Bus detection project – Compute vision 0510.6251

By: Roy Hirsch 305052920, Mor-Avi Azulay 305685067

Submission date: 5/1/2019

# Introduction

The given problem at the project that we were ask to solve was buses detection and classification between 6 classes, in addition to match a bounding box (BBOX) for each bus detection in the original input image space. We were supplied with 60 train images with at least one bus (could had more than one bus), which we separated to three sets: train, evaluation and test.

When we first approached the problem, we decided to start with a basic model of a known state-of-the-art neural network, perform transfer learning on the pre-trained model and fine tune the network’s weights by further training of the network with our data base.

We thought that the right choice is to start implementing the R-CNN model with VGG-16 as its base network. Then, if the results will not satisfy us we can try to improve the model to the known advanced methods of R-CNN, Fast R-CNN and Faster R-CNN.

# R-CNN

To understand better how the R-CNN model works we present in Figure 1 its stages:

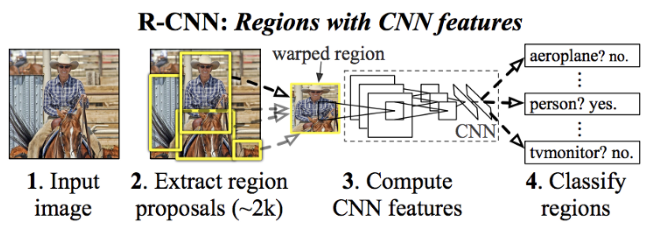


Figure : R-CNN stages.

We can see that for each input image we perform a selective search in order to extract region proposals (thousands per each image) with a condition of IOU>0.5 to be labeled as true, than we inject each region proposal to a pre-trained VGG-16 network and changed it by truncating the classification layers and add new of our own to achieve a new output layer built of 2 heads, one for classification with two outputs (bus or not)\* and BBOX regression with 4 outputs (four transformations of the top-left-x,top-left-y,width and height) reaching a total of 6 outputs.

\*At that stage we have decided to do the classification by colors after the networks prediction if there is a bus or not.

In Figure 2 we present the R-CNN outputs method for each region proposal:

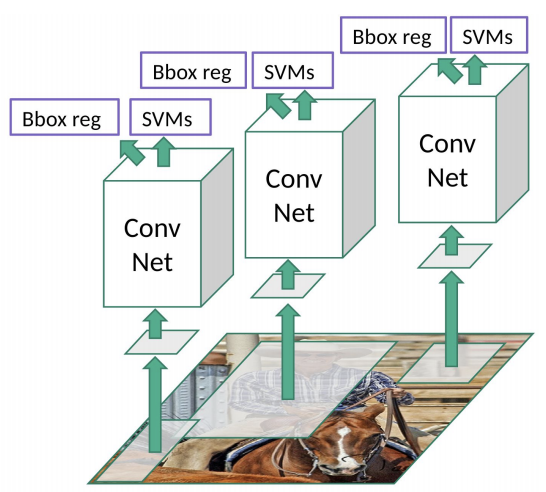


Figure : R-CNN classification (SVMs) and BBOX regression outputs for a proposed region input.

# Training R-CNN

The training level fine tune was operated only on the new classification layers, we froze the learning for the rest of the model, taking in consideration that our data set is not large and we cannot fine tune the whole network.

When we started training the R-CNN model we have encountered a few difficulties. The first and the major one was that we did not get good results and the training was not working perfectly. Other difficulties were that we don’t have enough data to train the model to the desired state (tried unsuccessfully to solve this problem with augmentation on the train set), the region proposals extraction level was too slow and used a lot of memory.

At the evaluation level we used the fast region selection mode of the region proposal block, the network made predictions and after we used non-maximum suppression to predict only the predictions with the highest confidence. One image evaluation took approximately 6 minutes on regular CPU which in our opinion, that is far too slow then what our goal is.

# Nest Step

At this point we chose not to proceed with the current model and there were a few options ahead of us such as improve the model to fast/faster R-CNN, use a different model (e.g. YOLO, SSD etc.) or perform classic computer vision methods (without a network). We noticed that improving to one of the advanced R-CNN model will require the same work frame as switching the model and finally we have decided to implement a new model based on a pre-trained SSD (Single Shot MultiBox Detection) network.

# SSD

The SSD approach is based on a feed-forward convolutional network that produces a fixed-size collection of bounding boxes and scores for the presence of object class instances in those boxes, followed by a non-maximum suppression step to produce the final detections. The early network layers are based on a standard architecture of VGG-16 used for high quality image classification (truncated before any classification layers), which we will call the base network.

The SSD architecture combines predictions from feature maps of various resolutions to achieve comparable accuracy to Faster R-CNN, while using lower resolution input images.

SSD only needs an input image and ground truth boxes for each object during training. In a convolutional fashion, it evaluates a small set of default boxes of different aspect ratios at each location in several feature maps with different scales (Figure 3 gives an example).

