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:

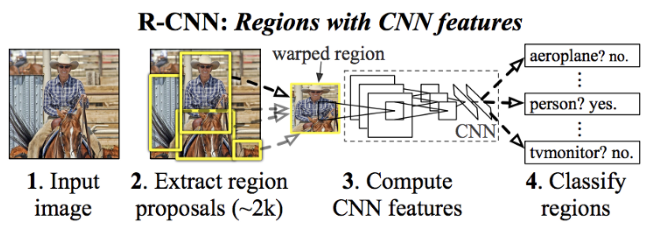


Figure 1: R-CNN stages.

For each input image we extracted regions of interest using selective search algorithm (thousands per each image). We conditioned with regions of interest with IOU higher than 0.5 with ground true labeling to be labeled as true. Then we warp and inject each region proposal to a pre-trained VGG-16 network with two new heads. One head for classification with two outputs (bus or backroad) and a head for bounding box regression with 4 outputs (four transformations of the top-left-x, top-left-y, width and height). The positive region of interest are inserted into a non-maximum suppression block to extract the final predictions.

We also decided to simplify the classification task into binary classification and deal with the color classification in a different manner. In Figure 2 we present the R-CNN outputs method for each region proposal:

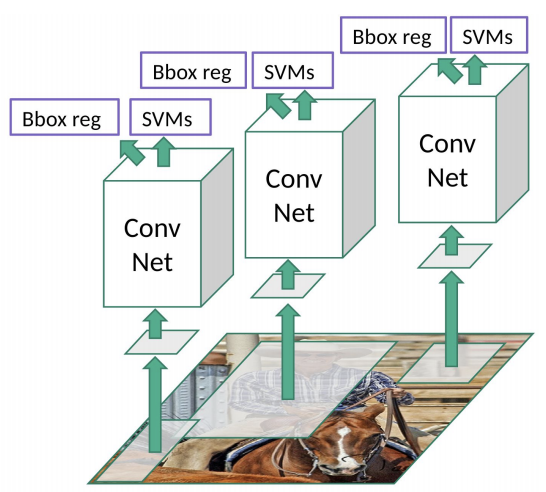


Figure 2: R-CNN classification (SVMs) and BBOX regression outputs for a proposed region input.

We implemented the whole model and data pipeline in Pytorch using the original article as a reference.

# Training R-CNN

We only trained the two new classification and regression heads while freezing the backbone VGG16 backbone (we used pre-trained VGG16). We chose to use transfer learning and not end-to-end training because of data scarcity. The classification and regression heads where trained separately.

When we started training the R-CNN model we encountered a few difficulties. The first and the major one was that we haven’t got good results and the training was not working perfectly.

In order to solve this problem, we applied augmentation on the training set. As a result, we got better results but still got lots of false positives. After tuning the non-maximum suppression block we got better results but encountered bigger problem. The runtime during evaluation was very slow.

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 six minutes on regular CPU, which was not satisfying in our opinion.

# Nest Step

At this point we chose not to proceed with the current model thought two next steps:

* Improve our base model into faster R-CNN
* Use a different model (e.g. YOLO, SSD etc.

We decided to implement a new model based on a pre-trained SSD (Single Shot MultiBox Detection) network. We choose to use the SSD model in order to achieve better runtime and to increase our model’s accuracy.

# 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. Next the predictions are inserted into non-maximum suppression block 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).

