Valley

DECLARATION

We hereby declare that the report of the B.Tech Major Project Work entitled “MULTIPLE FACE MASK DETECTION USING DEEP LEARNING” which is being submitted to Rajiv Gandhi University of Knowledge Technologies, RK Valley, in partial fulfillment of the requirements for the award of Degree of Bachelor of Technology in Computer Science and Engineering, is a bonafide report of the work carried out by us. The material contained in this report has not been submitted to any university or institution for award of any degree.

Y HariPriya – R170498
K Saikeerthana– R170225
Dept. Of Computer Science and Engineering.

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

RGUKT

CERTIFICATE FOR PROJECT COMPLETION

This is certify that the project entitled " MULTIPLE FACE MASK DETECTION USING DEEP LEARNING " submitted by Y Hari Priya(R170498), K Saikeerthana(R170225) under our guidance and supervision for the partial fulfillment for the degree Bachelor of Technology in Computer Science and Engineering during the academic semester- -2 2022-2023 at RGUKT, RK VALLEY. To the best of my knowledge, the results embodied in this dissertation work have not been submitted to any University or Institute for the award of any degree or diploma.

Project Internal Guide	Head of the Department	Project External Guide
Mr.N.Satya Nandaram	Mr.N.Satya Nandaram	
Assistant Professor	Assistant Professor	

INDEX

1.ABSTRACT

In order to prevent the spread of CORONA virus, everyone must wear a mask during the pandemic. In December 2019, the outbreak of pneumonia with an unknown cause was discovered in Wuhan, Hubei Province, China. The virus that caused this outbreak was later found and named ‘Severe Acute Respiratory Syndrome’ Coronavirus 2 (SARS-CoV-2).In Covid-19 situation the major protective measure for individuals is to wear masks in public areas. Several regions applied a compulsory mask-wearing rule in public areas to prevent transmission of the virus.

Xinqi Fan et. al. used the significance of novel single-shot light-weight face mask detector (SL-FMDet). And to extract rich context features and focus on crucial face mask related regions by using Novel Residual Context Attention Module (RCAM) and Synthesized Gaussian Heatmap Regression (SGHR). YOLOv3-tiny is used to compare the both datasets (AIZOO, Moxa3K).This project aims at improving the performance by using CenterNet deep learning detector.

2.EXISTING SYSTEM

Millions of people are being infected by Corona Virus throughout the world rapidly. Even WHO recommends to wear a mask in public places to cut the spread of virus as it is contagious. In this paper, a CNN model is proposed for face mask detection. Object detection when applied on faces helps in detecting faces in the image. Face mask detection refers to detecting faces in the image and then classifying each face as with mask or without mask. In existing face mask detection which are designed using different models detects only single face at a time.

Disadvantages

- No 24/7 is Monitored.
- It is difficult to handle person to identify person with mask or without mask in surveillance.
- Detects only single face.
- Separate Manpower is need for Fine Calculations and other punishments.

3. PROPOSED SYSTEM

This project proposes a method to detect the face mask is put on or not for offices, or any other work place with a lot of people coming to work. We have used convolutional neural network for the same. The model is trained on a real world dataset and tested with live video streaming with a good accuracy. Further the accuracy of the model with different hyper parameters and multiple people at different distance and location of the frame is done. The proposed CNN, classifies faces with and without masks as the output layer of proposed CNN architecture contains two neurons with Softmax activation to classify the same. Categorical cross-entropy is employed as loss function. The proposed model has Validation accuracy of 96%. If anyone in the video stream is not wearing a protective mask a Red coloured rectangle is drawn around the face with a dialog entitled as NO MASK and a Green coloured rectangle is drawn around the face of a person wearing MASK. If anyone in the video stream is not wearing a protective mask and a Red colored rectangle is drawn around the face with a

dialog entitled as NO MASK. Similarly a Green color rectangle is drawn around the face of a person wearing MASK.

