Reflective Journal: Object Detection Using TensorFlow and Pascal VOC

In this lab, I worked on object detection with TensorFlow and the Pascal VOC 2007 dataset. The main goal was to learn how to not just recognize objects in a picture, but also show *where* they are by drawing boxes around them. This is different from image classification, where the computer only tells you the name of what is in the picture. Here, the model gives both the name and the location, so the output looks more complete.

I started by installing the tools I needed, like TensorFlow, TensorFlow Hub, TensorFlow Datasets, Matplotlib, NumPy, OpenCV, and PIL. These tools helped me load the dataset, run the object detection model, and show the images with boxes on them. I also checked the versions to make sure everything would run without problems.

After that, I loaded a small part of the Pascal VOC dataset. I only used 10% of the training and validation sets so the notebook would run faster. I grabbed the class names from the dataset so the model could label the objects. I also wrote a function to show some example images with the correct (ground truth) boxes. These boxes show what the dataset says is in the picture. When I looked at the examples, I saw green boxes around the objects, which helped me understand what the model is supposed to detect.

Next, I loaded the SSD MobileNet V2 model from TensorFlow Hub. I picked this model because it is fast and works well on low-power systems like Google Colab. It is a small model, which makes it quick, but the downside is that it is not as accurate as bigger models. It sometimes misses small objects or gets confused if objects look similar.

I also made a function that runs the detector on pictures and shows the predictions along with the ground truth boxes. I had to make one small but important change: sometimes the model gives a class ID that does not exist in the Pascal VOC dataset. When that happens, the code tries to look up a label that isn't there, and the whole notebook crashes. To fix it, I added a simple check that skips those invalid classes. This keeps everything running smoothly.

I then tested the model on a few pictures from the training set. I saw red boxes from the model and green boxes from the ground truth. The model did well with big and clear objects like people and cars. But when objects were small or partly hidden, the model had a harder time finding them. This made me see both the strengths and limits of SSD MobileNet V2.

Later, I wrote a function to test how well the model works. It compares predicted boxes with the real boxes and counts true positives (correct hits), false positives (wrong hits), and false negatives (missed objects). It also calculates precision and recall. Precision tells me how many of the predictions were correct. Recall tells me how many real objects the model found. I also made a small fix here by counting false negatives per image instead of doing it for the whole dataset at once. This gives more accurate results.

When I ran the evaluation, I got numbers for true positives, false positives, false negatives, precision, and recall. These numbers gave me a better idea of how the model performs when using only a small amount of data.

While working through the images, I noticed the model detects big, common objects better than small ones. Cars, dogs, and people were often detected correctly. But small items like bottles or objects hiding behind others were harder for the model. Some bounding boxes were not perfectly placed, and sometimes the model missed an object completely. Lighting, clutter, and the size of the object can all affect this. If I used the full Pascal VOC dataset instead of a small piece, the accuracy would probably be better because the model would see more examples.

The threshold value of 0.5 also affects what is shown. If the score is lower than 0.5, the box is not shown. A higher threshold means fewer boxes but more confident predictions. A lower threshold means more boxes but also more mistakes. The heatmap visualization also helped because it shows how sure the model is. Brighter areas mean stronger confidence, and darker areas mean the model is not sure.

Overall, this lab helped me understand how object detection works, how to use a pre-trained model, how to show the results, and how to measure performance. I also learned that small code changes can prevent errors. SSD MobileNet V2 is good when the hardware is limited, but it is not the most accurate model. This exercise helped me see what the model can do, what it struggles with, and how I might improve things in the future.