1. **Conceptual Understanding:**

What is the main difference between image classification and object detection? How is this difference evident in the output of this exercise?

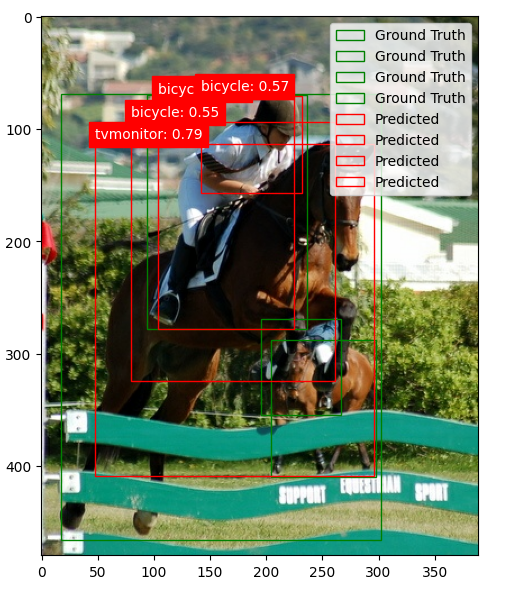
Image Classification and object detection sound similar, but they are different. Image classification is a technique of labeling the entire image. It will label one or multiple objects in the image without identifying the object's location. Object detection identifies the objects in the image and locates them by drawing bounding boxes around each object. It labels multiple objects along with the coordinates.

For example, in our previous lab on image classification, we got the following output. The program was trying to classify the images as chihuahuas or muffins. There was no mention of where the chihuahua was.

A collage of dogs

Description automatically generated

In this lab, when we ran the code to detect objects, the predicted boxes (in red) and ground truth boxes (in green) were drawn around the objects to indicate their locations within the image. In the example image below, the program successfully detected the horses and their riders, although it did not label them correctly. In this lab, we not only classified the image but also; by drawing the boxes, the program provided the coordinates showing the precise location of each object.



Explain why we chose the SSD MobileNet V2 model for this task. What are its advantages and limitations, especially in the context of limited computational resources?

We chose the SSD MobileNet V2 model for this task because it is a lightweight model, fast and efficient compared to larger and more computationally expensive models like Faster R-CNN or YOLO. It can detect objects quickly without requiring excessive computational power. It uses depthwise separable convolutions, resulting in lower memory usage and faster processing times. Single Shot Multibox Detector (SSD) used by MobileNet V2 is a single-shot object detection model used to detect multiple objects in one go through the network. It also comes with pre-trained weights on large datasets like COCO, which allows transfer learning. So, instead of doing the whole project from scratch, we can use pre-trained weights from COCO.

Although MobileNet V2 is fast and efficient, requiring minimal computational power, it may struggle to detect smaller objects in cluttered images that contain many elements.

**Code Interpretation:**

Describe the role of the find\_images\_with\_classes function. Why is it useful when working with a large dataset like COCO?

Large datasets like COCO contain thousands of images, making it impractical and time-consuming to manually search through each one for a specific class. The find\_images\_with\_classes function automates this process, allowing you to quickly and efficiently narrow down the images to those relevant to your task. Since large datasets require significant storage space and computational resources, filtering images by class enables you to focus only on the subset that meets your specific needs.

In the plot\_detections function, how does the threshold value (threshold=0.5) impact the number of objects displayed?

Threshold values play a significant role in object detection models. Images with a confidence score of **0.5 or higher** are displayed in the output. This confidence score reflects the model's certainty or probability that a detected object in an image belongs to a specific class. It indicates how strongly the model believes the detected object matches the predicted class. Adjusting this threshold helps fine-tune the balance between showing relevant detections and filtering out noisy or incorrect ones.

Explain how the heatmap visualization helps you understand the model's confidence in its detections.