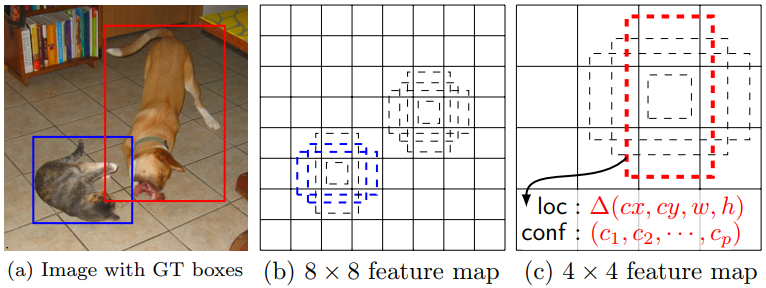
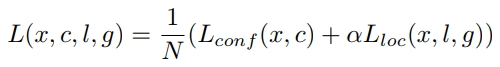
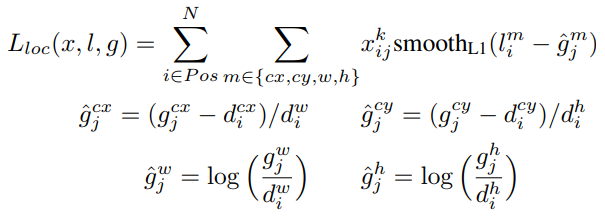


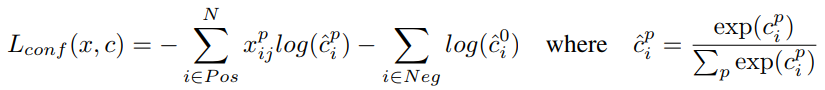
Figure 3: SSD default boxes from different scales and ratios. Colored boxes are labeled as true.

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:

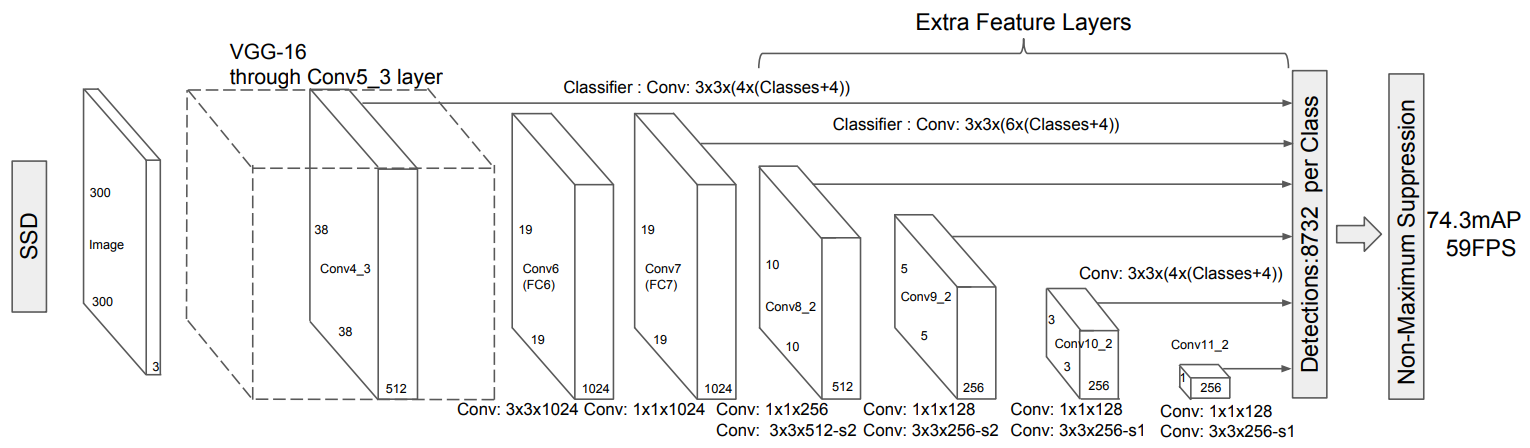


Figure 4: SSD architecture.

The architecture is consisting of an input layer followed by a pre-trained VGG-16 network truncated before classification layers. Next a set of few convolution layers are applied. The final layer aggregates the multiple convolution 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

We used a pre trained SSD model (trained over COCO dataset – 80 classes), we have reconfigured the model with the right parameters. Moreover, we used pre-trained weights, we extracted seven of the classes weights, which are the following: background, bicycle, car, bus, truck, train and boat. Six will be used for object detection and the last one for background. We wanted to take advantage of a pre-trained weight as a starting point to our train. 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 training set 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 got satisfying results on the evaluation and test sets. At the evaluation step the model achieves promising results with faster inference runtime.

# Conclusions

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