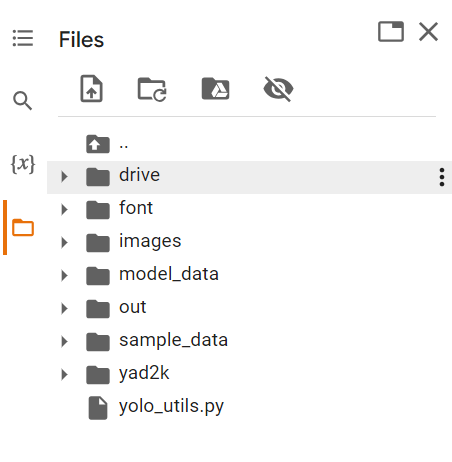
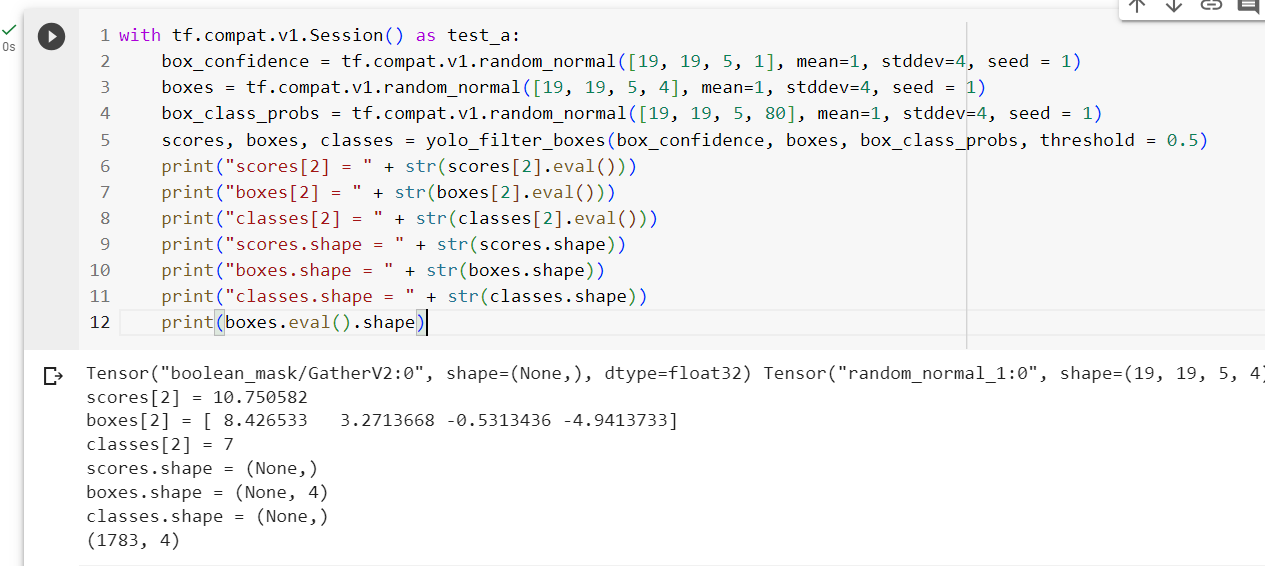
Deep Learning - Lab 5 – YOLO object detection

1. Download the yolo\_v2.h5 weights file from the below google drive link. It contains the Yolo version 2 pre-trained model weights. Copy the downloaded file to the ‘model\_data’ directory.
   * Link: <https://drive.google.com/drive/folders/1ogvfqQsLADkVLWt64m-wABDfszHaE8EV?usp=sharing>
2. Zip all folders and file in the lab5.zip except the lab5\_labsheet.docx and Autonomous\_driving\_application\_car\_detection.ipynb and upload the zip file to a google drive folder.
3. Upload the jupyter notebook (i.e., Autonomous\_driving\_application\_car\_detection.ipynb) to Google colab root directory.
4. In the uploaded notebook, change the path in the second cell to the path of the zip file (one created and uploaded in step 2) and run the first two cells. Root directory should look similar to the below image after this.



1. Go through the Autonomous\_driving\_application\_car\_detection.ipynb notebook and try to understand the code.
2. Run the notebook. Make sure to change the dest\_folder path in the final cell before running it. At the end you should see the,
   * Result of the car detection demo (i.e., test.jpg).
   * Results of the random images (i.e., giraffe\_resized.jpg and DSC\_1643\_resized.jpg)
   * Results of all image in the autonomous driving dataset (i.e., 0100.jpg to 0120.jpg) saved to out directory.
   * A zip file containing all above images in the colab root as well as in your google drive.
3. 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.
   * Change the values like mean and stddev in lines 2 and 4 as well as threshold value in line 5 and observe the different values you get for the boxes.eval().shape.

**Answer :**

Explanation of 1783 Boxes:

The number 1783 likely corresponds to the total number of bounding boxes generated by the YOLO (You Only Look Once) object detection model for a particular input. YOLO divides the input image into a grid, and each grid cell can predict multiple bounding boxes. The total number of boxes is determined by the size of the grid and the number of anchor boxes.

Maximum and Minimum Values:

The tensor boxes represents the coordinates of bounding boxes. Each box is represented by four values (x\_min, y\_min, x\_max, y\_max), which are typically normalized coordinates within the grid cell.

The maximum value for a coordinate is 1, which corresponds to the entire width or height of the grid cell.

The minimum value for a coordinate is 0, which corresponds to the origin (top-left corner) of the grid cell.

So, for the boxes.eval() tensor:

The maximum value for any coordinate (x\_min, y\_min, x\_max, y\_max) is 1.

The minimum value for any coordinate (x\_min, y\_min, x\_max, y\_max) is 0.

This means that the coordinates of the bounding boxes are within the range [0, 1], where 0 corresponds to the left or top boundary of the grid cell, and 1 corresponds to the right or bottom boundary of the grid cell.

