VEHICLE DETECTION USING FASTER R-CNN

BY

PRAKHAR DOGRA

Abstract

Main objective of this project is to understand and use a state-of-the-art architecture to train a model that can detect vehicles. Here detecting vehicles in images simply means localizing vehicle(s) in an image using rectangular bounding boxes (using x and y coordinates of two opposite corners of the rectangle) and classifying said bounding boxes into a particular label, i.e, type of vehicle like car, pickup truck, etc. For our project we have used state-of-the-art Faster R-CNN (Region-based Convolutional Neural Network)[1] to train on the MIOvision TCD (Traffic Camera Dataset) [2] that includes images of vehicles captured using CCTV cameras during different times of the day. During training we used weights of two state-of-the-art convolutional neural networks, VGG-16 [3] and ResNet-50 [4], (pre-trained on ImageNet dataset). Then the trained models were tested on separate images and their results (Mean Average Precision scores) are compared. We also used data augmentation techniques like flipping and rotating which resulted in significant improvement in the results.

Introduction and Related Work

The development of intelligent traffic surveillance systems has emerged as an important issue in the past decade with the introduction of state-of-the-art object detection deep learning architectures. The MIOvision traffic camera dataset that was recently released is a large scale dataset for vehicle classification and localization. The images have been taken in real time traffic surveillance environments. Our goal in this project is to study and use state-of-the-art Faster R-CNN architecture for vehicle detection. Our approaches also include data augmentation and comparing the results by using pre-trained weights of two state-of-the-art Convolutional Neural Networks (VGG and ResNet).

R-CNN family of architectures, as the name suggests, work on region proposal based techniques. Before going directly on to Faster R-CNN [1] architecture we would like to get an intuition and discuss the limitations of R-CNN [5] and Fast R-CNN [6] in order to better understand the Faster R-CNN architecture.
R-CNN takes an image as input, performs selective search algorithm on image to extract regions of interest and passes each of these regions of interest to separate Convolutional Neural Network (CNN) (notice here that the each region proposal is trained on a separate CNN) to extract feature vector. Finally, these features are consumed by a binary SVM trained for each class independently. The positive samples are proposed regions with IoU (intersection over union) greater than a certain threshold, and the negative samples are the irrelevant others. And in order to reduce the localization errors, a regression model is trained to correct the predicted detection window on bounding box correction offset using the CNN features. Here, the regression model fine-tunes the coordinates of predicted bounding box and the classification model classifies the detected region into a single label. Looking through the R-CNN learning steps, you can easily find out that training an R-CNN model is expensive and slow, as the following steps involve a lot of work:

- Running selective search to propose approximately 2000 region candidates for every image
- Generating the CNN feature vector for every image region (Number of images times the number of proposals per image (approximately 2000)). The whole process involves three models separately without much shared computation:
    - the convolutional neural network for image classification and feature extraction
    - the top SVM classifier for identifying target objects and
    - the regression model for tightening region bounding boxes.

Fast R-CNN, as name suggests, it is quite faster than R-CNN. It extracts features using CNN (i.e, pre-trains on image classification tasks) and then pass it to a ROI (region of interest) pooling layer. The ROI pooling layer outputs fixed-length feature vectors of region proposals. But just like R-CNN it also requires selective search method to propose regions of interest. Finally the model branches into two output layers:

- A softmax estimator of K + 1 classes (same as in R-CNN and +1 is the “background” class), outputting a discrete probability distribution per ROI.
- A bounding-box regression model which predicts offsets relative to the original RoI for each of K classes.

As you can see, due to sharing the CNN computation this architecture trains and tests faster than the original R-CNN but is still slower because of selective search algorithm.

The need for a selective search algorithm was removed in Faster R-CNN network with the introduction of Region proposal networks that uses the concept of Anchors, Intersection over Union and Non-Maximum Suppression to generate region proposals.

The dataset we have used in this project is MIO-TCD (MIOvision Traffic Camera Dataset) which contains 110000 images with ground truth labels (type of vehicle and (x, y) coordinates of rectangular bounding boxes) out of which 90000 were used for training and validation and 20000 were used for testing. Using this data our objective was to train an object detection model that can detect vehicles i.e, localizing the vehicle with rectangular bounding boxes and classifying it to a specific type of vehicle like car, pickup truck, bus, etc. Application of this project will help detect traffic flow such as which type of vehicles are are passing through a particular road and how many are passing in order to redirect routes. Moreover, being an object detection problem it has great applications in self-driving cars.

