Computer Vision Project 2

Olteanu Fabian Cristian

FMI, AI Master, Year 1

1. Introduction

At the core of both approaches which were taken to tackle tasks one and two lies the a pre-trained version (using the MS COCO dataset [1]) of the deep neural network YOLOv5s[2] by Ultralytics. The "s" variant of the model was chosen for its lightweightness, ease of use with the OpenCV Python library and good object detection accuracy. It also features the ability to detect and differentiate between 80 classes of objects, although it was restricted to only five of them (bicycles, cars, motorbikes, busses and trucks).

YOLOv5s takes as input images in the RGB format with pixel values between zero and one and of size 640x640. Based on this input, the model generates prediction as a data structure comprised of a 25200x85 list of lists, each row representing the x,y,w,h coordinates of the detected bounding box rectangle, the confidence level of the detection and the scores of each of the 80 classes.

2. Task 1

The steps taken to achieve a 97% accuracy on the training dataset are summerized as follows:

1. The coordinates of the four corners of each polygon representing the lanes were hand-picked and stored in nine lists. The Python library "shapely"[3] was also used in order to compute the areas of intersection between the lanes and the detected bounding boxes, which will be described in the next steps.

2. The YOLOv5s pre-trained model is loaded into OpenCV (using the compatible .onnx converted format) and is used in order to obtain the bounding box of the five specified types of vehicles (example in figure 1). This is done by unwrapping the predictions generated by the model of an image using some constants taking the roles of tresholds. The values of these thresholds are as follows:

   - Minimum confidence threshold: 0.4,
   - Non-maximum suppression threshold (to avoid duplicate, overlapping bounding boxes): 0.45,
   - Score threshold: 0.25.

   The values were obtained through trial and error (lowering or increasing the confidence threshold was found to cause the most differences in accuracy).

3. In order to determine which lanes are occupied in an image, an algorithm involving the maximum intersection of the bounding box and lane polygons had to be devised. For an individual image, the algorithm goes through each of the bounding boxes detected at the prior step. For each lane, it tries to find the maximum overlap between the bounding box and the lane, that is the maximum area of the intersection between these two polygons. For

lanes four through 6, because of the perspective, it is better to compute this overlap based on the lower half of the vehicle bounding box and the lane. Finally, after the maximum area has been computed, a threshold of 500 is imposed so that situations such as those when a vehicle's bounding box that is outside of the lanes slightly intersects them are avoided. The lane corresponding to the maximum overlap is marked as occupied.

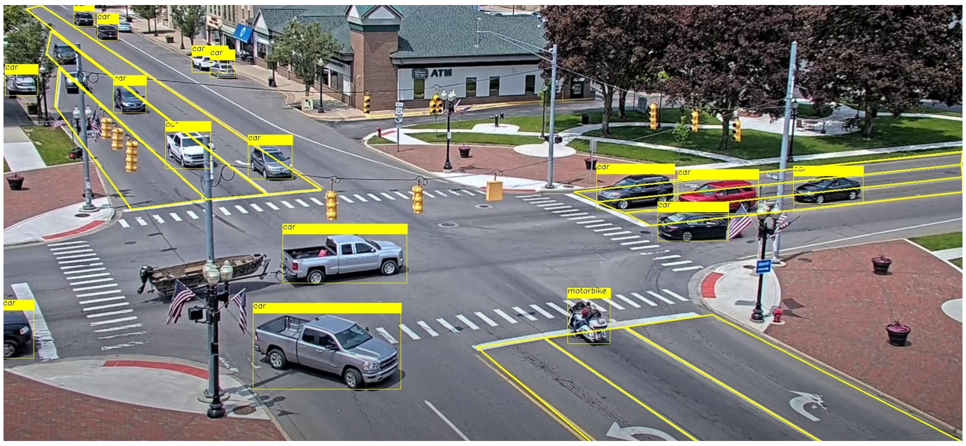

Figure 1. Detection of vehicles and highlighting of lanes in 01_1.jpg

3. Task 2

The solution implemented for Task 2 also fully takes advantage of YOLOv5s for detection. The tracking algorithm works as follows:

1. Likewise to task 1, the predictions for an image are unwrapped in order to obtain the bounding boxes of the detected vehicles in the image (in this case the image is a frame of the 60fps video).
2. The initial tracked bounding box is extracted from the query text file and assigned to a tracked bounding box list.
3. For each frame in the video, the current tracked object bounding box (located in the tracked object bounding box list at the previous position) has some padding added (50 pixels - the value was found through trial and error) and is then extracted from the initial image as a separate image. The model is used to detect the position of the new bounding box. This is done achieved through detections of the bounding box in the restricted area, calculating the maximum intersection over union for each detection with the tracked bounding box at the last step. If there are no detections at the current frame, the tracked bounding box from the last step is appended to the array one more time.

References

[1] https://cocodataset.org/#home

[2] https://github.com/ultralytics/yolov5/wiki

[3] https://pypi.org/project/shapely/