The difference between image classification and object detection is that classification puts a label on the entire image and object detection identifies multiple objects in an image and draws bounding boxes around them, indicating their locations.

In this all the images have bounding boxes around them like boxing a person or a car or a car so it shows their location.

SSD MobileNetV2 is selected because it has a good balance between speed and accuracy, especially for environments like Google Collab. Some advantages are that it is lightweight and fast and can be done on a mobile device but some limits are that it is less accurate and it struggles with smaller overlapping objects.

The code find_images_with_classes would filter the dataset to only show images containing certain classes. It is useful because COCO and Pascal VOC are large datasets with many object types. This function helps reduce noise, save computation time, and focus analysis on relevant objects.

In the plot_detections function the threshold of 0.5 filters out low-confidence detections. Higher thresholds will have more confident detections and lower thresholds will have more detections but that can also include false positives.

The heat map shows the area of the image where the model believes an object exists with a higher confidence. The brighter areas are higher confidence and vice versa with the Dim areas. This help[s because it helps us understand why predictions are made and helps us find missed detections.

The most accurate detections in my case were the People, Cars, and the Horses. They were boxed in every time I did the exercise. They are very represented in training data. The more challenging ones were all the smaller objects and things that were overlapping. Also things in the background that just looked smaller due to the distance.

Yes, some boxes dont even touch the object and are too big or small and sometimes it can become mislabeled. The factors are usually the poor quality of the images or the object size and the overlapping.

I think it would have improved the model's generalization and increased the accuracy on some of the less common classes it can define; it will also have more diverse examples to learn from.

You could add a filter by label inside the visualization loop. Or use a find_images_with_classes function to focus on processing only relevant samples.

The steps you would need to take are, Collect and Label, Convert the data to format, Choose architecture, Set training config and start training, Fine tune for better results. Some challenges are the time it takes to collect and label all the data, it needs a lot of images for each class, it requires GPU resources.

Even though it has limits it still works well for Mobile Apps like barcode scanning and detecting wildlife on our cameras. It can be used for surveillance detecting a person or vehicle and even tech to assist people like the blind.