Project aims at improving the performance by using CenterNet deep learning detector. Categorical cross-entropy is employed as loss function. The proposed model has Validation accuracy of 96%. If anyone in the video stream is not wearing a protective mask a Red coloured rectangle is drawn around the face with a dialog entitled as NO MASK and a Green coloured rectangle is drawn around the face of a person wearing MASK.

Advantages

- 24/7 is Monitored.
- Less Expenses In Implementation as Code can connected to all CC Cameras.
- Detects Multiple Faces at a time.
- Automatic Detection of Multiple Faces with or without masks.
- The proposed model has Validation accuracy of 96%. If anyone in the video stream is not wearing a protectiv

mask a Red coloured rectangle is drawn around the face with a dialog entitled as NO MASK and a Green coloured rectangle is drawn around the face of a person wearing MASK.

MFMD SRS Document

4.Introduction:

This document has the requirements of Multiple Face Mask Detection using Deep Learning project.This project is used to detect the faces with masks and without masks in a crowd or in offices,super markets and shopping malls etc.

4.1. Purpose:

The purpose of this document is to gather the requirements that are needed for implementing the MFMD. It also focuses on various key features, the product,product vision and scope,product overview.The main purpose of MFMD is to detect the faces that are with masks and without masks in a group of people.

4.2. Inteded Audiance:

The intended audian will be the user who can provide dataset (the dataset contains the video stream of people) to the program which detects the faces with mask with green mark and faces without mask with red mark.

Users:

1. User
   -Can be a hotel manager.
   -Can be a clinic operator.
   -Any other officer who can provide a stream of video as input.

4.3 Product Vision:

The product vision is to implement a system that can take input of a dataset (the dataset contains the video stream of people) by the user which can detect the faces with masks with green mark and faces without masks with green mark.

4.4 Technologies:

1.Python:-

Below are some facts about Python. Python is currently the most widely used multi-purpose, high-level

programming language. Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java. Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time. Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc. The biggest strength of Python is huge collection of standard library which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Image processing (like Opencv, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

2.Machine Learning

ML Tools used:

Numpy:

NumPy is a library for the Python Programming language, adding support for large, multi- dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. In 2005,TravisOliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source softwareand has many contributors.

Pandas:

pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

Matplotlib:

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack.Matplotlib consists of several plots like line, bar, scatter, histogram etc.

Scikit Learn:

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence

interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

Seaborn:

Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and color palettes to make statistical plots more attractive. It is built on the top of matplotlib library and also closely integrated to the data structures from pandas.

3.Flask:

Flask is a micro web framework written in Python. It is classified as a micro-framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions .However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation,upload handling, various open authentication technologies and several common framework related tools

5.Algorithm Used to design the MFMD System:-

YOLO ALGORITHM:-

YOLO is an abbreviation for the term ‘You Only Look Once’. This is an algorithm that detects and recognizes various objects in a picture (in real-time). Object detection in YOLO is done as a regression problem and provides the class probabilities of the detected images.

YOLO algorithm employs convolutional neural networks (CNN) to detect objects in real-time. As the name suggests, the algorithm requires only a single forward propagation through a neural network to detect objects.

YOLO is an algorithm that uses neural networks to provide real-time object detection. This algorithm is popular because of its speed and accuracy. It has been used in various applications to detect traffic signals, people, parking meters, and animals.

