SE4050
Deep Learning
4th Year, 1st Semester

Lab 04

Submitted to
Sri Lanka Institute of Information Technology

IT21166488

In partial fulfillment of the requirements for the
Bachelor of Science Special Honors Degree in Information Technology

30/08/2024

1. In the below given cell, shape of the boxes.eval() is (1783,4). Why are there 1783 boxes? Explain the reason for it. What is the maximum number and minimum number you can get for that? Write these answers in a word file.

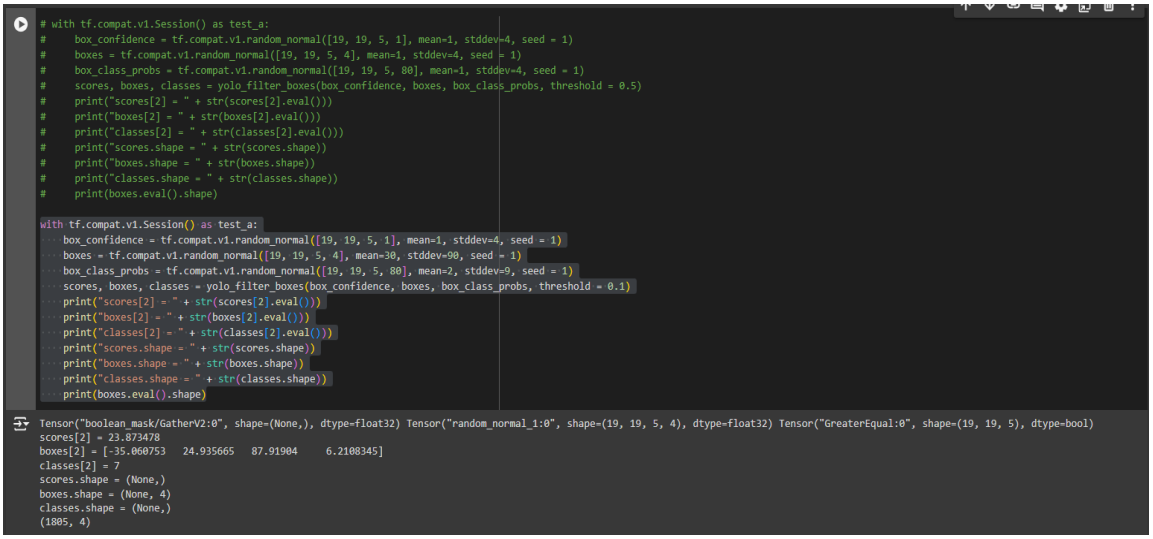
   The 1783 boxes are the result of filtering out the original set of 1805 predicted boxes. The YOLO model divides the image into a grid of 19x19 cells, with each cell predicting 5 boxes. This gives a total of 1805 boxes. After applying a confidence threshold, which removes boxes with low confidence, 1783 boxes remain.

   What is the maximum number of boxes?

   The maximum number of boxes you can get is 1805. This number comes from multiplying the 19x19 grid cells by 5 anchor boxes per cell.

   What is the minimum number of boxes?

