# Reflection and Analysis for Object Detection Lab

## Conceptual Understanding

In this lab, I learned that image classification and object detection have fundamentally different goals. Image classification involves identifying the primary object within an image and assigning a single label to it. In contrast, object detection goes beyond labeling—it also localizes multiple objects in an image using bounding boxes. This difference was clear in this exercise, as the model didn’t just label objects, but also marked each of them individually in the images, showing both their classes and positions.

We used the SSD MobileNet V2 model for its balance between speed and efficiency, which made it ideal for this task. I realized this model is well-suited for environments where computational resources are limited, such as with a single GPU. While SSD MobileNet V2 provides quick detections, it does have some limitations, especially when working with smaller or intricate objects, where more complex models might outperform it. However, in a resource-constrained setup, it strikes a good balance between accuracy and speed.

## Code Interpretation

The find\_images\_with\_classes function played an essential role by filtering images based on specified classes, which was especially helpful when working with a large dataset like COCO. With so many categories in the dataset, this function made it manageable to focus only on the classes relevant to our task and allowed the model to run faster by narrowing down the dataset. In the plot detection’s function, the threshold value directly impacted how many objects appeared in the output. Setting the threshold to 0.5 meant that only objects with a confidence score of 50% or higher were displayed. By raising this threshold, I could make the model display only the most certain detections, while lowering it showed even the less confident ones.

Using a heatmap visualization helped me understand where the model was focusing its attention within an image. This technique made it easier to see how confident the model was in certain regions, especially around the detected objects. It provided a clearer picture of whether the model was recognizing boundaries well or struggling with overlapping or blended objects.

## Observing Results and Limitations

Running the exercise multiple times showed that the model was consistently accurate with certain types of objects, like larger vehicles and animals, while it struggled with smaller or overlapping objects. This seemed to be due to the model’s limited resolution, which made it harder to capture fine details or distinguish objects that were very close together. I noticed some bounding boxes were slightly off or didn’t fully capture the object. These inaccuracies were likely due to factors like lighting, object occlusion, or backgrounds that blended with the objects. These issues made it difficult for the model to fully separate the object from its surroundings, highlighting some limitations of the SSD MobileNet V2 model.

If we had used the entire Pascal VOC 2007 dataset instead of a small subset, I believe the model would have performed better overall. With more examples to learn from, the model could generalize better and improve its object detection accuracy. However, working with the full dataset would require more computational resources and time.

## Critical Thinking

If I wanted to detect only specific types of objects, like animals or vehicles, I could modify the filtering step in the find\_images\_with\_classes function to only include those particular classes. This way, the model would focus solely on those objects, making it more efficient for targeted tasks. Training my own object detection model would require gathering and labeling a large dataset, selecting an appropriate model, and tuning the model’s hyperparameters. I’d also need to overcome challenges like dataset size, ensuring model generalization, and handling the high computational demands involved in training from scratch.

Despite some limitations, the SSD MobileNet V2 model could be quite useful in real-world applications that need quick processing but don’t require the highest accuracy. For example, it could be used in a monitoring system that detects vehicles or pedestrians in real-time without needing to analyze every detail.