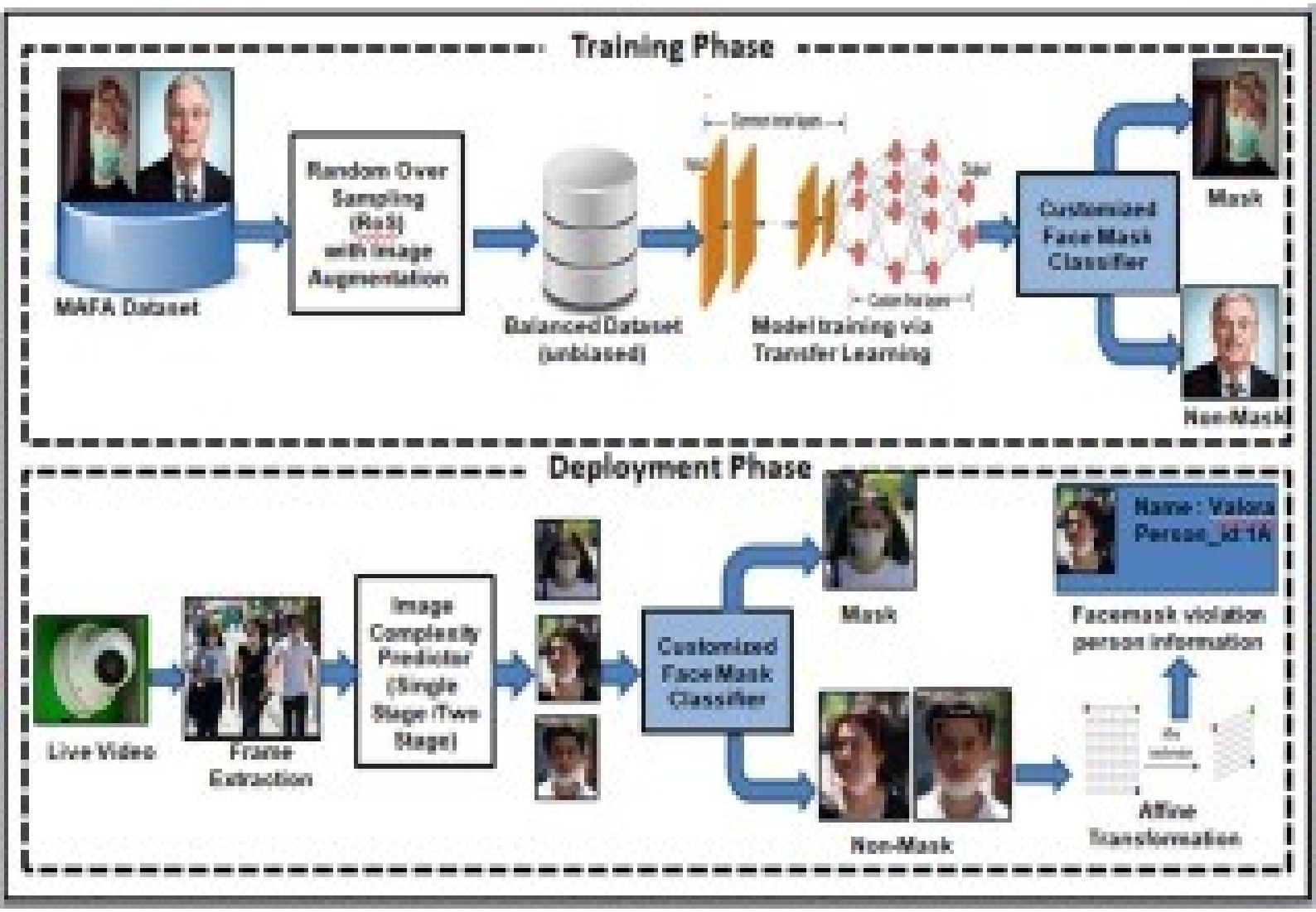
YOLO is an algorithm used for 2 types of detection.

1.object detection

2.face detection

In this project we use object detection yolo algorithm to process and detect the faces that are masked and unmasked with green and red marks respectively.

6.Generating Model



7.Requirement Specification:

Hardware Configuration:

i)Ram At least 4 GB

ii)Hard disk 10 GB

iii)Processor Intel i3 or more 2.00FHZ

Software Requirement:

