ROB 599 Perception Project

Group 10 Members: Ajaay Chandrasekaran
Valerie Chen
Eric Harding
Royce Hwang
Prashin Sharma
Ted Xiao

Description of Approach

We used Google's Object Detection API, which is an open source framework built on top of Tensorflow that is designed to simplify the process of training a neural network to detect and draw bounding boxes around objects. After some preliminary research and testing to familiarize ourselves with the system, we chose to go with a fast R-CNN with 50 layers (frcnn-50), since we felt that that architecture offered the best accuracy at a reasonable speed. Due to the small size of our dataset, we concluded that larger networks, such as fast R-CNNs with 101 or 152 layers, would lead to more extreme overfitting.

The data that we used all came from the full dataset containing the pictures and bounding boxes. We wrote a program to convert all of that data into TFrecords files that could be read in by Tensorflow. For our ground truths, we simply took the maximum and minimum x and y values to represent the four corners from the 8 points of the bounding box data given to us. This simplified the process, since we did not have to deal with its pose, orientation, etc. Additionally, we decided to exclude any images that had bounding boxes exceeding the dimensions of the image, because we would have to significantly increase the maximum allowed bounds in the Object Detection API, which we felt would potentially slow training for questionable gain. To maximize the size of our training set, almost all of the labelled pictures were reserved for training, with only around 60 used for evaluation.

To speed up the process, we took a pretrained frcnn-50 network that was already trained on the COCO dataset, removed the last fully connected layer, and replaced it with one corresponding to our own set of car classifications. The model was trained on an AWS p2.8xlarge instance containing 8 GPUs, using an batch size of 16 for approximately 30,000 iterations. Data augmentation included random horizontal flips, and cropping. Along the way, we evaluated the accuracy on our eval dataset to determine when to stop training. Once it reached a satisfactory mAP (0.9 or above), we exported the model (frozen inference graph), and ran it on our test dataset and submitted the results. The detection threshold that yielded the best results was 0.6.

From there, we realized that our first model had overfitted the dataset, so we repeated the process of taking a pretrained model and changing the last fully connected layer. This time, however, we simplified it further and simply used a binary "Car/Not_Car" classification scheme to speed up training. The model was trained for around 22,000 iterations, before being used to generate and submit our best results (as a result of not overfitting).

References

"Speed/accuracy trade-offs for modern convolutional object detectors."
Huang J, Rathod V, Sun C, Zhu M, Korattikara A, Fathi A, Fischer I, Wojna Z,
Song Y, Guadarrama S, Murphy K, CVPR 2017

Tensorflow Models and Object Detection API:
https://github.com/tensorflow/models

Direct Link to Object Detection API:
https://github.com/tensorflow/models/tree/master/research/object_detection

Helpful Tutorial Link:
https://medium.com/@WuStangDan/step-by-step-tensorflow-object-detection-api-tutorial-part-1-selecting-a-model-a02b6aabe39e