The heatmap visualization provides an intuitive way to assess the model's confidence in its detections by visually highlighting where and how strongly the model believes objects are present in an image. In a heatmap, various regions of the image are color-coded, with warmer colors typically indicating higher confidence and cooler colors representing lower confidence. This visualization is useful for fine-tuning the model, as it allows us to analyze the cooler-colored areas to identify which parameters need adjustment, helping the model improve its object detection in those regions.

1. **Observing Results and Limitations:**

Run the exercise multiple times. Which types of objects does the model tend to detect more accurately? Which ones are more challenging? Can you explain why?

Although running the model about 6 to 7 times, the result did not get better, the model was still making mistakes in predicting the object. Even with 10% training data, its precision and recall were 0, and with lesser training data like 2%, the model did not improve. The background images were challenging but black car and galloping horse was taking most of the image. Still, the model was not able to detect it. In both evaluated images, the predicted box almost perfectly matched the ground truth box, so this model does exhibit Localization Accuracy over Intersection over Union but Classification Accuracy is not there. Since the model has misclassified the object, its Precision and Recall are zeroA screenshot of a computer program

Description automatically generated

Observe the bounding boxes. Are there any instances where the boxes are inaccurate or miss the object entirely? What factors in the images might be contributing to these errors?

In the lab, the boxes are frequently inaccurate. Typically, image detection software can identify larger objects like cars with less variation, but this model struggled to detect the car and horse correctly. For instance, the horse was misidentified as a bicycle. Although the predicted and ground truth boxes overlap, the labels were incorrect. The red truck behind the black car was labeled as a boat, while the horse was identified as a bicycle, and its jockey was labeled as a TV monitor.A screenshot of a screen capture

Description automatically generated

A car with its hood open

Description automatically generated

How would you expect the accuracy of the model to change if we had used the entire Pascal VOC 2007 dataset instead of a small subset? Why?

If we use the entire Pascal VOC 2007 dataset. The model will have more variety of data to learn about patterns and appearance of the same objects. This will help generalize the model better for different objects and scenes and improve its accuracy. This would also reduce the risk of overfitting. It would help in the fine-tuning of the model as well. This model will come across different variations of cars and horses, like riding or galloping horses, and open or closed trunks of cars. If it comes across this variation of images during training then it would greatly increase its accuracy. However, large datasets may also increase noise and complexity during training. If the data is not carefully curated, it may impact the accuracy.

1. **Critical Thinking:**

How could you modify the code to detect a specific set of objects, like only animals or only vehicles?

To detect only a specific set of objects, we must create a subset to filter out the categories that we need. We must filter the annotation to only include classes that we want to detect. Then, we have to modify model’s output layer to match the number of classes. For example, if we are planning the model to detect only vehicles and animals, then we have to specify the number of animal classes, such as “dog”, “cat”, “horse”, sheep” and vehicle classes as well.

The next step will be to modify the training data to load only images that contain those specific objects. After that, we have to retrain and fine-tune our model so that the model can learn to detect only the specified classes.

If you wanted to train your own object detection model, what steps would you need to take? What are some challenges you might encounter?

To train my object detection model, I must do the following tasks in specific order to achieve successful results:

* 1. Select object classes for my model to detect.
  2. I must determine whether I want to use the bounding box detection method or the segmentation method.
  3. To train my model, I must gather large and diverse datasets like Pascal VOC or COCO or Open images.
  4. Then I will label or annotate the images using tools like CVAT or VGG Image Annotator. This annotation should include class labels and coordinates of the bounding box.
  5. Then split 70% data for training, 15% for validation, and 15% for testing.
  6. According to the available computational power, I will either choose YOLO, R-CNN, or RetinaNet to pre-train data
  7. To train the model, I will adjust the learning rate, batch size, number of epochs and hidden layers.
  8. I will then track the training and validation loss to ensure the model is learning and not overfitting. I will evaluate the trained model using mean Average Precision (mAP) or intersection over Union (IOU).
  9. Test a separate dataset to make sure my model is detecting the objects in the images.