i)Front end HTML,CSS

ii)Server side Language FLASK

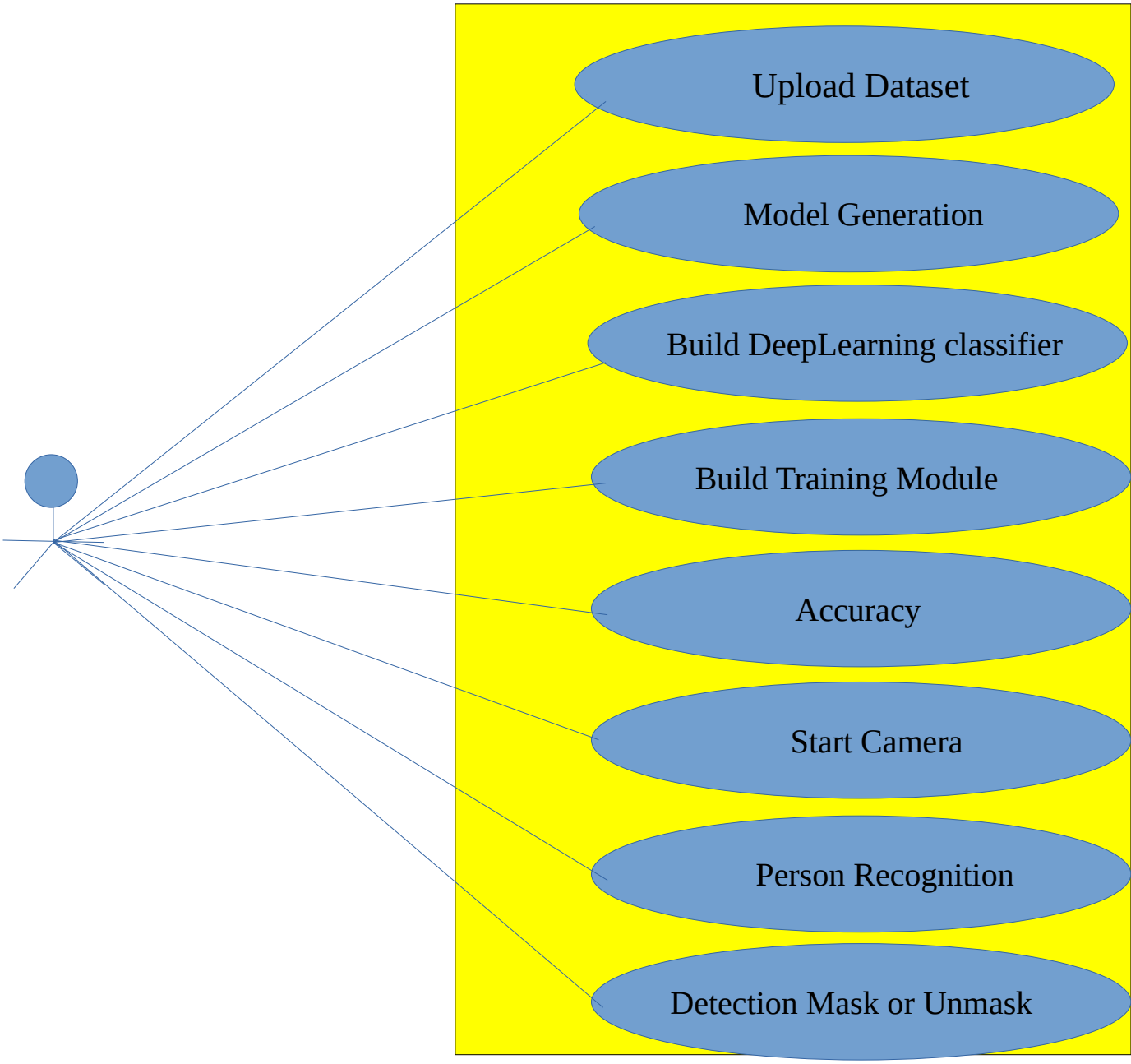
iii)Web Browser Operating System Firefox , Google Chrome or any compatible browser

