```
from tkinter import *
import tkinter
from tkinter import filedialog
import numpy as np
from tkinter.filedialog import askopenfilename
import pandas as pd
from tkinter import simpledialog
import numpy as np
import cv2 as cv
import subprocess
import time
import os
from yoloDetection import detectObject, displayImage
import sys
from time import sleep
from tkinter import messagebox
import pytesseract as tess
from keras.models import model_from_json
from keras.utils.np_utils import to_categorical

main = tkinter.Tk()
main.title("Helmet Detection") #designing main screen
main.geometry("800x700")

global filename
global loaded_model

global class_labels
global cnn_model
```

```
global cnn_layer_names

frame_count = 0
frame_count_out=0

confThreshold = 0.5
nmsThreshold = 0.4
inpWidth = 416
inpHeight = 416
global option
labels_value = []
with open("Models/labels.txt", "r") as file: #reading MRC dictionary
    for line in file:
        line = line.strip('\n')
        line = line.strip()
        labels_value.append(line)
    file.close()



with open('Models/model.json', "r") as json_file:
    loaded_model_json = json_file.read()
    plate_detecter = model_from_json(loaded_model_json)

plate_detecter.load_weights("Models/model_weights.h5")
plate_detecter._make_predict_function()

classesFile = "Models/obj.names";
classes = None
```

```
with open(classesFile, 'rt') as f:
    classes = f.read().rstrip('\n').split('\n')

modelConfiguration = "Models/yolov3-obj.cfg";
modelWeights = "Models/yolov3-obj_2400.weights";

net = cv.dnn.readNetFromDarknet(modelConfiguration, modelWeights)
net.setPreferableBackend(cv.dnn.DNN_BACKEND_OPENCV)
net.setPreferableTarget(cv.dnn.DNN_TARGET_CPU)

def getOutputsNames(net):
    layersNames = net.getLayerNames()
    return [layersNames[i[0] - 1] for i in net.getUnconnectedOutLayers()]

def loadLibraries(): #function to load yolov3 model weight and class labels
        global class_labels
        global cnn_model
        global cnn_layer_names
        class_labels = open('yolov3model/yolov3-labels').read().strip().split('\n') #reading labels from yolov3 model
        print(str(class_labels)+" == "+str(len(class_labels)))
        cnn_model = cv.dnn.readNetFromDarknet('yolov3model/yolov3.cfg', 'yolov3model/yolov3.weights') #reading model
        cnn_layer_names = cnn_model.getLayerNames() #getting layers from cnn model
        cnn_layer_names = [cnn_layer_names[i[0] - 1] for i in cnn_model.getUnconnectedOutLayers()] #assigning all layers

def upload(): #function to upload tweeter profile
    global filename
```

```
    filename = filedialog.askopenfilename(initialdir="bikes")
    #messagebox.showinfo("File Information", "image file loaded")



def detectBike():
    global option
    option = 0
    indexno = 0
    label_colors = (0,255,0)
    try:
        image = cv.imread(filename)
        image_height, image_width = image.shape[:2]
    except:
        raise 'Invalid image path'
    finally:
        image, ops = detectObject(cnn_model, cnn_layer_names, image_height, image_width, image,
label_colors, class_labels,indexno)
        if ops == 1:
            displayImage(image,0)#display image with detected objects label
            option = 1
        else:
            displayImage(image,0)




def drawPred(classId, conf, left, top, right, bottom,frame,option):
    global frame_count
```

```
    #cv.rectangle(frame, (left, top), (right, bottom), (255, 178, 50), 3)
    label = '%.2f' % conf
    if classes:
        assert(classId < len(classes))
        label = '%s:%s' % (classes[classId], label)
    labelSize, baseLine = cv.getTextSize(label, cv.FONT_HERSHEY_SIMPLEX, 0.5, 1)
