Calvin Cramer

EEC 193 Lab 3

Jan 31, 2018

PHASE 1

***Q1. How does your choice of loss affect the accuracy of your bounding box regressor. Specifically, what is the accuracy difference between L1 or L2 loss? Which works better and why? (To properly answer this question you will need to understand what is happening both conceptually as well have done some experimentation on your model).***

The loss function does not affect the accuracy of the final model, but it does affect how quickly the model takes the train and converge. Here are the final train and test accuracies for each loss function:

Training accuracy L1: 89.7%

Testing accuracy L1: 78.1%

Training accuracy L2: 89.7%

Testing accuracy L2: 78.1%

I changed the accuracy threshold to 65%, but before with the original 60% the accuracies were very close to 100%, which didn’t give much information. The L2 loss is preferable, since it’s a devalues being farther away from the correct bounding box more heavily the farther away the prediction is, rather than L1 loss which devalues the error linearly. The result is that using L2 will converge the training faster than L1, but both loss functions will end up in the same trained state given enough time.

***Q2. What is the purpose of IoU? Why is it so important in object detection?***

The purpose of IoU is to measure how well bounding boxes are aligned. It is basically a measure of “how close are these two rectangles aligned”. Without IoU, the obvious other metric to go by would be loss, but that does not tell any visual information, and also loss could be any number but that does not tell how well the bounding boxes are aligned. IoU is so important because it will tell us how well our model performs.

***Q3. The current model in this phase can only do single object detection. How would you transform this model to handle multiple object detection? (There are multiple valid answers to this. You should compare the merits of each method.)***

First of all, you need a dataset that has bounding box labels for all of the images.

Method 1: Add more output nodes to the final output layer, 4 nodes for each additional object you want to detect.

Method 2: Have a separate network trained for each different type of object. In phase 1, we are only tracking a single car, but we can run 1000 simultaneous models and track 1000 simultaneous objects, but only at most one of each type of object.

Method 3: If you only want to detect multiple cares then you can split the image into a grid, and run the single object detection network on each grid.

Method 1 would be the best of all three options, but it has a set number of objects it can detect. It has the advantage of the bounding boxes being able to overlap as well, whereas method 3 will have bounding boxes only within each grid cell. Method 3 has the huge disadvantage when the grid cells are too small, and an object spans multiple cells, that the bounding boxes, even if perfectly predicted, there will be multiple bounding boxes for each object, which is visually unappealing, and the network isn’t actually finding a single box for a single object.

Method 2 is terrible in terms of efficiency, but method 1 and 3 should be about as fast as the original model in phase 1. In terms of changes to the dataset, method 1 and 2 would just require all bounding boxes for all important objects in the image, but method 3 will need the bounding boxes inside of every single grid cell, and even worse, there needs to be a different dataset for each grid size. Don’t use method 3. Method 2 is too slow. Method 1 would work the best.

PHASE 2