Vehicle detection based on YOLOv2 algorithm

Deep learning brings a revolution to the object detection field, yet the method taught in the class is still using HOG and SVM. So I decided to tackle this problem using deep learning approach. From my learning, there are several deep learning algorithm for objection detection, like R-CNN, fast R-CNN, faster R-CNN , mask R-CNN , YOLO, SSD etc. I'm going to implement YOLO because it's pretty fast which is very important in real time task like vehicle detection for self driving car. However I don't have a computer with gpu, training yolo from zero is an impossible task for me. So my plan is to use a trained weight and fine tune it on the udacity dataset.

Brief introduction to YoloV2 algorithm

YoloV2 is quite complex algorithm for object detection based on CNN. I'm trying to describe it here as short as I can.
YoloV2 takes input image sized as 608*608*3 and output a feature map of 19*19*5*85. This feature map contains a lot of information…including whether there is a object in some location, what class is it and location and size of the target box. Because I only want to predict 2 class not 80 class, so output of my version of yolo is 19*19*5*7.
Let me explain the output in more detail. 19*19 here means the algorithm divided the image in 19*19 grids. If a target's center is located in this grid, then this grid is responsible for the detection of this target. The first 2 axis is the coordinate of my information map. So the information in [0][0] gird refers to what my algorithm learned for the first upper and left grid.
What is 5*7 about? 5 is the number of targets a grid trying to detect. YoloV2 author used K-means to figure out 5 is the best number one grid should detect and he also calculated the best shape of ground truth box for these 5 boxes.
Under every box, there are 7 numbers. 1st element is the probability of there is target in this grid in the size of this box. 2,3nd elements are the x,y coordinates of the middle of the predicted box. X and y is normalized by the size of the grid, so x, y is between -1 and 1. If your output x,y is 0 , it means that your predicted boxes located exactly in the middle of this grid. If it's -1,-1, then box is located at the bottom left point of this grid. 4,5th elements are the width and height of the predicted box. Notice that we have 5 ground-truth boxes related to 5 boxes for each grid. So the equation below is used to calculate the exact width of the box. Height calculation is the same way.

$$\text{Width}_i = e^{(bwi)} * \text{anchor\_box}_i$$

Width_i denotes the width of i_{th} box for a grid. bw_i denotes the width output number of i_{th} box. anchor_box_i denotes the ratio of the i_{th} anchor_box.
6,7th elements are the class classification for the predicted box. There were a lot class when the author implement the yolo algorithm, I kept 2 only 2 class: car, and truck.
To get the output from input image, a vgg like deep convolution network is built. Unlike in vgg, they used fc layers in the end to predict the final result. Instead, YOLO used 1*1 convolution to do this task so the algorithm can keep the location information of the convolution network. A skip_connection technique is applied inspired by object detection algorithm SSD to improve the result.

Data preparation

Udaicty provides us with 2 dataset for vehicle detection and I'm going to use them both. So first I read them from csv file and concatenate them into 1 dataset. As you can see below, in the dataset, for each line, we have xmin, xmax, ymin, ymax and class information. I need to convert these information into a target data which I can use for training later. As mentioned in first chapter of this report, the target data should be the size of 19*19*5*7 , same with the output of my network. I need to place every box in the right grid according to their location and need to place them in the right anchor box after calculating the IOU(Intersection over Union). I don't want to go into more details with the process because the code in my notebook file should be clear to understand.

Network building

The structure of yolo model is in yolo.png. It's too large to paste here☹. I implemented my own version of this network. But as I have to use trained weights, my version is not used. Both weights and model is loaded from yolo.h5 file. Original yolo model outputs a feature of size 19*19*425(19*19*5*85). Because I only want to predict 2 classes: car and truck, the size of my output is 19*19*35(5*7). I changed the output size of original yolo model by defining last 2 convolutional layers.

Loss function building

Loss function for yolo is quite complex. In fact this part is hardest part for me. Tensorflow is just not a good tool for complex math. Implementing loss function forced me to understand every digit in the output. There are several part in loss function including coordinates loss, classification loss, and confidence loss.

$$\begin{aligned}
&\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \\
&+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i\right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}$$

Figure1: Loss function for YOLO model

Let me explain it in detail. Coordinates loss is used for penalize the wrong position and size of the bounding box. For position error, sum-square error is used. But for size error, according to the paper, square root is applied to width and height of the box before calculate the loss. Because we want the large box with higher loss and small box with lower loss. If there is no object in this grid,

we will not computer coordinates loss in this grid.
For confidence loss , we treat the grid cell with and without object differently. For grid with object , we will computer the target confidence equals Best IOU with ground truth box. For grid without object, if Best IOU with anchor box is bigger than 0.6, then the loss is ignored. A lower parameter is multiplied to the loss without object to avoid model instability.
For classification loss, if there is no object, we don't have this loss. I used cross entropy loss for the classification loss.

Training for Transfer learning

Because the original yolo model weights works perfectly on our project video, implementing the loss function and doing transfer learning is more for practice.

First, I build 2 convolutional layers to replace the last 2 layers from yolo model. Yolo model gives an output of 19*19*5*85 and mine is 19*19*5*7. Then I freezed all the weights from yolo-model so during training only weights in my custom layer is updated…yet after several epochs of training, loss stays around 7, outputed a nonsense prediction.

So I began to reflect what is going wrong. I rechecked my loss function, my model…everything seems all right. Conclusion is that the problem is not about my code. Cutting down the output class number is easy for picture classification network but it seems it's not that easy for an object detection problem. We have a lot information in the dataset not just class but also the box size and location. It should be normal that Udacity dataset is quite different from the dataset yolo model used for training. So maybe it is just too hard for a single layer to learn such complex different mapping. Or maybe it's something deep in math? I still don't have a solid conclusion.

Then I changed my way of transfer learning. This time, instead of giving up the last 2 layers of the yolo model, I added a reshape and clipping layer after the whole yolo model. There is 4 coordiantes digits ,1 confidence digits and 80 class digits in the orginal output. I just clipped 7 useful digits for my model. 4 coordiantes, 1 confidence digits and 2 class digits(car and truck). In general, what am I doing this time is to leave everything in yolo model untouched except the weights of 7 neuron. After training, the loss declined a bit and still gave out the right prediction.

Discussion

Yolo is a much better solution for object detection problem but requires a ton of training most people can't afford. It's fast and accurate. But there are also some disadvantage for this algorithm. If you have more than 1 obejct in a single grid with similar shape, YOLO could only detect only one of them because one grid only output 1 target in a anchor box. This in fact happens a lot in the udacity database. Better solution to this is pixel wise object segmentation algorithm, like recently published Mask R-CNN. But it's not that fast as YOLO. I think some better and faster object segmentation algorithm will come out soon. This is Cambrian period for deep learning☺