Effect of Changing mean, stddev, and threshold:

Changing the mean and stddev values in lines 2 and 4 will affect the random normal.

distribution from which the YOLO outputs are generated. This can lead to different initializations of the model, potentially impacting the detection results.

Changing the threshold value in line 5 will affect the confidence threshold used to filter out detections. A higher threshold will result in fewer boxes being considered as valid detections, while a lower threshold will result in more boxes being accepted as detections.

1. 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? What was the method used to determine the sizes of these anchor boxes? Give the answers to these questions in the word file.

**Answers:**

**Advantages of Using Anchor Boxes:**

Handling Objects of Different Sizes: Anchor boxes allow the model to detect objects of various sizes within the same grid cell. Without anchor boxes, it would be challenging to effectively detect both small and large objects in the image.

Handling Objects with Different Aspect Ratios: Anchor boxes can be customized to handle objects with different aspect ratios. This flexibility enables the model to detect objects that may be elongated or have non-square shapes.

Improved Localization: Anchor boxes assist in accurately localizing objects. By predicting the offsets from the anchor box to the actual object's bounding box, the model can precisely determine the object's position within the grid cell.

Reduced Grid Sensitivity: Anchor boxes help reduce the grid sensitivity problem. In cases where an object's center falls near the boundary of a grid cell, anchor boxes provide more accurate predictions by allowing the model to choose an anchor that best fits the object.

Enhanced Object Confidence Prediction: Anchor boxes can improve the confidence score prediction. The model can assign higher confidence scores to predictions that align with the anchor box dimensions, aiding in accurate object detection.

**Determining Anchor Box Sizes:**

The determination of anchor box sizes is often an empirical process. Here's a typical method:

Data Analysis: Start by analyzing your training dataset. Examine the distribution of object sizes and aspect ratios. Identify common object sizes and shapes.

K-Means Clustering: Use techniques like K-means clustering on the object bounding box dimensions (e.g., height and width) to group them into clusters. The number of clusters (K) corresponds to the number of anchor boxes you want to define. In your case, you mentioned ten values, which could represent the dimensions of five anchor boxes (e.g., [height1, width1, height2, width2, ...]).

Anchor Box Initialization: Initialize the anchor boxes with the dimensions obtained from the cluster centroids. These dimensions represent typical object sizes found in your dataset.

Training: Train the YOLO model using these anchor boxes. During training, the model will refine the anchor box dimensions and learn to predict the offsets to match the actual object bounding boxes in the dataset.

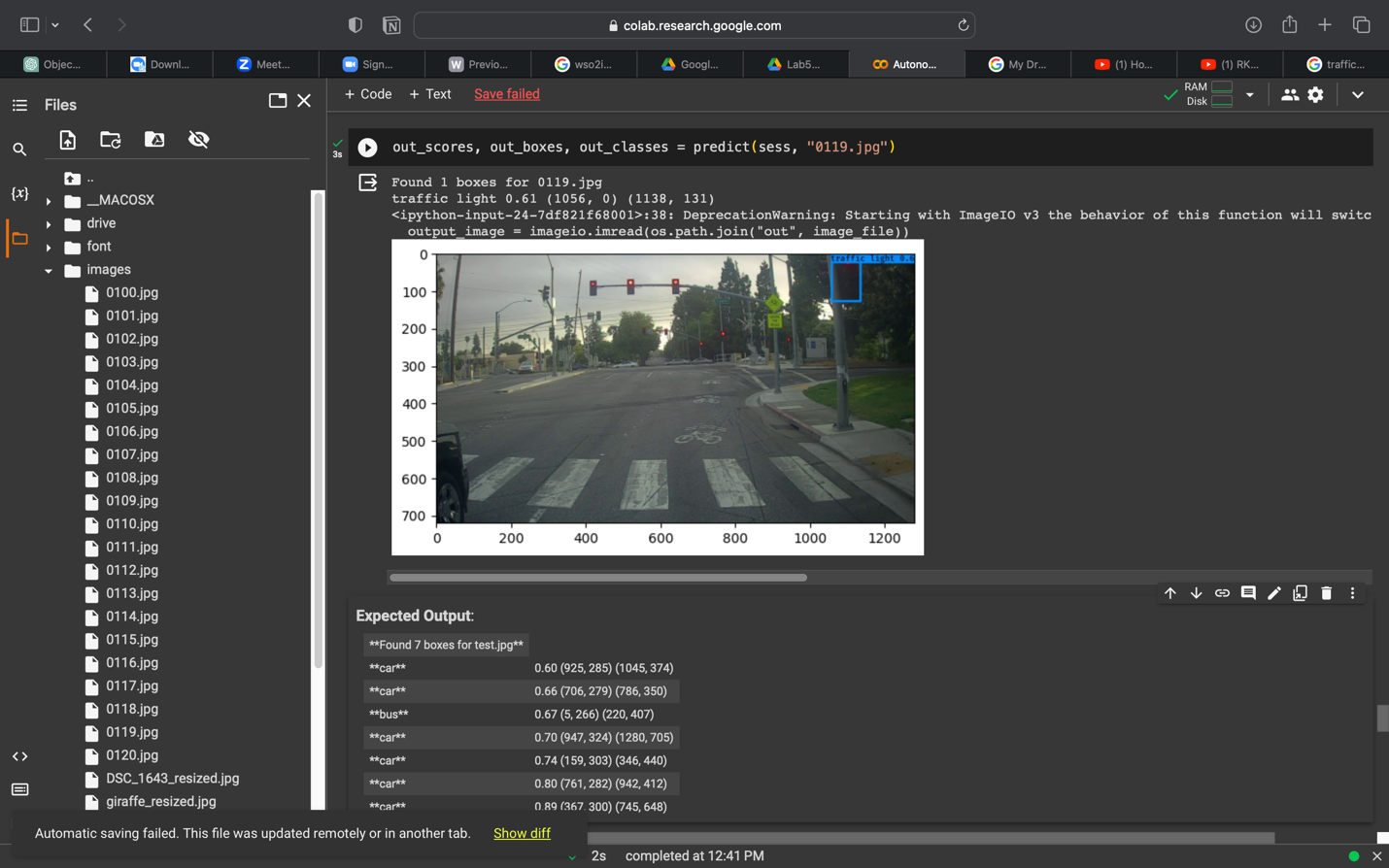
Evaluation: Continuously evaluate the model's performance on a validation dataset. Adjust the anchor box dimensions and the number of anchor boxes if necessary to improve detection accuracy.