Challenges Encounter:

* 1. The biggest challenge in object detection model training would be ensuring the quality of the data. For the model to perform well, it is essential to collect a balanced set of image samples with an appropriate representation of each class.
  2. Overfitting can also occur if the model is not performing well on unseen data.
  3. Some detection model requires significant computational resources as GPU or TPU and can take days or even weeks to train the model, if we do not have enough computational power.
  4. To fine-tune hyperparameters, it takes a lot of experimentation and domain knowledge.

Given the limitations of this model, in what real-world scenarios might it still be useful for object detection?

While this model may not be fully sufficient, it has the potential to improve with more training data and computational resources. Object detection models can be applied in areas such as surveillance and security, retail and inventory management, and manufacturing and quality control. Despite its limitations, the model can still be valuable for specific tasks.

1. **Going Further (Optional):** (Bonus points)

Research other object detection models available in TensorFlow Hub. Compare and contrast them with SSD MobileNet V2 in terms of accuracy, speed, and resource requirements.

**Faster R-CNN**

Faster R-CNN is another object detection model that provides higher accuracy than SSD MobileNet V2. It is specially designed for complex or dense object detection tasks. It uses a Regional Proposal Network (RPN) that is efficient in object localization and detection accuracy. It is slower than SSD MobileNet V2 and needs a powerful GPU. It is not suitable for real-time applications.

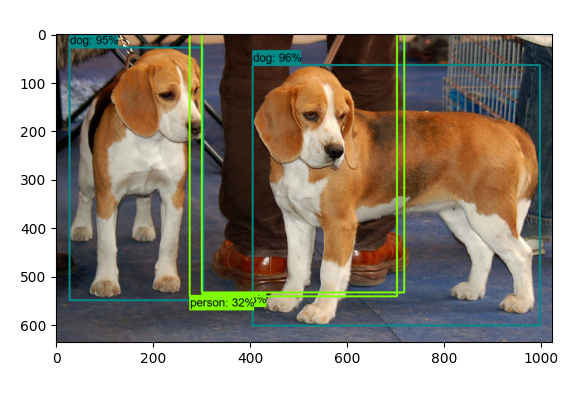
**EfficientDet**

EfficientDet provides a good trade-off between accuracy and efficiency. It requires less computational power than Faster R-CNN, but it is slower than MobileNet V2(Tan et al., 2019).

**YOLOv4**

YOLOv4 is known for its high accuracy, and it is useful in detecting objects in large-scale or complex datasets. It is optimized for both speed and accuracy, performing well in real-time applications. It is still slower than MobileNet V2 but it is highly accurate. It requires substantial computational power and is therefore not suitable for smaller devices (Bochkovskiy et al., 2020).

* Try running a few images through a more powerful object detection model online (if available). Compare the results to the output of this exercise. What differences do you notice?
* I did not have the computational power to run other TensorFlow detection model, but I found a tutorial with test images that show with more computational power a good balance of accuracy and efficiency can be achieved from model detection model (*Object Detection From TF2 Saved Model — TensorFlow 2 Object Detection API Tutorial  Documentation*, n.d.)



A person walking on a beach

Description automatically generated.

**Work Cited**

Tan, M., Pang, R., & Le, Q., V. (2019, November 20). *EFfIcientDET: Scalable and efficient object Detection*. arXiv.org. <https://arxiv.org/abs/1911.09070>

Bochkovskiy, A., Wang, C., & Liao, H. M. (2020, April 23). *YOLOV4: Optimal speed and accuracy of object detection*. arXiv.org. <https://arxiv.org/abs/2004.10934>

*Object Detection From TF2 Saved Model — TensorFlow 2 Object Detection API tutorial  documentation*. (n.d.). https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/auto\_examples/plot\_object\_detection\_saved\_model.html#sphx-glr-auto-examples-plot-object-detection-saved-model-py