Angel Candelas

Prof. P. McManus

ITAI 1378

July 27, 2024

L09 Object Detection using Transfer Learning and Pascal VOC 207 Dataset

In this lab we went through the process of demonstrating object detection using a pre-trained model from TensorFlow. As with the last few labs, we started by installing any necessary libraries (TensorFlow, TensorFlow Hub, TensorFlow Datasets and Matplot). Then, we used a -pre-trained SSD MobileNet V2 detection model which is good for limited resources. Finally, we were able to evaluate the model performance and display detected objects with bounding boxes. This lab took a lot longer than those that we’ve done before, although, it was still quicker than the original L09 that would time-out because of the size. I tried changing the code a few times to try different numbers of pictures to see different results, but unfortunately, Colab cut me off after the second run stating that I reached the maximum usage for the day.

The difference between image classification and object detection is that image classification assigns a label to an image and tells you what is in the image whereas object detection identifies and locates multiple objects in an image and gives both class labels and bounding boxes for each object. It tells you what is in the image and where it is in the image. We can see the differences in the output of this exercise because instead of only getting a label for an image, we can also see the bounding boxes that are drawn around detected objects. Each box also has its class label and confidence score to demonstrate the object detection capability.

We chose the SSD MobileNet V2 model for this task because of its efficiency with environments with limited resources. Some of its advantages include efficiency, speed and good accuracy. MobileNetV2 is lightweight and designed for mobile and embedded devices. It is computationally less demanding than larger models. Even though it is lightweight, it still provides reasonable accuracy for object detection tasks. Some of the limitations of using the SSD MobileNet V2 model include sensitivity to image quality, limited object categories and lower accuracy when compared to larger models. Using this model can have an effect on an image’s resolution or lighting conditions. It also may not be able to recognize objects due to its limited training dataset. Also, compared to a larger model, it may not be able to get to the same level of accuracy as the larger model.

There are a couple of roles for the find\_images\_with\_classes function in this lab. The first being that it filters images. It goes through the dataset to find images that contain specific object classes. The other role is that it creates efficient subsets. Rather than processing an entire data set, it creates smaller sets to focus on relevant images. This function is also useful when working with large data sets like COCO because it helps with performance, optimization and analysis. Since it trains on smaller data sets, this can lead to better performance for specific object detection tasks. In addition, datasets like COCO can have millions of images and processing all of them can take lots of computational power and can be very time consuming. This function helps alleviate that with targeted analysis. In the plot\_detections function, the threshold value controls how many objects are displayed by filtering detection results based on their confidence score. The value that is used acts as a cutoff point and only detections with a confidence score higher than that threshold will display. For example, if we use a low threshold value of 0.3, more objects will display but it will include those with low confidence scores, whereas a higher threshold value such as 0.7 will show fewer objects but may have more accurate detections. Heatmaps can be very helpful in understanding the model’s confidence in its detections. Heatmaps use color intensity to help with visual representation. Colors like red and yellow typically indicate higher confidence in a detected object and colors like blue and green suggest lower confidence. You can also overlay the heatmap on top of the original image to see which areas of the image the model is most confident about.

Unfortunately, I was not able to run the lab multiple times at first because Colab kept cutting me off due to usage limitations (I am including a screenshot of the error message at the end of this journal after my sources). When I did run it the first couple of times before being cut off, I noticed that it kept trying to label the horses as bicycles or tv monitors. I understand maybe the bicycle, but I thought the tv monitor was way off (and a bit funny). When I ran it a second time with a higher threshold value, it got rid of the tv monitor, but still had the bicycle label on it. I think with a higher threshold and another run, it would have gotten rid of the bicycle label as well. If the entire Pascal VOC 2007 dataset was used instead of the small subset, I think that the model’s accuracy would improve. The reason I think this is because there would be more examples from the larger dataset for the model to learn from.

We could modify the code to detect only specific objects like animals or vehicles by filtering the detection results based on the predicted class labels. We would need to determine the class IDs that correspond to animals and vehicles using the class\_names list. Then we would need to add a condition to only display detections that belong to the animal and vehicle Class IDs.

The steps to train my own object detection model would include data preparation, choosing the model architecture, training the model, and evaluating and fine-tuning. In the data preparation step, I would need to put together a large and diverse dataset by collecting images that are relevant to the objects that I want to detect. I would need to label each object in the images with bounding boxes and class labels and then I would need to divide the data into training, validation and testing sets. Next, I would need to choose a model architecture such as the SSD that we used in this lab. I would then need to train the model to optimize the model’s parameters as well as evaluate the model on the validation set to prevent overfitting. I would also need to evaluate and assess the model’s performance on data using metrics like precision. This would also need to be where I can make adjustments to the model’s architecture to improve performance and results. One of the challenges I think I would encounter with this is putting the data together. Having to gather lots of images and manually add the bounding boxes and labels would be very time consuming with a large dataset. Another challenge that I think I would encounter would also depend on the size of the datasets. If they are extremely large, it would require lots of computationally power to run.

The SSD MobileNet V2 model used in this lab can be useful in real-world scenarios even with its limitations. If computational resources are limited, the model can be used on devices with lower processing power like smartphones. The model’s efficiency also makes it good for tasks where quick detection is needed like in video analysis or object tracking. It can also be used to test object detection during initial steps before spending more time building a more complex model. A couple of examples where this model can easily be applied are when you are trying to find specific objects in an image or when you want to count how many objects are in an image.

Sources:

<https://labelyourdata.com/articles/object-detection-vs-image-classification>

<https://www.analyticsvidhya.com/blog/2023/12/what-is-mobilenetv2/#:~:text=Firstly>

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10378376/>

<https://pub.towardsai.net/building-your-own-object-detector-from-scratch-with-tensorflow-bfeadfaddad8>

Colab error: A screen shot of a computer

Description automatically generated