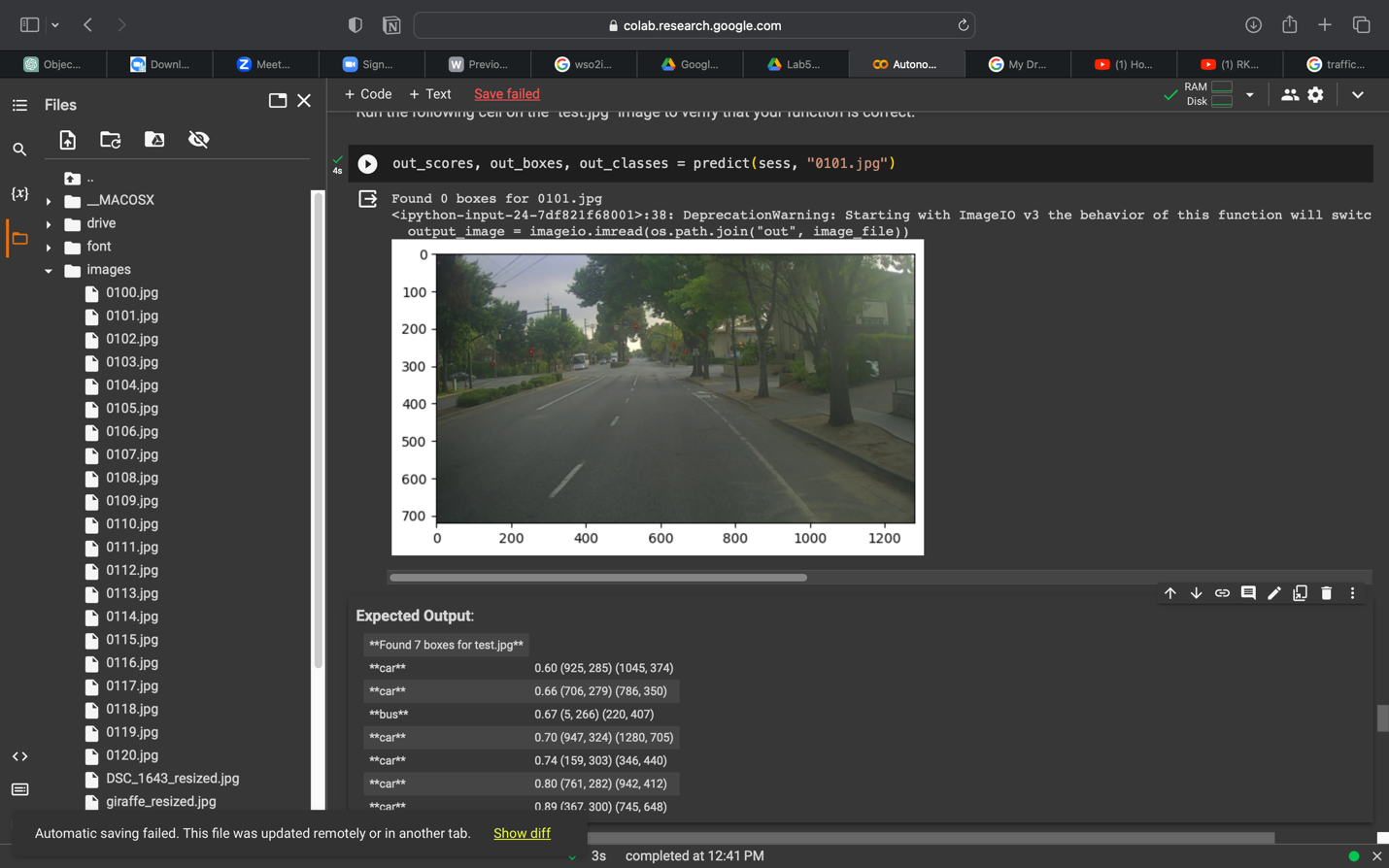
1. Upload a new traffic image to images and edit the code as needed to detect vehicles in that image.