Technical Approach

To understand the architecture of Faster R-CNN [1], we are going to discuss some key concepts such as data augmentation, anchors, Intersection over Union (IoU).

Data Augmentation:

Data augmentation is one of the techniques that is used to improve the performance of computer vision systems. Even for a single object in image instance might have different size, shape, lighting conditions because of number of factors such changing in angle, distance,

device or lighting condition etc. while capturing a picture in another instance. Sometimes dataset does not cover all such instances of particular object. Data augmentation is a technique which generates such instances by using image processing techniques such as scaling, translation, rotation, flipping or adding noise. In our case we have used rotation and flipping techniques for data augmentation. For our project, we trained the models with and without data augmentation and observed a significant increase in Mean Average Precision score when using data augmentation.
Also, for the purpose of our project data augmentation was used on 50% of randomly selected examples.

Anchors:
Anchors are nothing but sliding windows with different scales and aspects ratios. Following figure 1 shows 9 different anchors with 3 different scales and 3 different aspect ratios. We can use any number of anchors whereas in our case we used 9 anchors as shown in figure 1.

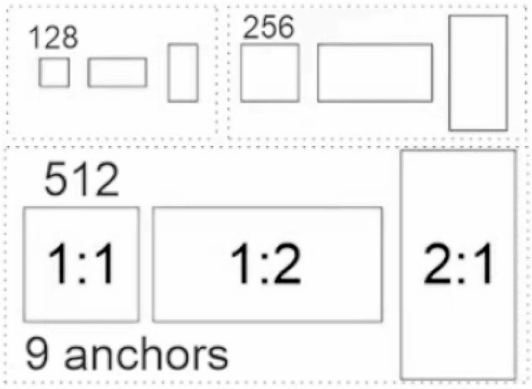

Figure 1:Example of anchors different scales and aspect ratios

Intersection Over Union (IoU):
Intersection over Union is simple evaluation metric. Any algorithm that provides predicted bounding boxes around object can be evaluated with respect to ground truth using IoU. The IoU can be explained with the help the left image in the following figure 2 where red bounding box represents ground truth and blue bounding box represents predicted bounding box by algorithm (any algorithm). We just take intersection of two bounding boxes and divide it with union of two bounding boxes i.e. ground truth box and predicted bounding box. It will give score between 0 and 1. If score large i.e. approaching 1 then predicted box is more accurate. Also, we use some threshold value for IoU score so that we only keep those predicted bounding boxes who meets the criteria of threshold. For the purpose of our project, we used a threshold of 0.7.

Example: Intersection Over Union (IoU)	Example: Non-Maximum Suppression
	

Figure 2: Examples of Intersection Over Union (IoU) and Non-Maximum Suppression

Non-Maximum Suppression:

Let's assume, our algorithm has detected object and we got multiple bounding boxes around that object. Also, all these bounding boxes are satisfying some threshold criteria 'T' for confidence or probability. The visualization of this situation is shown in right image of figure 2 where 3 bounding boxes are remaining from predicted bounding boxes after applying threshold. Then non-maximum suppression will select the box with highest confidence and will discard the boxes which are overlapping with selected box which detects its as of same class. For the purpose of our project, we used a threshold of 0.5

Region Proposal Network (RPN):

If we use sliding window and slide it all over the whole image to classify whether it contains an object or not then it will be very inefficient. Also, there will be lot of cases where the windows which we are processing has no object at all. In R-CNN paper, author tried to solve this problem by proposing regions using selective search algorithm. But it turns out that training and testing R-CNN architecture is still extremely slow. Faster R-CNN uses RPN to propose regions letting go of the need for the selective search algorithm.