    top = max(top, labelSize[1])
    label_name,label_conf = label.split(':')
    print(label_name+" === "+str(conf)+"==  "+str(option))
    if label_name == 'Helmet' and conf > 0.50:
        if option == 0 and conf > 0.90:
            cv.rectangle(frame, (left, top - round(1.5*labelSize[1])), (left + round(1.5*labelSize[0]), top +
baseLine), (255, 255, 255), cv.FILLED)
            cv.putText(frame, label, (left, top), cv.FONT_HERSHEY_SIMPLEX, 0.75, (0,0,0), 1)
            frame_count+=1
        if option == 0 and conf < 0.90:
            cv.putText(frame, "Helmet Not detected", (10, top), cv.FONT_HERSHEY_SIMPLEX, 0.75, (0,255,0),
2)
            frame_count+=1
            img = cv.imread(filename)
            img = cv.resize(img, (64,64))
            im2arr = np.array(img)
            im2arr = im2arr.reshape(1,64,64,3)
            X = np.asarray(im2arr)
            X = X.astype('float32')
            X = X/255
            preds = plate_detecter.predict(X)
            predict = np.argmax(preds)
            #img = cv.imread(filename)
```

```
            #img = cv.resize(img,(500,500))
            #text = tess.image_to_string(img, lang='eng')
            #text = text.replace("\n"," ")
            #messagebox.showinfo("Number Plate Detection Result", "Number plate detected as "+text)
            textarea.insert(END,filename+"\n\n")
            textarea.insert(END,"Number plate detected as "+str(labels_value[predict]))
        if option == 1:
            cv.rectangle(frame, (left, top - round(1.5*labelSize[1])), (left + round(1.5*labelSize[0]), top + baseLine), (255, 255, 255), cv.FILLED)
            cv.putText(frame, label, (left, top), cv.FONT_HERSHEY_SIMPLEX, 0.75, (0,0,0), 1)
            frame_count+=1


    if(frame_count> 0):
        return frame_count


def postprocess(frame, outs, option):
    frameHeight = frame.shape[0]
    frameWidth = frame.shape[1]
    global frame_count_out
    frame_count_out=0
    classIds = []
    confidences = []
    boxes = []
    classIds = []
    confidences = []
    boxes = []
    cc = 0
    for out in outs:
```

```
    for detection in out:
        scores = detection[5:]
        classId = np.argmax(scores)
        confidence = scores[classId]
        if confidence > confThreshold:
            center_x = int(detection[0] * frameWidth)
            center_y = int(detection[1] * frameHeight)
            width = int(detection[2] * frameWidth)
            height = int(detection[3] * frameHeight)
            left = int(center_x - width / 2)
            top = int(center_y - height / 2)
            classIds.append(classId)
            #print(classIds)
            confidences.append(float(confidence))
            boxes.append([left, top, width, height])

    indices = cv.dnn.NMSBoxes(boxes, confidences, confThreshold, nmsThreshold)
    count_person=0 # for counting the classes in this loop.
    for i in indices:
        i = i[0]
        box = boxes[i]
        left = box[0]
        top = box[1]
        width = box[2]
        height = box[3]
        frame_count_out = drawPred(classIds[i], confidences[i], left, top, left + width, top +
height,frame,option)
        my_class='Helmet'
        unknown_class = classes[classId]
```

```
            print("===="+str(unknown_class))
            if my_class == unknown_class:
                count_person += 1


        print(str(frame_count_out))
        if count_person == 0 and option == 1:
            cv.putText(frame, "Helmet Not detected", (10, 50), cv.FONT_HERSHEY_SIMPLEX, 0.75, (0,255,0), 2)
        if count_person >= 1 and option == 0:
            #path = 'test_out/'
            #cv.imwrite(str(path)+str(cc)+".jpg", frame)     # writing to folder.