iv)Ubuntu,Windows or any equivalent OS

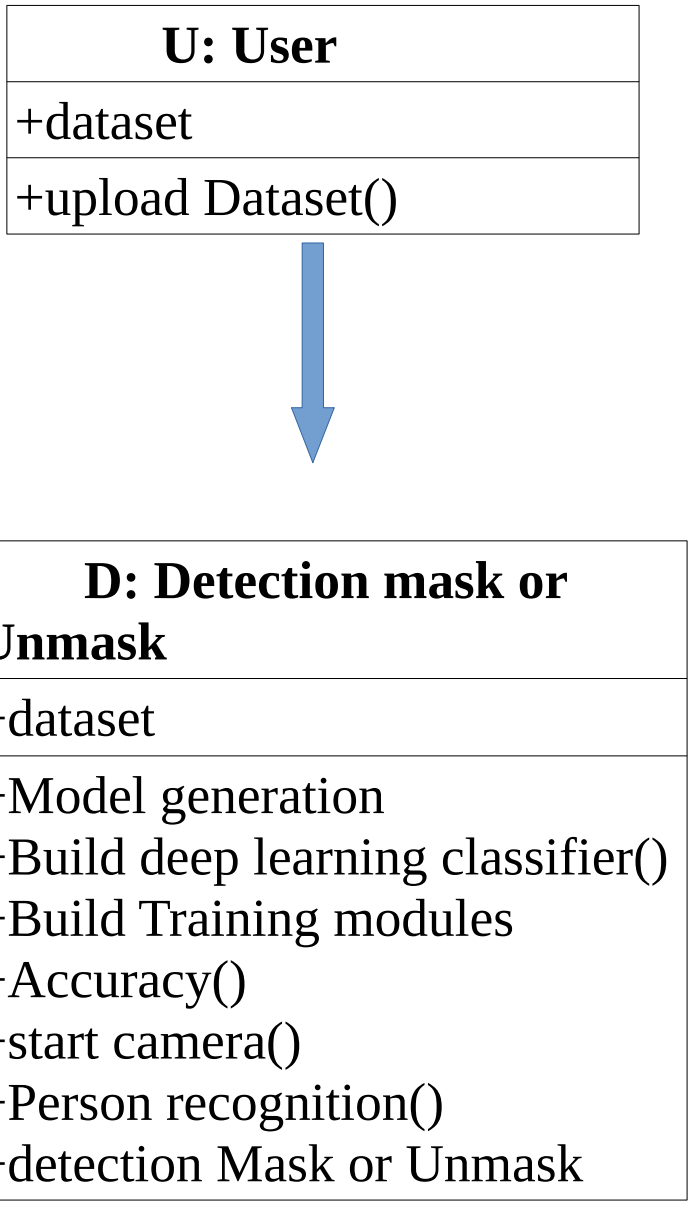
v)Software xampp

8.Use case Diagram:

MFMD



9.Class Diagram:-



10.SOURCE CODE

MAIN.PY

```
import time

import threading

try:

    from greenlet import getcurrent as get_ident

except ImportError:

    try:

        from thread import get_ident except ImportError:

        from _thread import get_ident



class CameraEvent(object):

    """An Event-like class that signals all active clients when

a new frame is
```

```
    available.
    """
    def __init__(self):
        self.events = {}
def wait(self):
        """Invoked from each client's thread to wait for the
next frame."""
        ident = get_ident()
        if ident not in self.events:
            # this is a new client
            # add an entry for it in the self.events dict
            # each entry has two elements, a threading.Event()
and a timestamp
            self.events[ident] = [threading.Event(), time.time()]
        return self.events[ident][0].wait()
```

```
 def set(self):
        """Invoked by the camera thread when a new frame is
available."""
        now = time.time()
        remove = None
        for ident, event in self.events.items():
            if not event[0].isSet():
                # if this client's event is not set, then set it
                # also update the last set timestamp to now
                event[0].set()
                event[1] = now
            else:
                # if the client's event is already set, it means the
client
                # did not process a previous frame
```

```
            # if the event stays set for more than 5 seconds,
then assume
            # the client is gone and remove it
            if now - event[1] > 5:
                remove = ident
        if remove:
            del self.events[remove]
 def clear(self):
        """Invoked from each client's thread after a frame was
processed."""
        self.events[get_ident()][0].clear()
class BaseCamera(object):
    thread = None  # background thread that reads frames
from camera
```

```
    frame = None  # current frame is stored here by
background thread
    last_access = 0  # time of last client access to the camera
    event = CameraEvent()
 def __init__(self):
        """Start the background camera thread if it isn't
running yet."""
        if BaseCamera.thread is None:
            BaseCamera.last_access = time.time()
 # start background frame thread
            BaseCamera.thread =
threading.Thread(target=self._thread)
            BaseCamera.thread.start()
# wait until frames are available
            while self.get_frame() is None:
```

```
                time.sleep(0)
 def get_frame(self):
        """Return the current camera frame."""
        BaseCamera.last_access = time.time()
# wait for a signal from the camera thread
        BaseCamera.event.wait()
        BaseCamera.event.clear()
  return BaseCamera.frame
 @staticmethod
    def frames():
        """Generator that returns frames from the camera."""
        raise RuntimeError('Must be implemented by
subclasses.')


    @classmethod
```

```
  def _thread(cls):
      """Camera background thread."""
      print('Starting camera thread.')
      frames_iterator = cls.frames()
      for frame in frames_iterator:
          BaseCamera.frame = frame
          BaseCamera.event.set()  # send signal to clients
          time.sleep(0)
# if there hasn't been any clients asking for frames in
          # the last 10 seconds then stop the thread
          if time.time() - BaseCamera.last_access > 10:
              frames_iterator.close()
              print('Stopping camera thread due to inactivity.')
              break
      BaseCamera.thread = None
```

```
# USAGE
# python train_mask_detector.py --dataset dataset
# 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 argparse
import os
# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
```

```
ap.add_argument("-d", "--dataset", required=True,
	help="path to input dataset")
ap.add_argument("-p", "--plot", type=str,
default="plot.png",
	help="path to output loss/accuracy plot")
ap.add_argument("-m", "--model", type=str,
	default="mask_detector.model",
	help="path to output face mask detector model")
args = vars(ap.parse_args())
# initialize the initial learning rate, number of epochs to
train for,
# and batch size
INIT_LR = 1e-4
EPOCHS = 20
BS = 32
```

```
# 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...")
imagePaths = list(paths.list_images(args["dataset"]))
data = []
labels = []
# loop over the image paths
for imagePath in imagePaths:
    # extract the class label from the filename
    label = imagePath.split(os.path.sep)[-2]
# load the input image (224x224) and preprocess it
    image = load_img(imagePath, target_size=(224, 224))
    image = img_to_array(image)
    image = preprocess_input(image)
```

```
# update the data and labels lists, respectively
        data.append(image)
        labels.append(label)


# convert the data and labels to NumPy arrays
data = np.array(data, dtype="float32")
labels = np.array(labels)
# perform one-hot encoding on the labels
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)
# partition the data into training and testing splits using
75% of
# the data for training and the remaining 25% for testing
(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)
model.compile(loss="binary_crossentropy", optimizer=opt,
	metrics=["accuracy"])
# train the head of the network
print("[INFO] training head...")
```