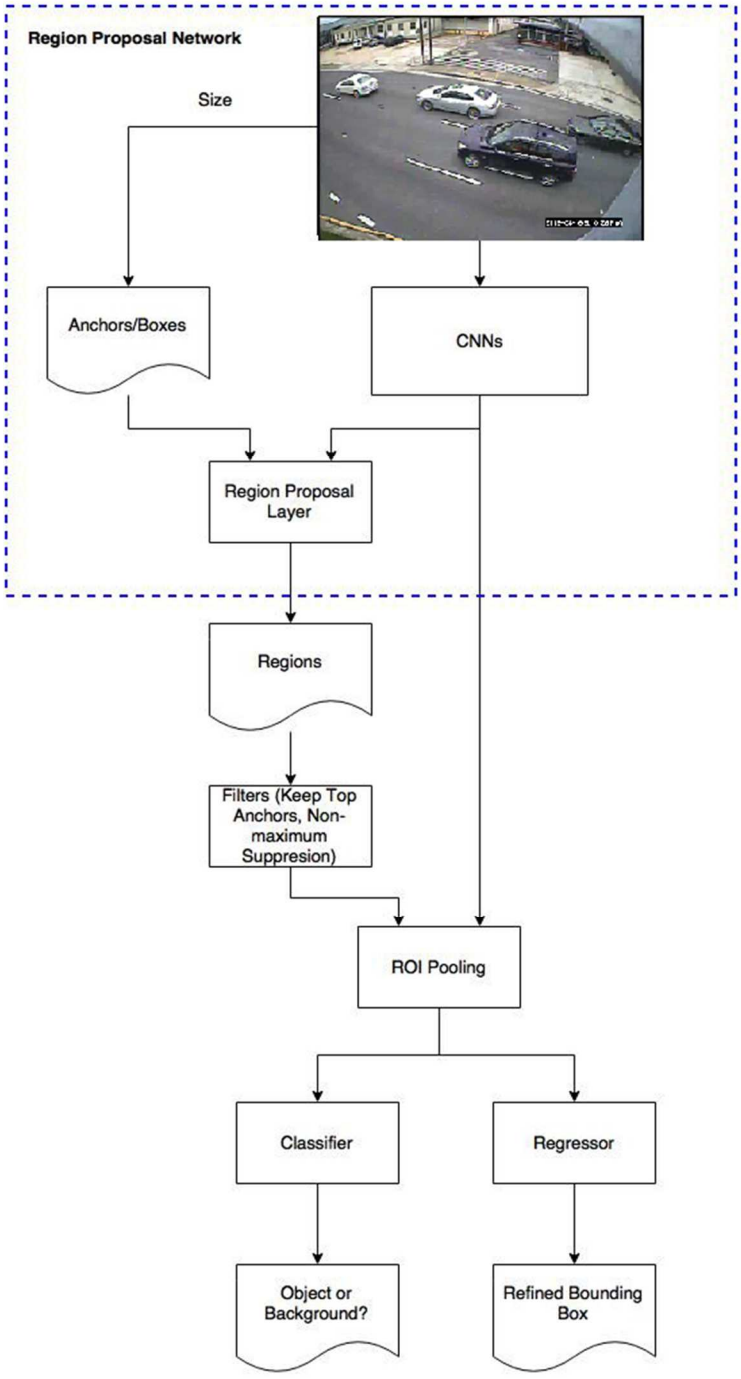

Figure 3 : RPN Highlighted n Faster R-CNN [8]

Architecture of Faster R-CNN:

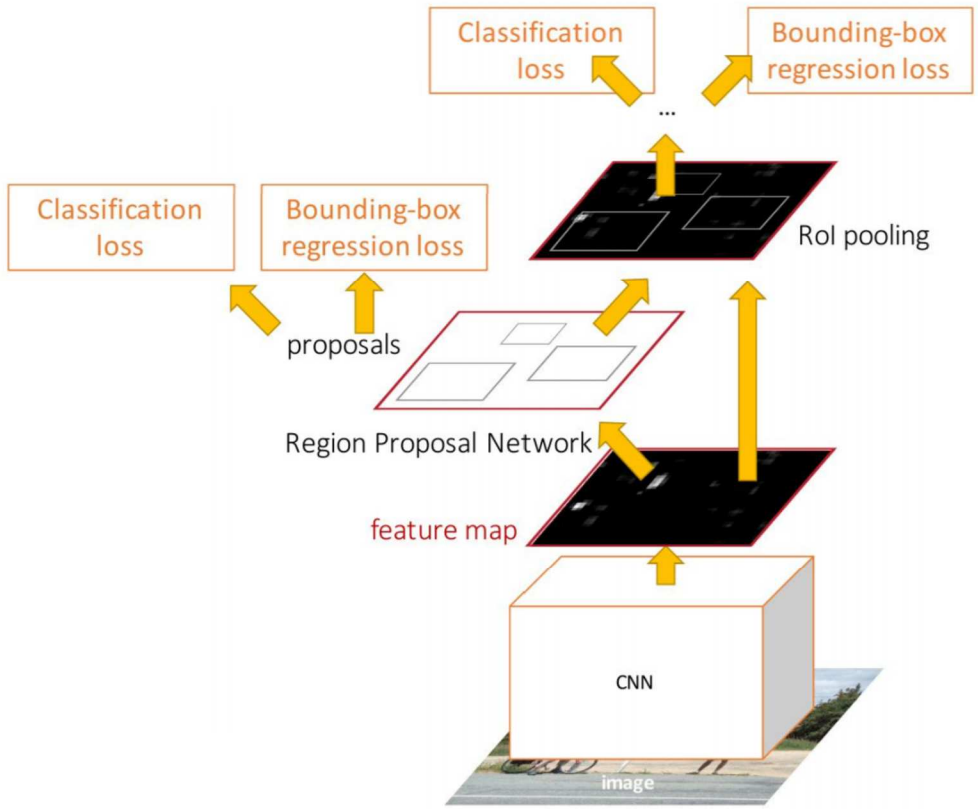

Figure 4: General architecture for Faster R-CNN [7]

Using RPN is much faster than using Selective Search during test time. Briefly, RPN ranks region boxes (called anchors) and proposes the ones most likely containing objects. The architecture is as follows.

The output of a RPN has one or more proposals or, we can say, bounding boxes that are examined by the classifier and regressor to eventually. It also predicts the possibility of an anchor being background or foreground.

Loss Function

Faster R-CNN is optimized for a multi-task loss function, similar to fast R-CNN.

Symbol	Explanation
pi	Predicted probability of anchor i being an object
p*i	Ground truth label (binary) of whether anchor i is an object
ti	Predicted four parameterized coordinates.
t*i	Ground truth coordinates
Ncls	Normalization term, set to be mini-batch size (~256) in the paper
Nbox	Normalization term, set to the number of anchor locations (~2400) in the paper.
λ	balancing parameter, set to be ~10 in the paper (so that both Lcls and Lbox terms are roughly equally weighted)

The multi-task loss function combines the losses of classification and bounding box regression:

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{box}}$$

$$\mathcal{L}(\{p_i\},\{t_i\}) = \frac{1}{N_{\text{cls}}}\sum_i \mathcal{L}_{\text{cls}}(p_i, p_i^*) + \frac{\lambda}{N_{\text{box}}}\sum_i p_i^* \cdot L_1^{\text{smooth}}(t_i - t_i^*)$$

where Lcls is the log loss function over two classes, as we can easily translate a multi-class classification into a binary classification by predicting a sample being a target object versus not. Lsmooth1 is the smooth L1 loss.

$$\mathcal{L}_{\text{cls}}(p_i, p_i^*) = -p_i^* \log p_i - (1 - p_i^*)\log(1 - p_i)$$