            #cc = cc + 1
            frame = cv.resize(frame,(500,500))
            cv.imshow('img',frame)
            cv.waitKey(50)



def detectHelmet():
    textarea.delete('1.0', END)
    if option == 1:
        frame = cv.imread(filename)
        frame_count =0
        blob = cv.dnn.blobFromImage(frame, 1/255, (inpWidth, inpHeight), [0,0,0], 1, crop=False)
        net.setInput(blob)
        outs = net.forward(getOutputsNames(net))
        postprocess(frame, outs,0)
        t, _ = net.getPerfProfile()
        label = 'Inference time: %.2f ms' % (t * 1000.0 / cv.getTickFrequency())
        print(label)
        cv.putText(frame, label, (0, 15), cv.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255))
```

```
        print(label)
    else:
        messagebox.showinfo("Person & Motor bike not detected in uploaded image", "Person & Motor bike not detected in uploaded image")



def videoHelmetDetect():
    global filename
    videofile = askopenfilename(initialdir = "videos")
    video = cv.VideoCapture(videofile)
    while(True):
        ret, frame = video.read()
        if ret == True:
            frame_count = 0
            filename = "temp.png"
            cv.imwrite("temp.png",frame)
            blob = cv.dnn.blobFromImage(frame, 1/255, (inpWidth, inpHeight), [0,0,0], 1, crop=False)
            net.setInput(blob)
            outs = net.forward(getOutputsNames(net))
            postprocess(frame, outs,1)
            t, _ = net.getPerfProfile()
            #label=''
            #cv.putText(frame, label, (0, 15), cv.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255))
            cv.imshow("Predicted Result", frame)
            if cv.waitKey(5) & 0xFF == ord('q'):
                break
        else:
            break
    video.release()
```

```
    cv.destroyAllWindows()



def exit():
    global main
    main.destroy()

font = ('times', 16, 'bold')
title = Label(main, text='Number Plate Detection without Helmet', justify=LEFT)
title.config(bg='lavender blush', fg='DarkOrchid1')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=100,y=5)
title.pack()

font1 = ('times', 14, 'bold')
model = Button(main, text="Upload Image", command=upload)
model.place(x=200,y=100)
model.config(font=font1)

uploadimage = Button(main, text="Detect Motor Bike & Person", command=detectBike)
uploadimage.place(x=200,y=150)
uploadimage.config(font=font1)

classifyimage = Button(main, text="Detect Helmet", command=detectHelmet)
classifyimage.place(x=200,y=200)
classifyimage.config(font=font1)

exitapp = Button(main, text="Exit", command=exit)
```

```
exitapp.place(x=200,y=250)
exitapp.config(font=font1)

font1 = ('times', 12, 'bold')
textarea=Text(main,height=15,width=60)
scroll=Scrollbar(textarea)
textarea.configure(yscrollcommand=scroll.set)
textarea.place(x=10,y=300)
textarea.config(font=font1)

loadLibraries()

main.config(bg='light coral')
main.mainloop()

import numpy as np
import cv2 as cv
import subprocess
import time
import os
from yoloDetection import detectObject, displayImage
import sys



global class_labels
global cnn_model
global cnn_layer_names

def loadLibraries(): #function to load yolov3 model weight and class labels
```

```
    global class_labels
    global cnn_model
    global cnn_layer_names
    class_labels = open('yolov3model/yolov3-labels').read().strip().split('\n') #reading labels from yolov3 model
    print(str(class_labels)+" == "+str(len(class_labels)))
    cnn_model = cv.dnn.readNetFromDarknet('yolov3model/yolov3.cfg', 'yolov3model/yolov3.weights') #reading model
    cnn_layer_names = cnn_model.getLayerNames() #getting layers from cnn model
    cnn_layer_names = [cnn_layer_names[i[0] - 1] for i in cnn_model.getUnconnectedOutLayers()] #assigning all layers

def detectFromImage(imagename): #function to detect object from images
    #random colors to assign unique color to each label
    label_colors = (0,255,0)#np.random.randint(0,255,size=(len(class_labels),3),dtype='uint8')
    try:
        image = cv.imread(imagename) #image reading
        image_height, image_width = image.shape[:2] #converting image to two dimensional array
    except:
        raise 'Invalid image path'
    finally:
        image, _, _, _, _ = detectObject(cnn_model, cnn_layer_names, image_height, image_width, image, label_colors, class_labels,indexno)#calling detection function
        displayImage(image,0)#display image with detected objects label

def detectFromVideo(videoFile): #function to read objects from video

    #random colors to assign unique color to each label
    label_colors = (0,255,0)#np.random.randint(0,255,size=(len(class_labels),3),dtype='uint8')