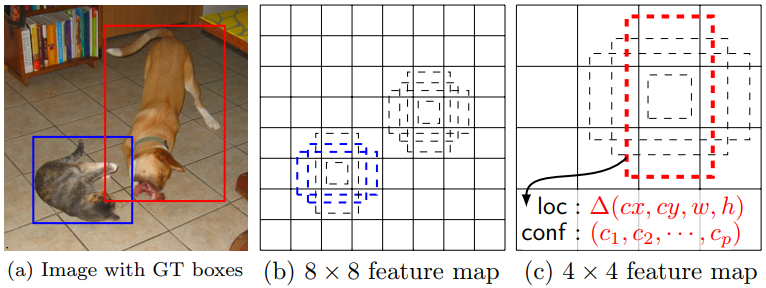
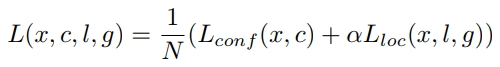
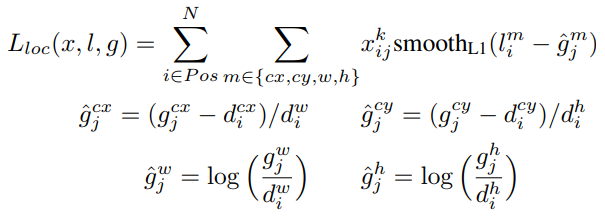
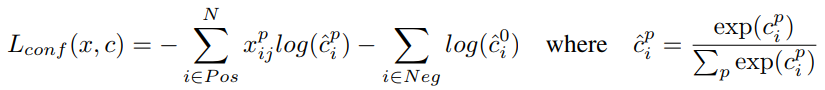


Figure : SSD default boxes from different scales and ratios. Colored boxes are labeled as true.

For each default box, the network predicts both the shape offsets and the confidences for all object categories. At training time, we first match these default boxes to the ground truth boxes. The model loss is a weighted sum between localization loss (Smooth L1) and confidence loss (Softmax):







Where are the ground truth parameters, are the correlated default bounding box parameters, are the predicted box parameters and are the classification outputs.

The architecture of the network is presented in figure 4:

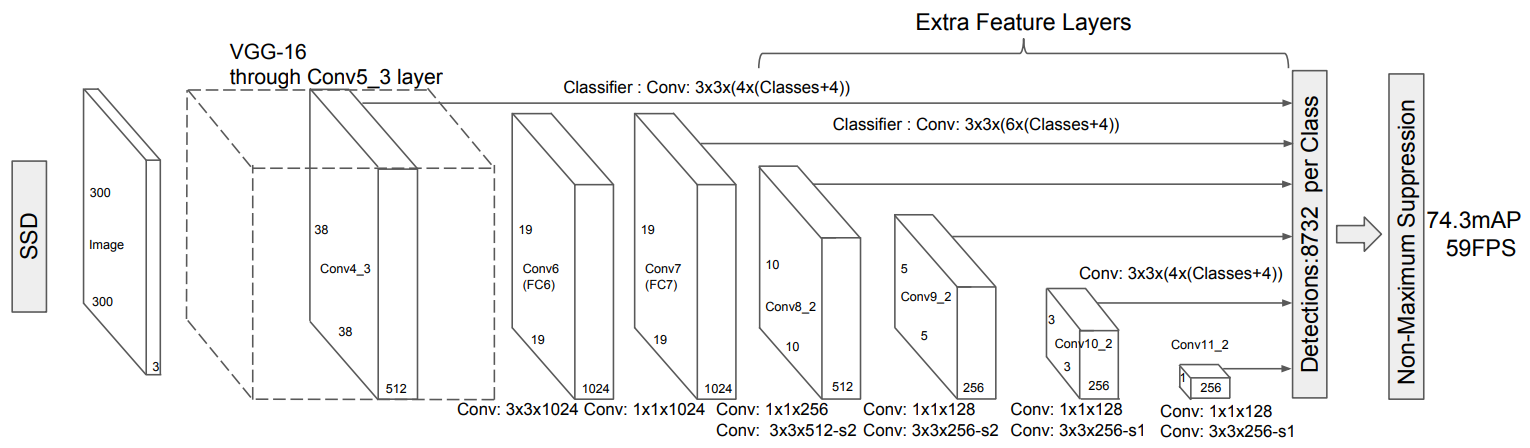


Figure : SSD architecture.

The architecture is built from an input layer followed by a pre-trained VGG-16 network truncated before classification layers and after, convolution layers where the output layer is injected by outputs of multiple layers in order to achieve a wider comprehension of the image features. At the end of the model there is a non-maximum suppression level.

# Training SSD

In our model we used a pre trained SSD model (COCO dataset – 80 classes), we have reconfigured the model with the right parameters and by extracted 7 classes’ weights, which are the following: background, bicycle, car, bus, truck, train and boat. 6 for the required 6 classes and one for the background. In contrast to Figure 4, our input size to the model was 512x512x3 (instead of 300x300x3), it was shown that better image resolution leads to better detection of small objects.

In order to enlarge our data set we made augmentations of the model inputs such as flipping, resizing, cropping or photometric distortions, each in a varying probability.

For each training epoch, we made 1000 steps using a batch size of 8 and a learning rate (Adam) of 0.00001. In addition we also made an early stopping after only 5 epochs

After several adjustments of the model parameters we have finally got good results on evaluation and test levels, in manners of classification and bounding box prediction within a very short time, especially when running on GPU.

# Conclusions

In this project we used transfer learning techniques on a pre-trained deep neural network and manage to achieve high results of buses detection and bounding box prediction. The SSD model is simple to use and comprehend, and on top of all it is efficient in memory and running time.