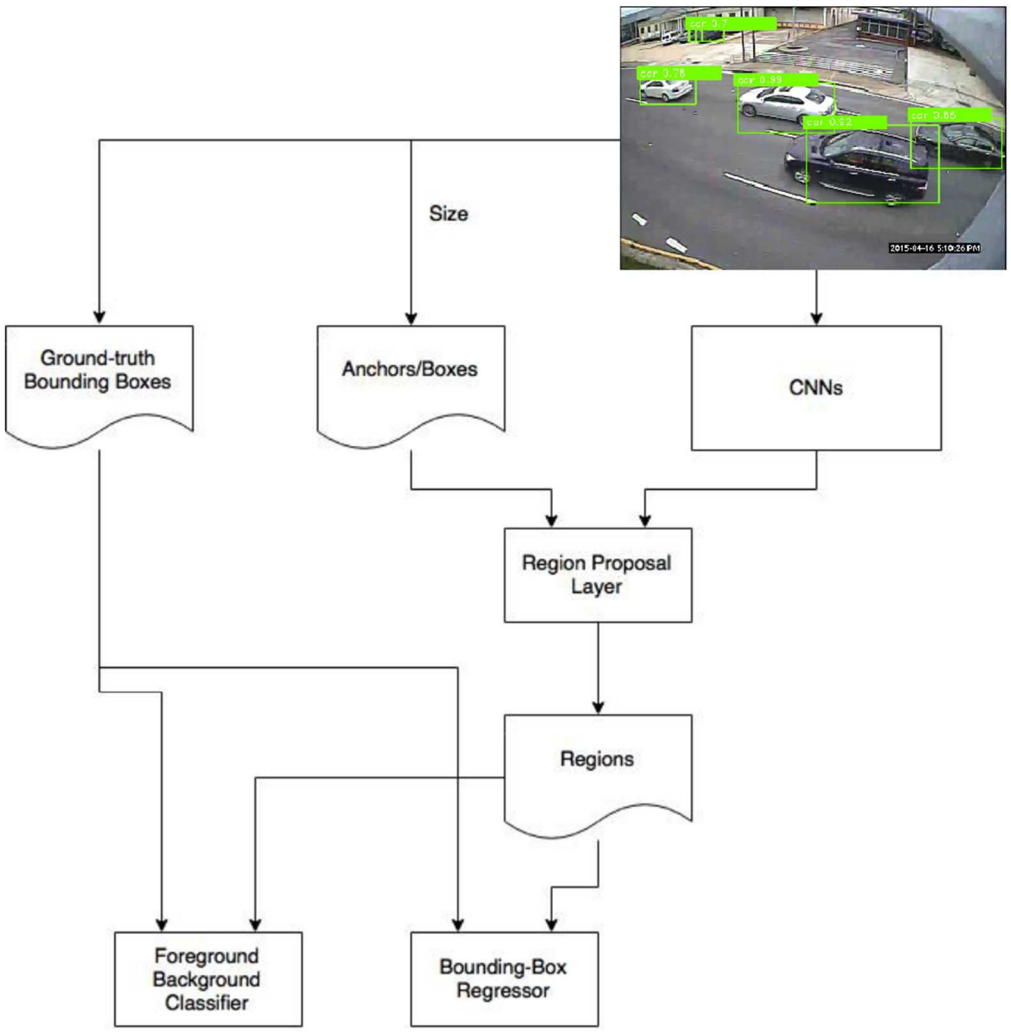

Figure 5 : Region Proposal Network [8]

Results

Results (Mean average precision scores):

	VGG	ResNet
Without Augmentation	0.443	0.451
With Augmentation	0.527	0.560

Figure : Results (Mean Average Precision scores)

Good Results:

Figure 6 : Examples of good results

Bad Results:

Overlapping Bounding Boxes	Unable to detect any vehicle	Incorrect Detection (detecting a roadway sign as a vehicle)

Figure 7: Examples of bad results

Discussion

From this project, I was able to learn about object detection. Starting by learning about one of earliest deep learning object detection architectures helped me understand about object detection in great detail. Also learning about the R-CNN family of architectures was challenging. But as I read the research papers on R-CNN then Fast R-CNN and then finally Faster R-CNN, I was able to understand them much better. I believe we could have tried using some other pretrained model weights or training on other state-of-the-art object detection architectures if we had more time. Also I would have liked to know if RPN could be improved to produce faster results because the current method is able to detect objects in images in 0.2 seconds at best (barely 5 frames per second).

Conclusions

From this project we conclude that Faster R-CNN, as an object detection technique, can be applied on a dataset like MIOvision TCD that has images of different types of still or moving vehicles captured at different times of the day (morning, afternoon, evening and night) i.e, different lighting conditions. It is observed that using slightly complex architecture improves the result a bit (also training time of ResNet is about half of that when using VGG). And using data augmentation significantly improves the result.

Future Work:

We will consider and do analysis on different lighting conditions such as images taken during morning, night etc. Our dataset contains large number of such example but in this project we have considered same for all images. Also, major problem with dataset was it has very imbalance no of instances for different class. We will increase instances for class with has low total number of instances using data augmentation.

References

[1] https://arxiv.org/pdf/1506.01497.pdf
(Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks)
[2] http://podoce.dinf.usherbrooke.ca/challenge/dataset/
(MIO-TCD dataset)
[3] https://arxiv.org/pdf/1409.1556.pdf
(Very Deep Convolutional Networks for Large-Scale Image Recognition)
[4] https://arxiv.org/pdf/1512.03385.pdf
(Deep Residual Learning for Image Recognition)
[5] https://arxiv.org/pdf/1311.2524.pdf
(Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation)
[6] https://arxiv.org/pdf/1504.08083.pdf
(Fast R-CNN)
[7] http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf
(Lecture Slides 11: Detection and Segmentation)
[8] https://medium.com/@smallfishbigsea/faster-r-cnn-explained-864d4fb7e3f8
(Faster R-CNN Explained)