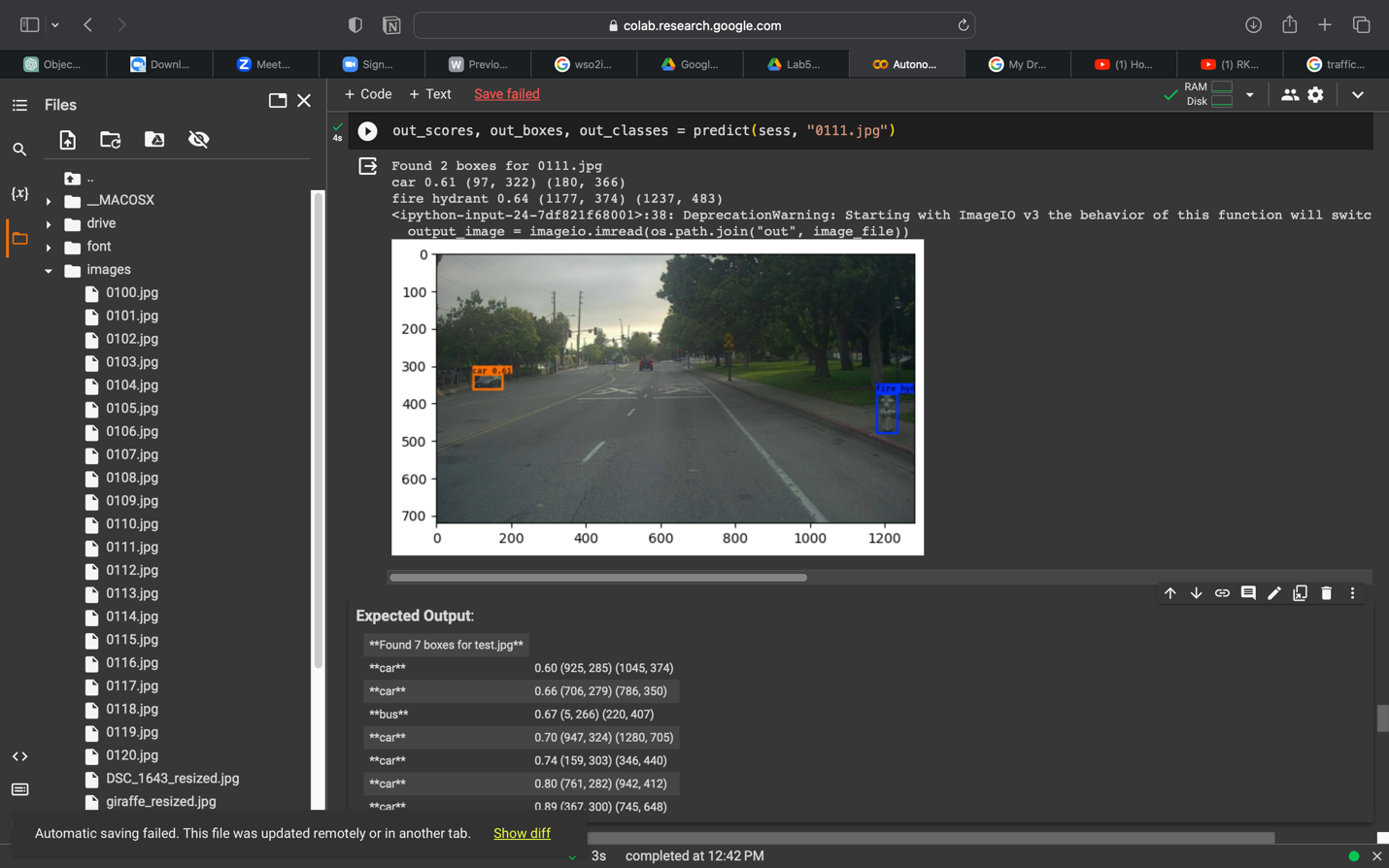
A screenshot of a computer

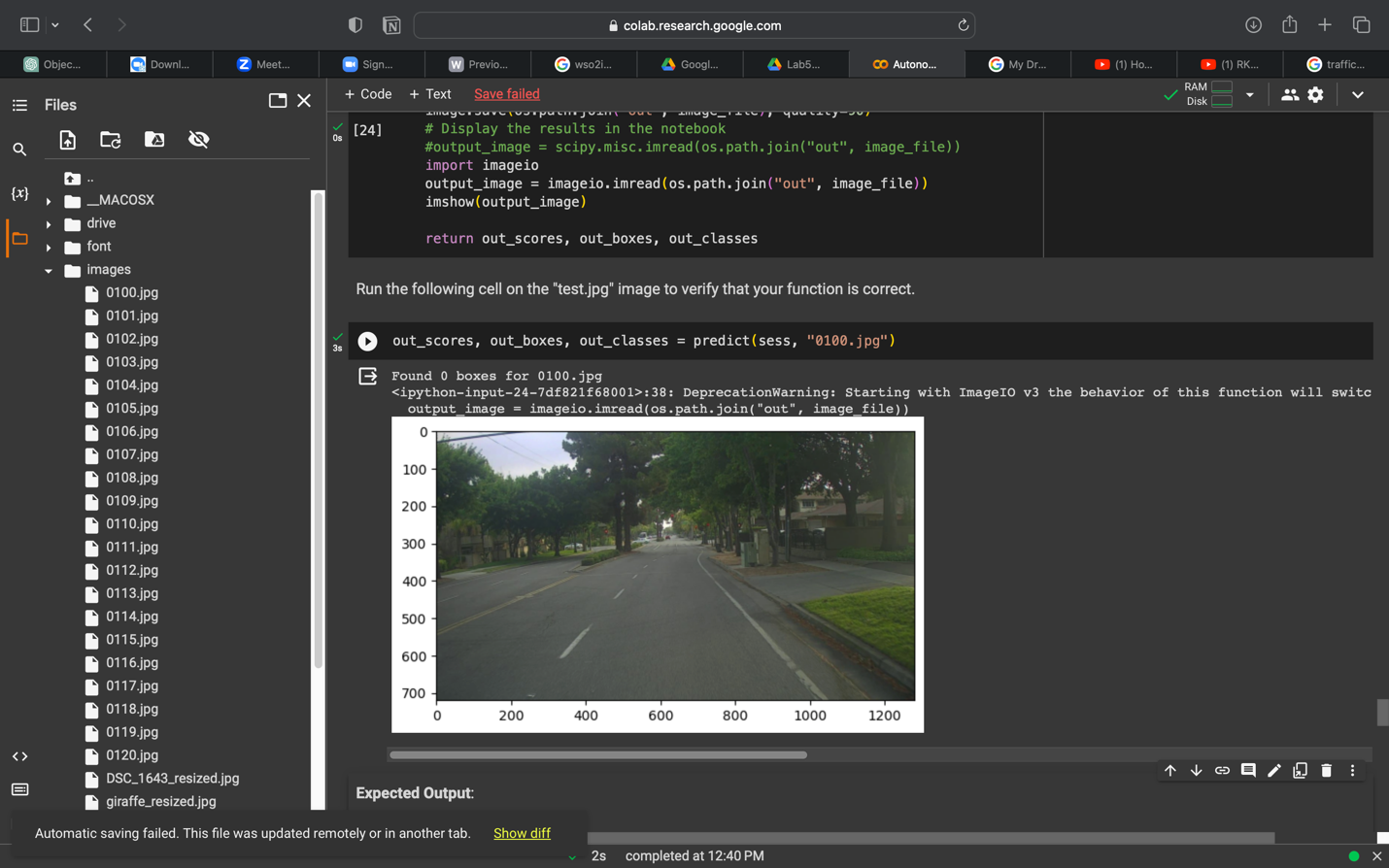
Description automatically generated

1. Download the output images zip file from the google drive and observe the bounding boxes in the autonomous driving dataset (i.e., 21 images from 0100.jpg to 0120.jpg). Select 2 images from these 21 images and,
