Jeffery Dirden
Professor McManus
October 28, 2024

L09 Reflection

This lab on object detection introduced essential concepts by using a pre-trained SSD MobileNet V2 model, making it accessible even with limited computational resources. Leveraging a pre-trained model means we could avoid the heavy computational demand of training from scratch. This approach allowed us to explore object detection without the need for a high-performance machine, focusing instead on understanding how the detection process works. The exercise covered key components like bounding boxes, class labels, and confidence scores, enabling us to see how models can classify multiple objects within a single image and accurately localize them with bounding boxes. A particularly interactive aspect of this lab was trying out detection on custom images, which was a hands-on way to deepen our understanding of object detection fundamentals.

The primary difference between image classification and object detection became clear during the exercise. Image classification involves assigning a single label to an entire image, while object detection takes it further by identifying and localizing multiple objects within an image, each with its bounding box and label. This difference was evident in the output as the model not only recognized the objects but also highlighted their locations within the image. SSD MobileNet V2 was chosen because it is optimized for performance on devices with limited resources. Its lightweight architecture strikes a balance between speed and accuracy, which is advantageous when running models in real-time or on devices without high-end GPUs. However, its limitations include reduced accuracy for detecting smaller or more intricate objects, a trade-off that is sometimes necessary when working with lower computational capacity.

Some code elements played specific roles in making the object detection process efficient, such as the find_images_with_classes function, which was crucial in filtering and retrieving relevant images from larger datasets like COCO. This function allowed me to focus on images containing specific classes, making it easier to manage a large dataset by loading only the images necessary for our analysis. Another key function was plot_detections, where the threshold value was set to 0.5 to filter out lower-confidence detections, ensuring that only objects with a confidence level above 50% were displayed. This threshold helped us visualize the most certain detections and understand how confidence impacts detection outputs. Additionally, the heatmap visualization provided insights into the model's confidence across different objects, visually indicating how certain the model was about each detection. The more intense the color, the higher the confidence, which was useful for assessing the model's reliability in various scenarios.

Running the exercise multiple times showed that the model performed best on larger, prominent objects like people and cars, while smaller or overlapping objects posed a challenge. Factors like object size, image resolution, and lighting could affect detection accuracy, especially with the SSD MobileNet V2 model, which isn't designed for fine-grained precision. Occasionally, the bounding boxes were inaccurate, especially when objects were partially obscured or the background was complex. These issues likely stem from limitations in the model architecture and training data, underscoring how challenging object detection can be in real-world settings.

If we had used the full Pascal VOC 2007 dataset instead of a smaller subset, we might have observed better generalization in the model. With more training data, models generally perform better due to exposure to a wider range of object types, angles, and backgrounds. This would potentially improve accuracy, although the pre-trained model might not benefit as much without further fine-tuning. To adapt the model for detecting specific objects, such as only animals or only vehicles, we could modify the code to include a filter that isolates certain class labels, improving relevance and efficiency by reducing the number of detections displayed.

If we wanted to train our own object detection model, we'd start with data preparation—labeling images with bounding boxes and class labels—followed by selecting a model architecture, adjusting hyperparameters, and setting up training and evaluation processes. Challenges would likely include the computational cost, the time required to label data accurately, and the potential need for a large dataset to achieve generalizable results. Despite these limitations, SSD MobileNet V2 could still be valuable in real-world scenarios where lightweight models are needed, such as on mobile devices or in applications where detection speed is more important than absolute precision.

If interested in exploring other models, TensorFlow Hub offers a variety of options like Faster R-CNN and EfficientDet, which provide different trade-offs between accuracy, speed, and resource requirements. In particular, Faster R-CNN can deliver higher accuracy, albeit at the cost of speed and resource usage. Running a few sample images through a more powerful model, such as YOLOv5, often results in better precision, especially with smaller or overlapping objects. However, these higher-end models are not always practical in settings with limited resources, illustrating why SSD MobileNet V2 remains a solid choice for lightweight applications. This lab made it clear that while accuracy is important, understanding the fundamentals of object detection, analyzing results critically, and working within resource constraints are equally essential skills for real-world applications.

Cited Sources

Confirm that TF2 is using my GPU when training. (n.d.). Stack Overflow. https://stackoverflow.com/questions/60542475/confirm-that-tf2-is-using-my-gpu-when-training

TensorFlow | ssd_mobilenet_v2 | Kaggle. (n.d.). https://www.kaggle.com/models/tensorflow/ssd-mobilenet-v2/tensorFlow2/ssd-mobilenet-v2/1?tfhub-redirect=true