   The minimum number of boxes you can get is 4. This could happen if only one box is detected with high confidence.



```
# with tf.compat.v1.Session() as test_a:
#     box_confidence = tf.compat.v1.random_normal([19, 19, 5, 1], mean=1, stddev=4, seed = 1)
#     boxes = tf.compat.v1.random_normal([19, 19, 5, 4], mean=1, stddev=4, seed = 1)
#     box_class_probs = tf.compat.v1.random_normal([19, 19, 5, 80], mean=1, stddev=4, seed = 1)
#     scores, boxes, classes = yolo_filter_boxes(box_confidence, boxes, box_class_probs, threshold = 0.5)
#     print("scores[2] = " + str(scores[2].eval()))
#     print("boxes[2] = " + str(boxes[2].eval()))
#     print("classes[2] = " + str(classes[2].eval()))
#     print("scores.shape = " + str(scores.shape))
#     print("boxes.shape = " + str(boxes.shape))
#     print("classes.shape = " + str(classes.shape))
#     print(boxes.eval().shape)

with tf.compat.v1.Session() as test_a:
    box_confidence = tf.compat.v1.random_normal([19, 19, 5, 1], mean=1, stddev=4, seed = 1)
    boxes = tf.compat.v1.random_normal([19, 19, 5, 4], mean=30, stddev=90, seed = 1)
    box_class_probs = tf.compat.v1.random_normal([19, 19, 5, 80], mean=2, stddev=9, seed = 1)
    scores, boxes, classes = yolo_filter_boxes(box_confidence, boxes, box_class_probs, threshold = 0.1)
    print("scores[2] = " + str(scores[2].eval()))
    print("boxes[2] = " + str(boxes[2].eval()))
    print("classes[2] = " + str(classes[2].eval()))
    print("scores.shape = " + str(scores.shape))
    print("boxes.shape = " + str(boxes.shape))
    print("classes.shape = " + str(classes.shape))
    print(boxes.eval().shape)
```

```
Tensor("boolean_mask/GatherV2:0", shape=(None,), dtype=float32) Tensor("random_normal_1:0", shape=(19, 19, 5, 4), dtype=float32) Tensor("GreaterEqual:0", shape=(19, 19, 5), dtype=bool)
scores[2] = 23.873478
boxes[2] = [-35.060753   24.935665   87.91904     6.2108345]
classes[2] = 7
scores.shape = (None,)
boxes.shape = (None, 4)
classes.shape = (None,)
(1805, 4)
```

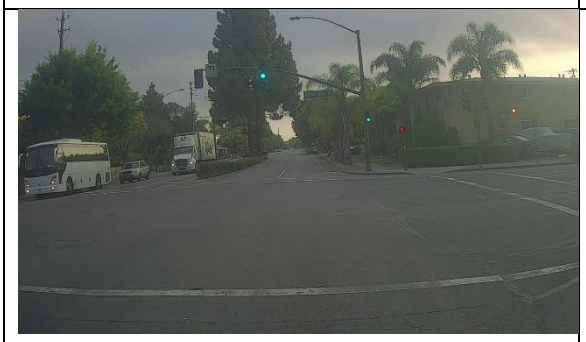
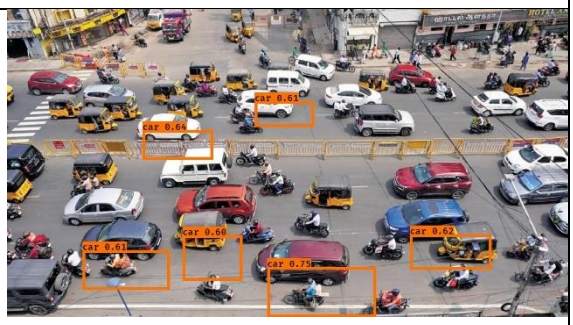
2. yolo_anchors.txt contains 10 values. They can be considered as height and width of 5 anchor boxes. What is the advantage of using such anchor boxes?

   Anchor boxes allow the YOLO model to detect objects of various shapes and sizes more effectively. By using predefined anchor boxes with different heights and widths, the model can better predict bounding boxes that match the actual objects in an image. This improves the accuracy of object detection, especially when objects have different aspect ratios.

3. What was the method used to determine the sizes of these anchor boxes?

   The sizes of the anchor boxes were determined using a method called **k-means clustering**. This technique groups the bounding boxes from a dataset into clusters based on their width and height. The center points of these clusters are chosen as the sizes for the anchor boxes, making them better suited to the objects in the images.