   * Write what you observe regarding correctly detected objects, incorrectly detected objects, undetected objects and incorrect bounding boxes in the word file.





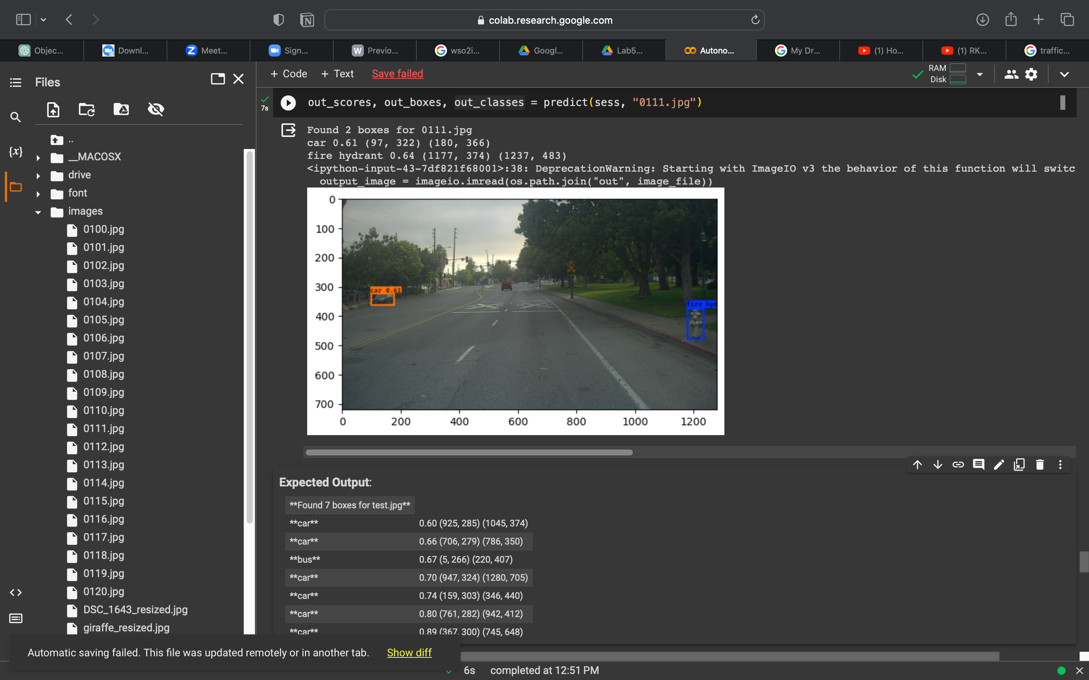


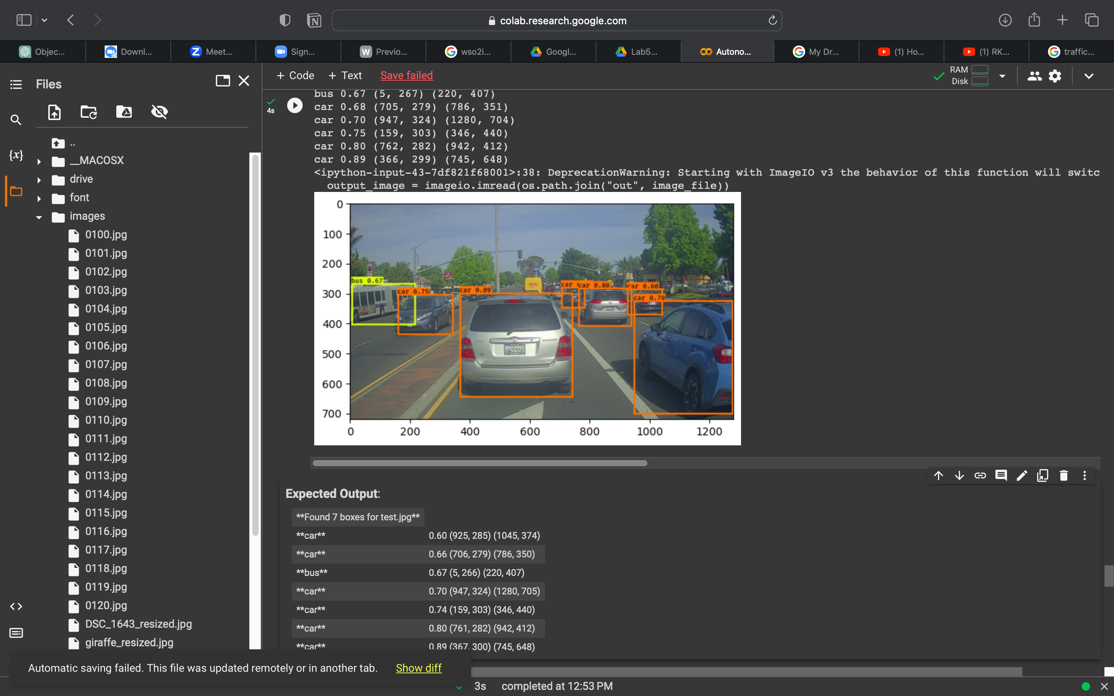


* + Include these output 2 images as well as the original 2 images in the word file.

