MSDS 462 Final Project — Object Detection with VGG Neural Network

James Dowd
Northwestern University

As Machine Learning for Computer Vision becomes more and more advanced, I've still been fascinated by some of the basic tasks that have become somewhat routine, so routine in fact, I can train them on my own computer and deploy it to my iPhone so that I have a useful tool with me at all times. One of those tasks is object detection in images, and for this final project, I chose to develop a model that could detect boats, bicycles, cars, trucks and taxis. I thought this was an interesting task to understand the development process and training process, while building something useful I could test while walking around the neighborhood.

To build this model, I first gathered training data from Google's Open Images Dataset V4. After downloading a file with the relevant links for each image and the annotations associated with that image, I then attempted to create a randomly selected set of 1000 images from each of the five selected classes and downloaded those images for use in the training process. Some images were unable to be processed, either because they were no longer available or there were errors in associating the annotations with the image, but we ended up with a relatively balanced training set with 959, 947, 959, 957 and 933 images for boats, bicycles, cars, trucks and taxis, respectively.

Having researched possible architectures for this task, I identified region-based convolutional neural networks (R-CNN) as a high-performing method of object detection that was a bit more specialized than some of the methods I had tried on previous projects in this course and others. R-CNNs were highly successful in theory, but "were computationally expensive as originally developed" (Ren, et. al.). Because of these performance issues, Ren, et. al. proposed Faster R-CNN in 2015, published in 2016, which uses input images represented as Height x Width x Depth tensors (Tryolabs) that are passed through a pre-trained CNN and passed to an intermediate layer which provides the convolutional feature map that becomes the feature extractor for the next part — similar to Transfer Learning.

To implement this process, I identified prior research and coding by Yinghan Xu in 2018 (Xu), who built off a base built by Yann Henon (Henon), which I was able to customize for my purposes. I was successful at being able to build and train a model, but was getting unsatisfactory results when it came to accuracy, as loss continued to decrease, but accuracy remained relatively flat throughout the training process. With some tweaks to learning rates and other parameters, I was able to generate marginal improvement, but ultimately was unsatistifed with the results. Additionally, when working to run against the test set and deploy to a CoreML file, I was having difficulty making it work end to end.

Given those difficulties, I narrowed the scope to include only Bicycles and Cars and then used Apple's Turi Create (Apple) to train a final model and then convert it to the required format for use in an iOS app. I modified the Apple Breakfast Finder app (Apple) to have a custom skin and to replace the standard model with my newly trained model and had successful results. After deploying to my iPhoneX, I took the phone out to the streets and was able to successfully capture and identify moving and stationary cars and bicycles with strong accuracy.

I plan to continue building on this format to ultimately try and implement the Faster R-CNN in an iOS environment. I'm pleased with the progress I made at customizing that infrastructure for my needs, and then being able to shift priorities to get a working product out the door. I have a strong interest in building this out further for alert type of techniques or to use in controls for robots, as I have interest in better understanding autonomous vehicles.

Link to Github Folder of Notebooks

https://github.com/jvdowd/msds462_final_project

References

Apple. “Recognizing Objects in Live Capture.” *Recognizing Objects in Live Capture | Apple Developer Documentation*, developer.apple.com/documentation/vision/recognizing_objects_in_live_capture.

Apple. “Turi Create API Documentation¶.” *Turi Create API Documentation - Turi Create API 6.3 Documentation*, apple.github.io/turicreate/docs/api/.

Hennon, Yan. “Yhenon/Keras-Rcnn.” *GitHub*, 25 Aug. 2017, github.com/yhenon/keras-rcnn.

Ren, Shaoqing, et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, 2017, pp. 1137–1149., doi:10.1109/tpami.2016.2577031.

Tryolabs. “Faster R-CNN: Down the Rabbit Hole of Modern Object Detection.” *Tryolabs Blog*, Tryolabs, 18 Jan. 2018, tryolabs.com/blog/2018/01/18/faster-r-cnn-down-the-rabbit-hole-of-modern-object-detection/.

Xu, Yinghan. “Faster R-CNN (Object Detection) Implemented by Keras for Custom Data from Google's Open Images...” *Medium*, Towards Data Science, 25 Feb. 2019, towardsdatascience.com/faster-r-cnn-object-detection-implemented-by-keras-for-custom-data-from-googles-open-images-125f62b9141a.

Xu, Yinghan. “RockyXu66/Faster_RCNN_for_Open_Images_Dataset_Keras.” *GitHub*, 8 Apr. 2020, github.com/RockyXu66/Faster_RCNN_for_Open_Images_Dataset_Keras.