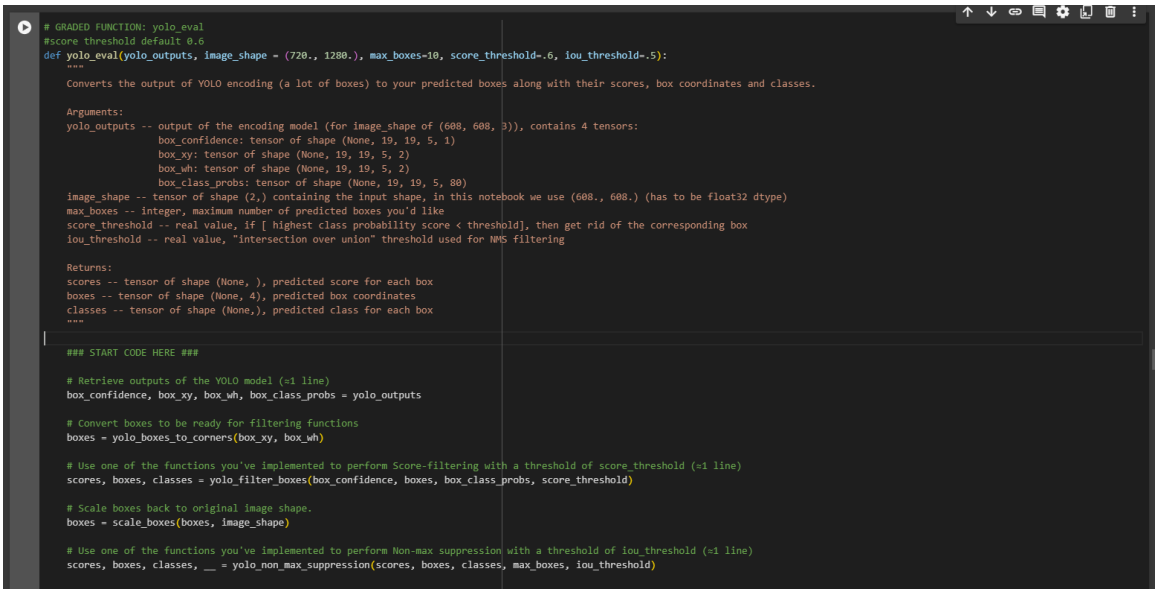
4. Write what you observe regarding correctly detected objects, incorrectly detected objects, undetected objects and incorrect bounding boxes.

Original Image	Output Image
	
	
	
	

Aspect	Image 1 (Rainy Road)	Image 2 (Urban Street)	Image 3 (Busy Intersection)	Image 3 (Highway Traffic)
Correctly detected objects	3 cars	1 bus	6 cars	10 cars
Incorrectly detected objects	Nonvisible	Nonvisible	Nonvisible	Nonvisible
Undetected objects	Road signs, guardrails	Cars, traffic lights, pedestrians	Many cars, motorcycles, auto-rickshaws, people	Many cars
Incorrect bounding boxes	Slightly large, but accurate	Accurate for the bus	Mostly accurate for detected cars	Generally accurate
Overall performance	Good in simple scene	Poor in urban scene	Poor in complex scene	Moderate in dense traffic

5. Adjusting parameters like max_boxes, score_threshold, and iou_threshold of the yolo_eval function can potentially address the limitations.

   Change the max_boxes [integer value]



```
# GRADED FUNCTION: yolo_eval
#score threshold default 0.6
def yolo_eval(yolo_outputs, image_shape = (720., 1280.), max_boxes=10, score_threshold=.6, iou_threshold=.5):
    """
    Converts the output of YOLO encoding (a lot of boxes) to your predicted boxes along with their scores, box coordinates and classes.

    Arguments:
    yolo_outputs -- output of the encoding model (for image_shape of (608, 608, 3)), contains 4 tensors:
                    box_confidence: tensor of shape (None, 19, 19, 5, 1)
                    box_xy: tensor of shape (None, 19, 19, 5, 2)
                    box_wh: tensor of shape (None, 19, 19, 5, 2)
                    box_class_probs: tensor of shape (None, 19, 19, 5, 80)
    image_shape -- tensor of shape (2,) containing the input shape, in this notebook we use (608., 608.) (has to be float32 dtype)
    max_boxes -- integer, maximum number of predicted boxes you'd like
    score_threshold -- real value, if [ highest class probability score < threshold], then get rid of the corresponding box
    iou_threshold -- real value, "intersection over union" threshold used for NMS filtering

    Returns:
    scores -- tensor of shape (None, ), predicted score for each box
    boxes -- tensor of shape (None, 4), predicted box coordinates
    classes -- tensor of shape (None,), predicted class for each box
    """

    ### START CODE HERE ###

    # Retrieve outputs of the YOLO model (≈1 line)
    box_confidence, box_xy, box_wh, box_class_probs = yolo_outputs

    # Convert boxes to be ready for filtering functions
    boxes = yolo_boxes_to_corners(box_xy, box_wh)

    # Use one of the functions you've implemented to perform Score-filtering with a threshold of score_threshold (≈1 line)
    scores, boxes, classes = yolo_filter_boxes(box_confidence, boxes, box_class_probs, score_threshold)

    # Scale boxes back to original image shape.
    boxes = scale_boxes(boxes, image_shape)

    # Use one of the functions you've implemented to perform Non-max suppression with a threshold of iou_threshold (≈1 line)
    scores, boxes, classes, _ = yolo_non_max_suppression(scores, boxes, classes, max_boxes, iou_threshold)
```