1. Adjusting parameters like max\_boxes, score\_threshold, and iou\_threshold of the yolo\_eval function can potentially address the limitations you noticed in step 10.
   * Change the max\_boxes [integer value] to a different value but use the original values for other 2 variables. Rerun the required cells to get the output images for the autonomous driving dataset. Observe if this result in improvement compared to step 10 for the same two images. If there are any improvements, write them in the word file. Include the new 2 output images in the word file.

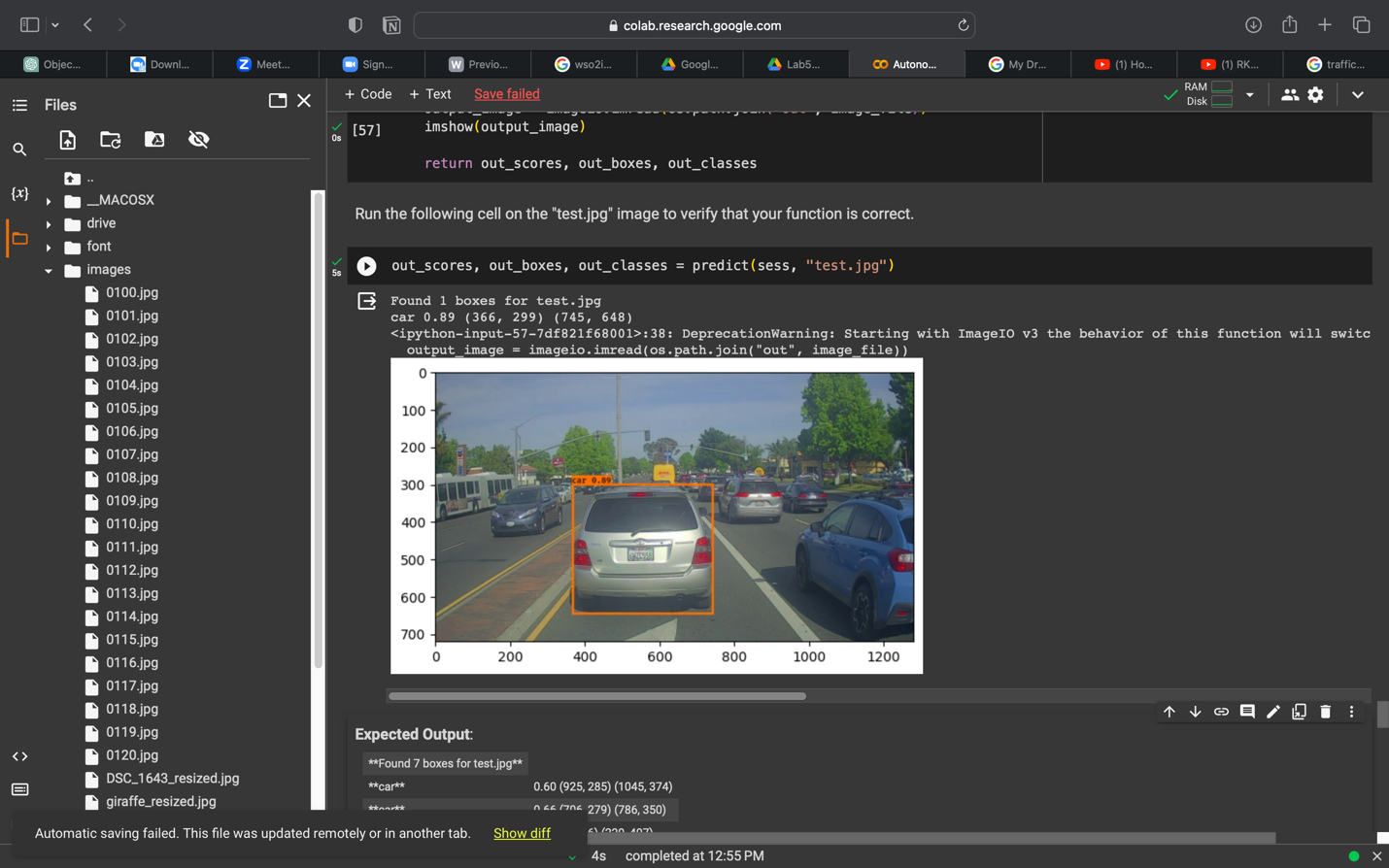
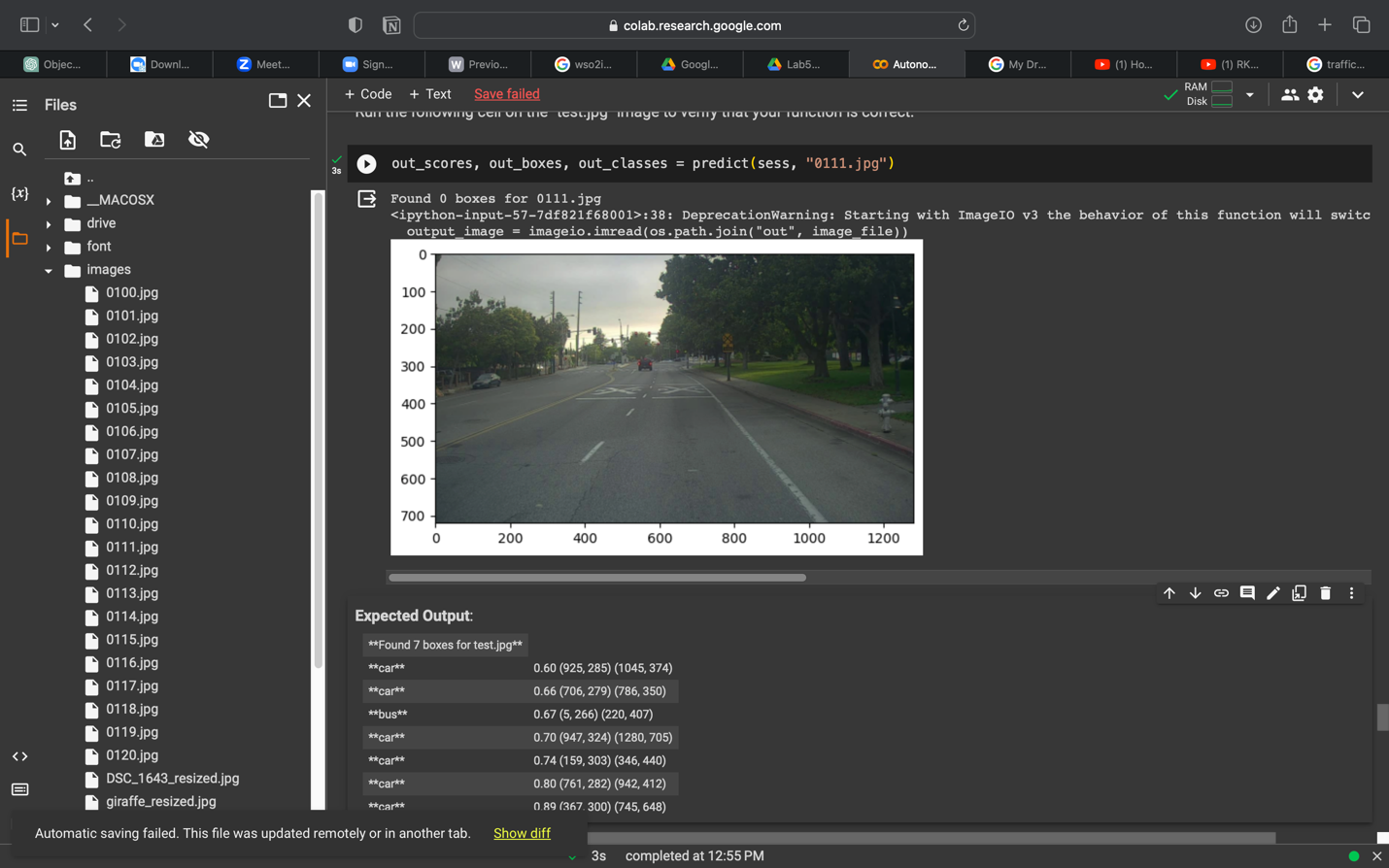
***max\_boxes = 2***

