Reflection

Overview

In this lab exercise, I explored object detection using a machine learning model to identify objects in images. I uploaded various images and observed the model's performance in detecting and labeling objects, specifically focusing on understanding their accuracy and limitations.

Key Activities

Image Uploading: I successfully implemented a function to upload images in Google Colab, which allowed me to test the object detection model on different images.

Detection and Visualization: I utilized a function to process uploaded images, which involved converting the image to a NumPy array, running the detector, and visualizing the results with bounding boxes and labels.

Observations

The model successfully identified some objects, like bicycles, but frequently returned "UNKNOWN" for others. This suggests that the model may not be trained in all relevant classes or that some class IDs fell outside the specified list of class names. Additionally, we did load a smaller subset of the PASCAL dataset, which could limit the model's ability.

The confidence scores varied significantly, revealing the model's strengths and weaknesses. For example, in one instance, the model confidently identified a bike in an image, but in another, it mistakenly labeled the bike as a bird, indicating a potential need for model retraining or fine-tuning.

Questions & Answers

1.Conceptual Understanding

Main Difference between Image Classification and Object Detection:

Image classification is all about determining what the main subject of an image is, while object detection goes a step further by not just identifying the classes of objects but also pinpointing their locations using bounding boxes. You can really see this distinction in the output, which shows both the detected classes and their bounding boxes.

Why SSD MobileNet V2 was Chosen:

I picked the SSD MobileNet V2 model because it's lightweight and works well on devices with limited processing power. It strikes a good balance between speed and accuracy, which is crucial for real-time object detection. However, it may not perform as well as larger models, especially in complex scenes or with smaller objects.

2.Code Interpretation

Role of find_images_with_classes Function:

The find_images_with_classes function helps filter images that contain specific classes. This is super handy when dealing with large datasets like COCO, as it lets you focus on particular categories without sifting through tons of irrelevant images.

Impact of Threshold Value in plot_detections: The threshold value (set to 0.5) plays a key role in determining which detected objects get displayed. A higher threshold might lead to fewer displayed objects, boosting precision but risking missed detections. On the flip side, a lower threshold could show more objects, but it might also result in more false positives.

Heatmap Visualization and Model Confidence:

The heatmap visualization offers insights into the model's confidence, highlighting areas where it believes it has detected objects accurately. Regions with higher confidence scores stand out, giving you a visual cue on how sure the model is about its detections.

3.Observing Results and Limitations

Types of Objects Detected Accurately vs. Challenging Ones:

The model generally performs better with larger, distinct objects because they have clearer features and contrast against their backgrounds. Smaller or partially hidden objects pose more challenges, often due to limited visibility.

Bounding Box Accuracy:

Sometimes, the bounding boxes might not align perfectly with the objects, particularly with overlapping items or those that are partially out of frame. Factors like image quality, occlusion, and the model's limitations in detecting certain shapes can contribute to these inaccuracies.

Accuracy with Full Pascal VOC 2007 Dataset:

Using the complete Pascal VOC 2007 dataset would likely boost accuracy due to the wider variety of training examples. A more extensive dataset helps the model generalize better, which reduces the chances of overfitting to a smaller data subset.

4.Critical Thinking

Modifying Code for Specific Object Detection:

To focus on specific objects, I could tweak the model’s output filtering based on class IDs or labels. This would involve adjusting the run_detector function to check against a specified list of desired class IDs.

Steps to Train Your Own Object Detection Model:

If I wanted to train my own model, I’d need to gather and annotate a dataset, choose a suitable architecture, set up training parameters, and run the training on hardware that can handle it. Some challenges I might face include data imbalance, overfitting, and ensuring I have enough computational resources.

Real-World Scenarios for This Model:

Despite its limitations, this model could be useful for basic applications like traffic monitoring or simple robotics tasks where speed is more critical than achieving perfect accuracy.

5.Going Further

Research on Other Object Detection Models:

There are various object detection models like Faster R-CNN, YOLO (You Only Look Once), and EfficientDet, each with its own pros and cons regarding accuracy and speed. For instance, Faster R-CNN is known for its higher accuracy but tends to be slower, while YOLO is optimized for speed, making it great for real-time applications.

Running Images Through More Powerful Models:

Testing images with more robust models like Detectron2 or YOLOv5 might deliver better accuracy and bounding box predictions, especially for challenging objects. You could expect improved localization, fewer false positives, and better handling of overlapping objects.

Conclusion

In summary, the exercise has provided valuable insights into the differences between image classification and object detection, highlighting the strengths and limitations of the SSD MobileNet V2 model. By understanding the nuances of threshold settings and bounding box accuracy, I've gained a clearer perspective on how to enhance model performance. The journey of modifying code for specific object detection and the potential of training my own model opens exciting avenues for practical applications. Exploring other advanced models further enriches this learning experience, paving the way for future projects that leverage the power of object detection in real-world scenarios. This not only fuels my curiosity but also prepares me for tackling more complex challenges in the field of computer vision.

Medium. (2021). Image Classification vs Object Detection: Understanding the Difference. Retrieved from https://medium.com/analytics-vidhya/image-classification-vs-object-detection-vs-image-segmentation-f36db85fe81

Medium. (2020). Exploring the SSD MobileNet V2 Model for Object Detection. Retrieved from https://vidishmehta204.medium.com/object-detection-using-ssd-mobilenet-v2-7ff3543d738d

Kaur, H. (2024). Understanding Confidence Threshold in Object Detection. Medium. Retrieved from https://medium.com/voxel51/finding-the-optimal-confidence-threshold-cd524f1afe92

Towards Data Science. (2019). R-CNN, Fast R-CNN, Faster R-CNN, YOLO: Object Detection Algorithms. Retrieved from https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

Detectron2 vs YOLOv5: Which One Suits Your Use Case Better? (2023). Retrieved from https://medium.com/ireadrx/detectron2-vs-yolov5-which-one-suits-your-use-case-better-d959a3d4bdf