    indexno = 0
```

```
    try:


        video = cv.VideoCapture(videoFile)
        frame_height, frame_width = None, None  #reading video from given path
        video_writer = None
    except:
        raise 'Unable to load video'
    finally:
        while True:
            frame_grabbed, frames = video.read() #taking each frame from video
            #print(frame_grabbed)
            if not frame_grabbed: #condition to check whether video loaded or not
                break
            if frame_width is None or frame_height is None:
                frame_height, frame_width = frames.shape[:2] #detecting object from frame
            frames, _, _, _, _ = detectObject(cnn_model, cnn_layer_names, frame_height,
frame_width, frames, label_colors, class_labels,indexno)
            #displayImage(frames,index)
            #indexno = indexno + 1
            print(indexno)
            if indexno == 5:
               video.release()
               break


    print ("Releasing resources")
    #video_writer.release()
    video.release()
```

```
if _name_ == '_main_':
    loadLibraries()
    print("sample commands to run code with image or video")
    print("python yolo.py image input_image_path")
    print("python yolo.py video input_video_path")
    if len(sys.argv) == 3:
        if sys.argv[1] == 'image':
            detectFromImage(sys.argv[2])
        elif sys.argv[1] == 'video':
            detectFromVideo(sys.argv[2])
        else:
            print("invalid input")
    else:
        print("follow sample command to run code")



        #video_path = None
        #video_output_path = "out.avi"
import numpy as np
import argparse
import cv2 as cv
import subprocess
import time
import os


def detectObject(CNNnet, total_layer_names, image_height, image_width, image, name_colors,
class_labels,indexno,
        Boundingboxes=None, confidence_value=None, class_ids=None, ids=None, detect=True):
```

```
    if detect:
        blob_object = cv.dnn.blobFromImage(image,1/255.0,(416, 416),swapRB=True,crop=False)
        CNNnet.setInput(blob_object)
        cnn_outs_layer = CNNnet.forward(total_layer_names)
        Boundingboxes, confidence_value, class_ids = listBoundingBoxes(cnn_outs_layer, image_height, image_width, 0.5)
        ids = cv.dnn.NMSBoxes(Boundingboxes, confidence_value, 0.5, 0.3)
        if Boundingboxes is None or confidence_value is None or ids is None or class_ids is None:
          raise '[ERROR] unable to draw boxes.'
        image,option = labelsBoundingBoxes(image, Boundingboxes, confidence_value, class_ids, ids, name_colors, class_labels,indexno)

    return image,option


def labelsBoundingBoxes(image, Boundingbox, conf_thr, classID, ids, color_names, predicted_labels,indexno):
    option = 0
    if len(ids) > 0:
        for i in ids.flatten():
            # draw boxes
            xx, yy = Boundingbox[i][0], Boundingbox[i][1]
            width, height = Boundingbox[i][2], Boundingbox[i][3]

            class_color = (0,255,0)#[int(color) for color in color_names[classID[i]]]

            cv.rectangle(image, (xx, yy), (xx+width, yy+height), class_color, 2)
            print(classID[i])
            if classID[i] <= 1:
                text_label = "{}: {:4f}".format(predicted_labels[classID[i]], conf_thr[i])
```

```
            #displayImage(image,indexno)
            cv.putText(image, text_label, (xx, yy-5), cv.FONT_HERSHEY_SIMPLEX, 0.5, class_color, 2)
            option = 1

    return image,option


def listBoundingBoxes(image, image_height, image_width, threshold_conf):
    box_array = []
    confidence_array = []
    class_ids_array = []

    for img in image:
        for obj_detection in img:
            detection_scores = obj_detection[5:]
            class_id = np.argmax(detection_scores)
            confidence_value = detection_scores[class_id]
            if confidence_value > threshold_conf and class_id <= 1:
                Boundbox = obj_detection[0:4] * np.array([image_width, image_height, image_width,
image_height])
                center_X, center_Y, box_width, box_height = Boundbox.astype('int')


                xx = int(center_X - (box_width / 2))
                yy = int(center_Y - (box_height / 2))


                box_array.append([xx, yy, int(box_width), int(box_height)])
                confidence_array.append(float(confidence_value))
                class_ids_array.append(class_id)
```

```
    return box_array, confidence_array, class_ids_array


def displayImage(image,index):
    #cv.imwrite('bikes/'+str(index)+'.jpg',image)
    #index = index + 1
    cv.imshow("Final Image", image)
    cv.waitKey(0)
```