Figure 1: with max box 10

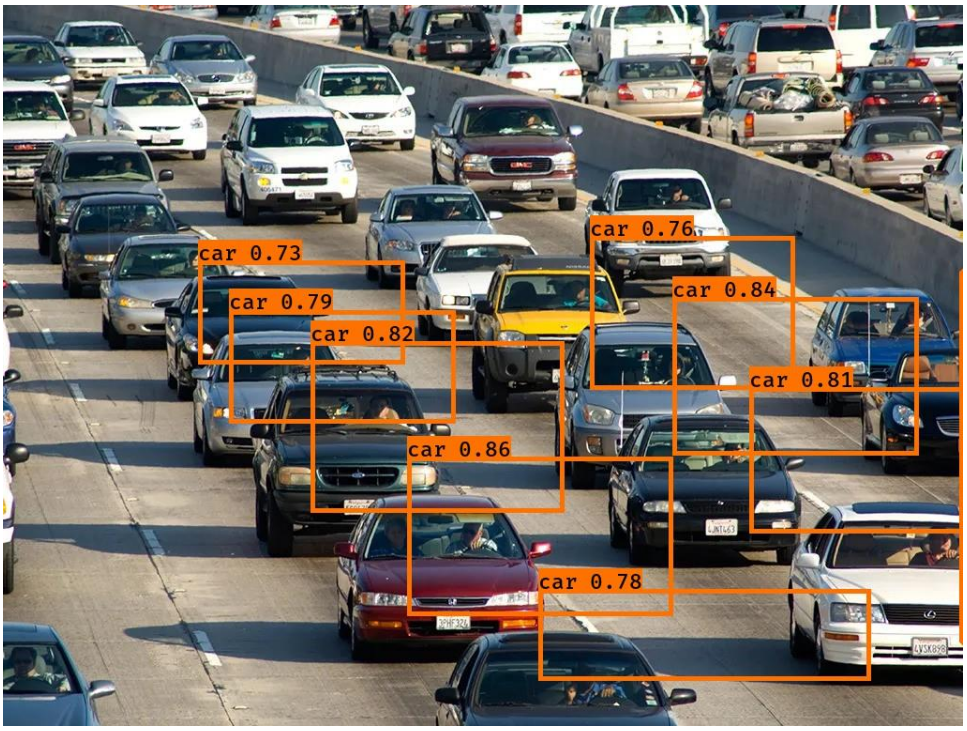


Figure 2: with max box 10

```python
# GRADED FUNCTION: yolo_eval
#score threshold default 0.6
def yolo_eval(yolo_outputs, image_shape = (720., 1280.), max_boxes=100, score_threshold=.6, iou_threshold=.5):
    """
    Converts the output of YOLO encoding (a lot of boxes) to your predicted boxes along with their scores, box coordinates and classes.

    Arguments:
    yolo_outputs -- output of the encoding model (for image_shape of (608, 608, 3)), contains 4 tensors:
                    box_confidence: tensor of shape (None, 19, 19, 5, 1)
                    box_xy: tensor of shape (None, 19, 19, 5, 2)
                    box_wh: tensor of shape (None, 19, 19, 5, 2)
                    box_class_probs: tensor of shape (None, 19, 19, 5, 80)
    image_shape -- tensor of shape (2,) containing the input shape, in this notebook we use (608., 608.) (has to be float32 dtype)
    max_boxes -- integer, maximum number of predicted boxes you'd like
    score_threshold -- real value, if [ highest class probability score < threshold], then get rid of the corresponding box
    iou_threshold -- real value, "intersection over union" threshold used for NMS filtering

    Returns:
    scores -- tensor of shape (None, ), predicted score for each box
    boxes -- tensor of shape (None, 4), predicted box coordinates
    classes -- tensor of shape (None,), predicted class for each box
    """

    ### START CODE HERE ###

    # Retrieve outputs of the YOLO model (≈1 line)
    box_confidence, box_xy, box_wh, box_class_probs = yolo_outputs

    # Convert boxes to be ready for filtering functions
    boxes = yolo_boxes_to_corners(box_xy, box_wh)

    # Use one of the functions you've implemented to perform Score-filtering with a threshold of score_threshold (≈1 line)
    scores, boxes, classes = yolo_filter_boxes(box_confidence, boxes, box_class_probs, score_threshold)

    # Scale boxes back to original image shape.
    boxes = scale_boxes(boxes, image_shape)

    # Use one of the functions you've implemented to perform Non-max suppression with a threshold of iou_threshold (≈1 line)
    scores, boxes, classes, _ = yolo_non_max_suppression(scores, boxes, classes, max_boxes, iou_threshold)
```