******

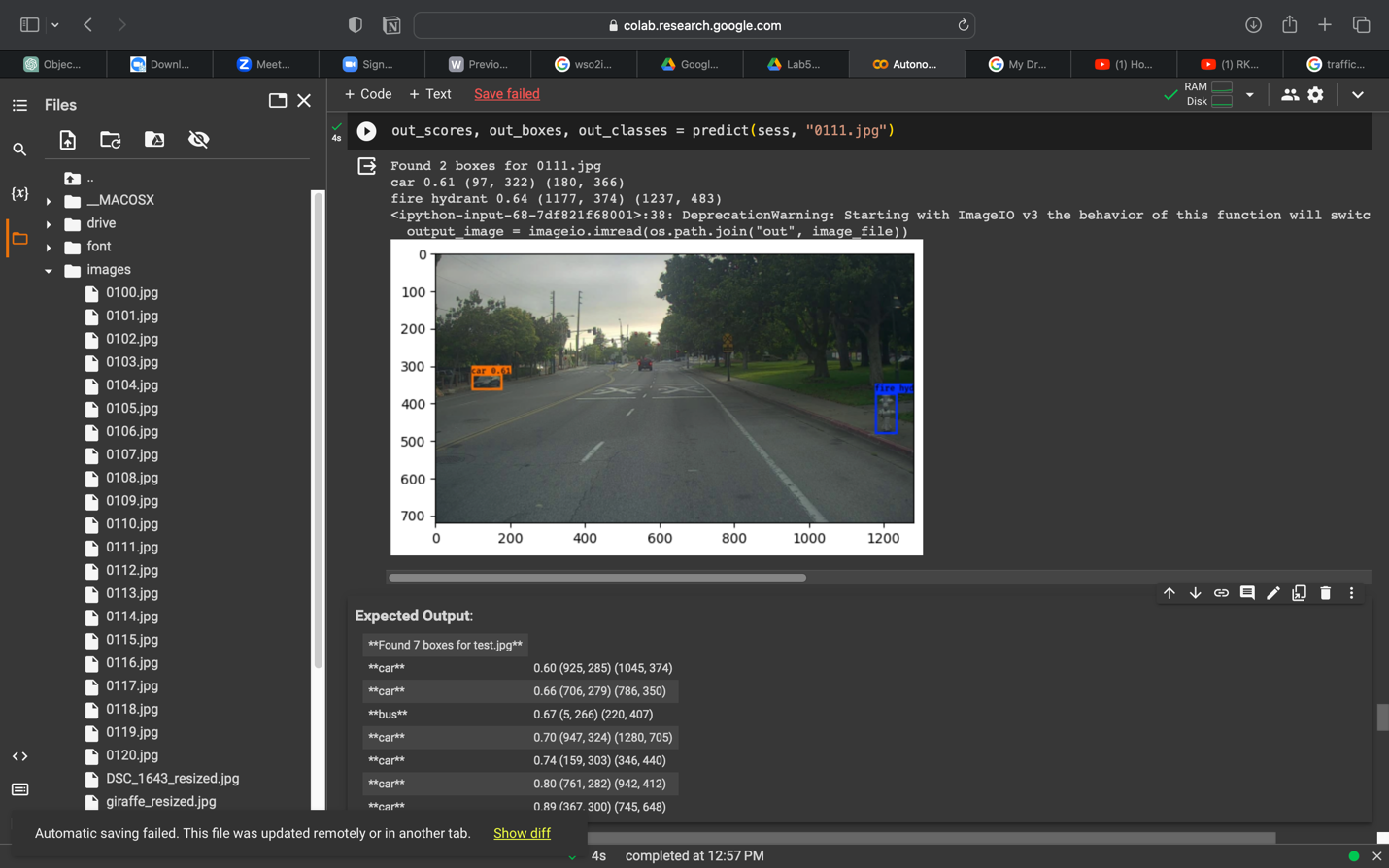
******

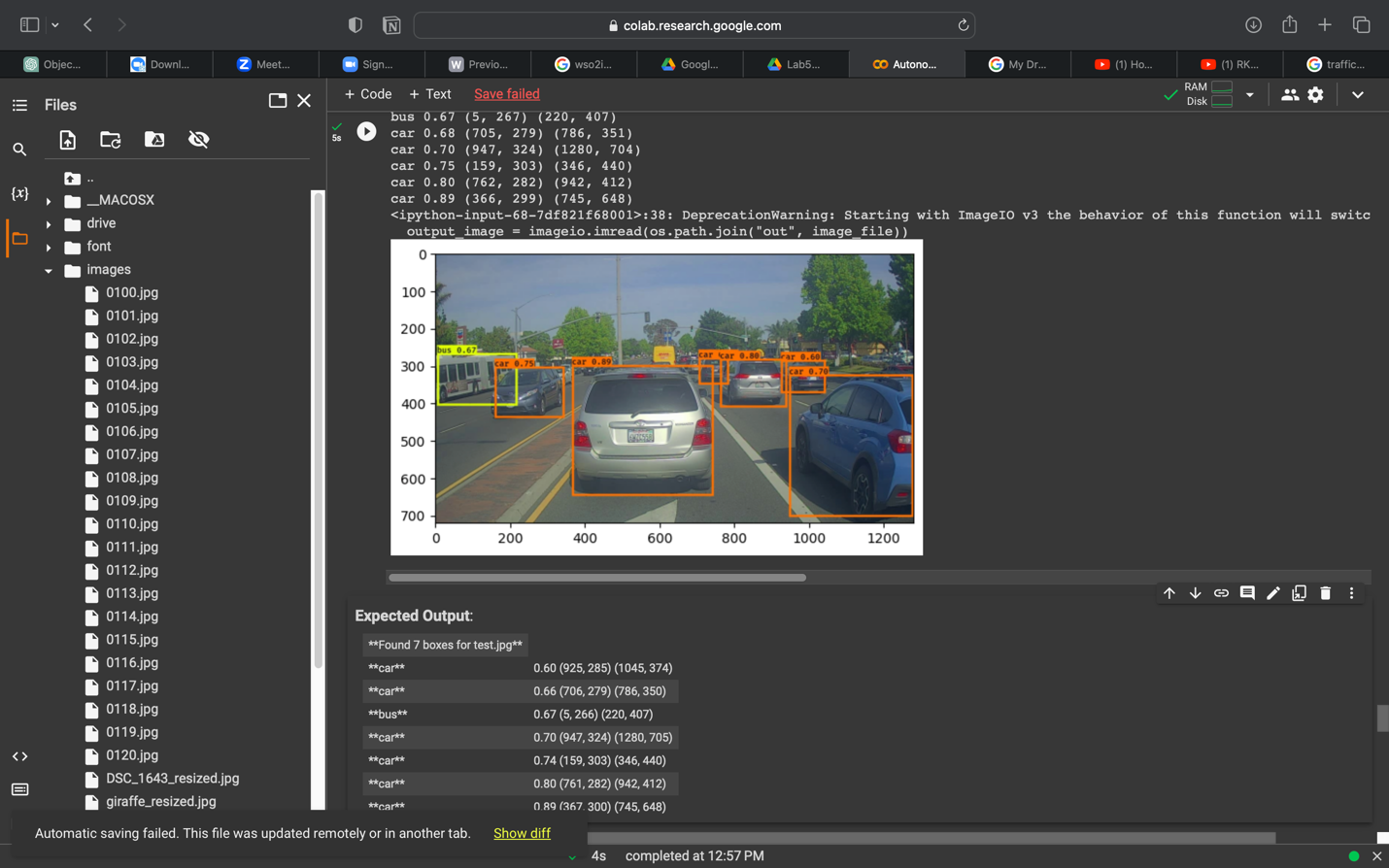
* + Change the score\_threshold [value between 0-1] to a different value but use the original values for other 2 variables. Rerun the required cells to get the output images for the autonomous driving dataset. Observe if this result in improvement compared to step 10 for the same two images. If there are any improvements, write them in the word file. Include the new 2 output images in the word file.

**score\_threshold = 0.8**

****

* + Change the iou\_threshold [value between 0-1] to a different value but use the original values for other 2 variables. Rerun the required cells to get the output images for the autonomous driving dataset. Observe if this result in improvement compared to step 10 for the same two images. If there are any improvements, write them in the word file. Include the new 2 output images in the word file.





**Submission.**

Download the final modified Autonomous\_driving\_application\_car\_detection.ipynb notebook. Add this notebook, the new image used in step 9 and the word file to a new zip file. Upload this zip file to the courseweb submission link. The file name should be your registration number.