```python
import cv2
import time
from my_functions import *

source = 'test_video.mp4'
#cap = cv2.VideoCapture(0)
frame_size = (640, 480)  # Change to the desired frame size
save_video = False  # Do not save video  processed video.
show_video = True  # set true when using a video file
save_img = False  # set true when using only an image file to save the image . won't save individual frames as images.

cap = cv2.VideoCapture(source)

while cap.isOpened():   #creates a VideoCapture object named
    ret, frame = cap.read()  #This reads a frame from the video. The ret variable will be True if a frame is read successfully,

    if ret:  #Checks if a frame is successfully read.
        frame = cv2.resize(frame, frame_size)  # resizing image
        original_frame = frame.copy()  #Creates a copy  current frame forprocessing, altering
        frame, results = object_detection(frame) #Passes the frame to an object_detection()
                                #which  performs object detection on the frame and returns a
                                #modified frame (frame) with drawn rectangles around detected objects and
                                # a list of detection results (results).

        rider_list = []  #Initializes empty lists to store detected objects categorized as riders, heads, and numbers.
        head_list = []
        number_list = []


#iterating the results return by object_detection()
#x1, y1: Coordinates of the top-left corner of the bounding box around the object. , bottom right
#cnf: Confidence score
#clas: Class of the detected object (0, 1, or 2 in this case).

        for result in results:
            x1, y1, x2, y2, cnf, clas = result
            if clas == 0:
                rider_list.append(result)  #detected riders.
            elif clas == 1:
                head_list.append(result) #head , no helmet
```

```
            elif clas == 2:
                number_list.append(result) #number plate

        for rdr in rider_list:     #Iterates over each detected rider in the rider_list
            time_stamp = str(time.time())   #Generates a timestamp for the current iteration.
            x1r, y1r, x2r, y2r, cnfr, clasr = rdr  #Unpacks the coordinates and other information of the current rider.
            for hd in head_list:    #Iterates over each detected head in the head_list.
                x1h, y1h, x2h, y2h, cnfh, clash = hd  #Unpacks the coordinates and other information of the current head.
                if inside_box([x1r, y1r, x2r, y2r], [x1h, y1h, x2h, y2h]):  # if this head inside this rider bbox
                    try:
                        head_img = original_frame[y1h:y2h, x1h:x2h]
                        helmet_present = img_classify(head_img) #classify whether the head is wearing a helmet using
                    except:
                        helmet_present[0] = None  #classify failed


#If a helmet is detected, a green rectangle is drawn around the head, and text indicating the confidence is added.

                    if helmet_present[0] == True:  # if helmet present
                        frame = cv2.rectangle(frame, (x1h, y1h), (x2h, y2h), (0, 255, 0), 1)
                        frame = cv2.putText(frame, f'{round(helmet_present[1], 1)}', (x1h, y1h + 40),
                                            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1, cv2.LINE_AA)

#If the prediction is uncertain (None), a cyan rectangle is drawn around the head, and text indicating the confidence is added.
                    elif helmet_present[0] == None:  # Poor prediction
                        frame = cv2.rectangle(frame, (x1h, y1h), (x2h, y2h), (0, 255, 255), 1)
                        frame = cv2.putText(frame, f'{round(helmet_present[1], 1)}', (x1h, y1h),
                                            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1, cv2.LINE_AA)

#If no helmet is detected, a red rectangle is drawn around the head, and text indicating the confidence is added.

                    elif helmet_present[0] == False:  # if helmet absent
                        frame = cv2.rectangle(frame, (x1h, y1h), (x2h, y2h), (0, 0, 255), 1)
                        frame = cv2.putText(frame, f'{round(helmet_present[1], 1)}', (x1h, y1h + 40),
                                            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1, cv2.LINE_AA)

#
#it attempts to extract a portion of the frame containing the number plate of the rider (number_plate_img) and
#saves it to the "number_plates" folder with a filename based on the timestamp.
```

```
                        try:
                            number_plate_img = original_frame[y2r:y2r + int(0.3 * (y2r - y1r)), x1r:x2r]
                            cv2.imwrite(f'number_plates/{time_stamp}.jpg', number_plate_img)
                        except:
                            print('Could not save number plate')

        if save_img:  # Checks if the save_img flag is set to True. ; otherwise, it skips the block.
            cv2.imwrite('saved_frame.jpg', frame)  #Saves the current frame as an image
        if show_video:  # Checks if the show_video flag is set to True
            frame = cv2.resize(frame, (900, 450))  # resizing to fit on the screen
            cv2.imshow('Frame', frame)#Displays the current frame in a
                                #window with the title "Frame". This is useful for real-time visualization of the processed video.

        if cv2.waitKey(1) & 0xFF == ord('q'):  #Waits for a key event. If the key pressed is 'q', it breaks out of the loop
            break
    else:
        break

cap.release()  #Releases the video capture object.
cv2.destroyAllWindows() #Destroys all OpenCV windows , closes windows
print('Execution completed')
```