Figure 3: with max box 100

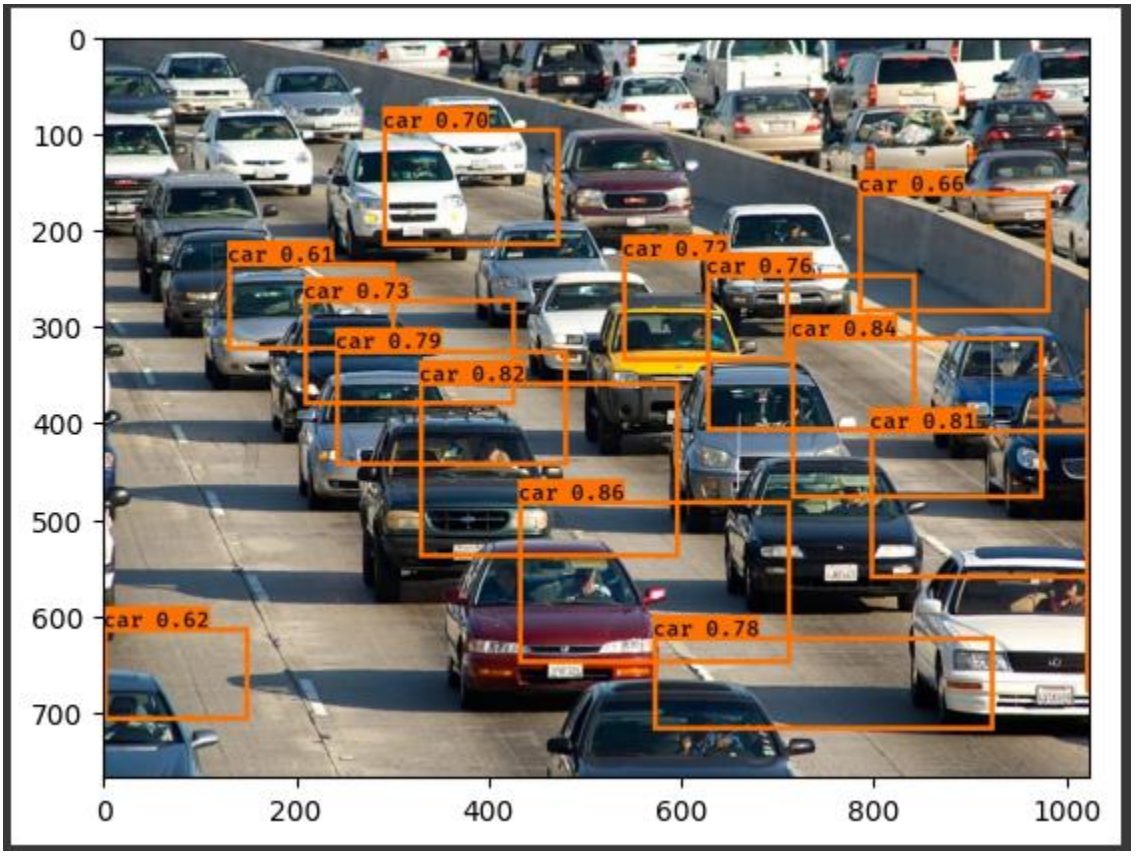


Figure 4: with max box 100

Overall: Not Much Effective

Also after changing score threshold, and iou_threshold values, it is same as before result.