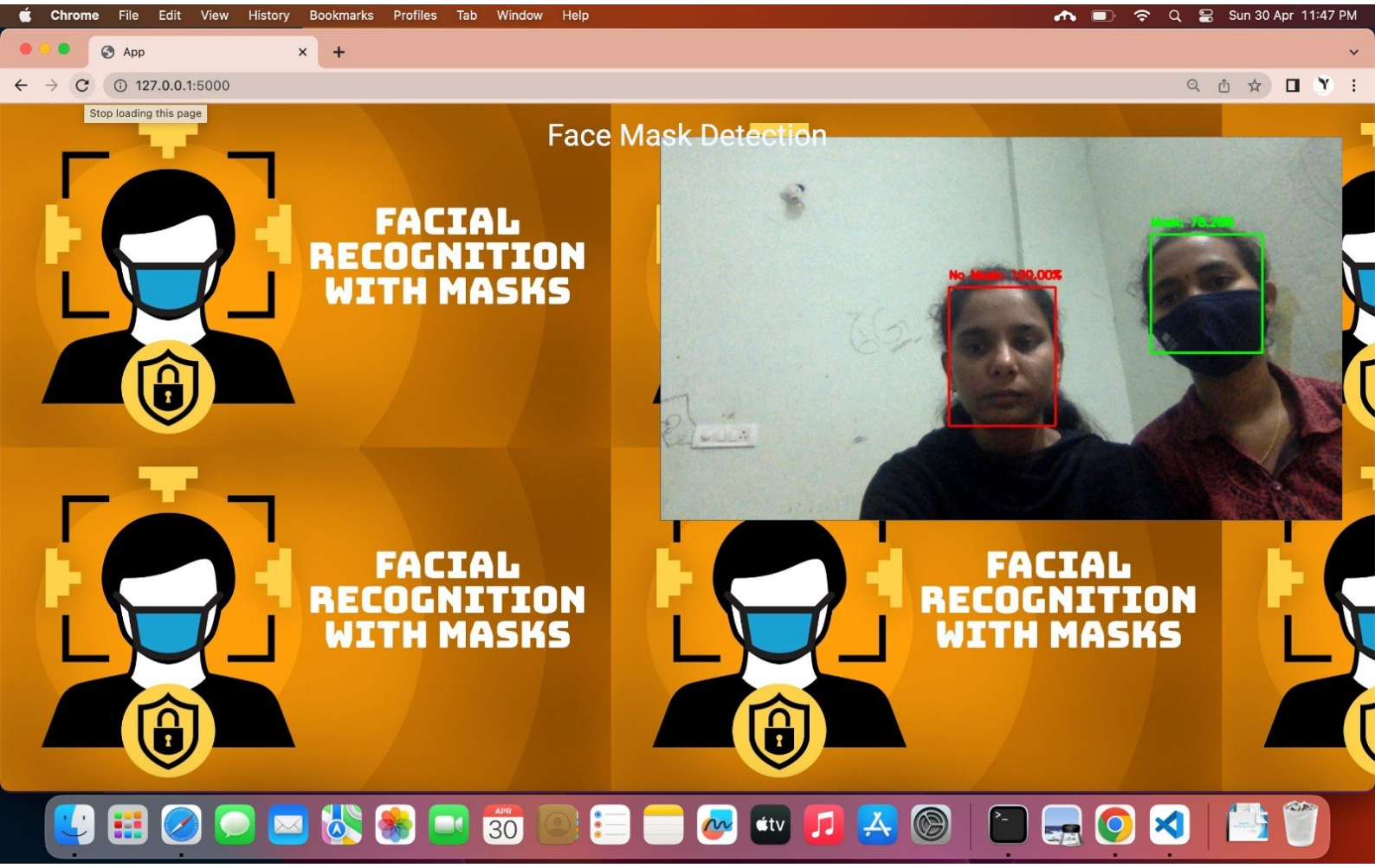
```
H = model.fit(
	aug.flow(trainX, trainY, batch_size=BS),
	steps_per_epoch=len(trainX) // BS,
	validation_data=(testX, testY),
	validation_steps=len(testX) // BS,
	epochs=EPOCHS)
# make predictions on the testing set
print("[INFO] evaluating network...")
predIdxs = model.predict(testX, batch_size=BS)
# 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(args["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(args["plot"])
```

11.OutPut Screens:-

1.



2.


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Sun 30 Apr 11:46 PM
App
127.0.0.1:5000
Stop loading this page
Face Mask Detection
No Mask: 99.94%
FACIAL RECOGNITION WITH MASKS
FACIAL RECOGNITION WITH MASKS
FACIAL RECOGNITION WITH MASKS

3.


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Sun 30 Apr 11:47 PM
App
127.0.0.1:5000
Stop loading this page
Face Mask Detection
FACIAL RECOGNITION WITH MASKS
FACIAL RECOGNITION WITH MASKS
FACIAL RECOGNITION WITH MASKS
No Mask 100.00%
No Mask 100.00%

12 References:-

1.https://neptune.ai/

2. https://www.shiksha.com/online-courses/articles/top-platforms-to-learn-data-science-and-machine-learning/