FASTER R-CNN: Towards Real-Time Object Detection with Region Convolutional Neural Network

CONQUERORS

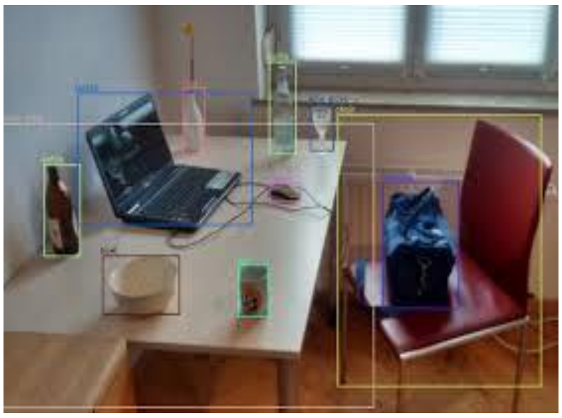
CHAMRUTHA S, KAILASH M S, NIVASHINI S, PARAMESHWARI S, SAI LAKSHMI HARINI C, SUSINDER N

Abstract:

Object detection is a computer vision technique for locating instances of objects in images or videos. Object detection algorithms typically leverage machine learning or deep learning to produce meaningful results.

Image classification involves predicting the class of one object in an image. Object localization refers to identifying the location of one or more objects in an image and drawing a bounding box around their extent. Object detection combines these two tasks and localizes and classifies one or more objects in an image.

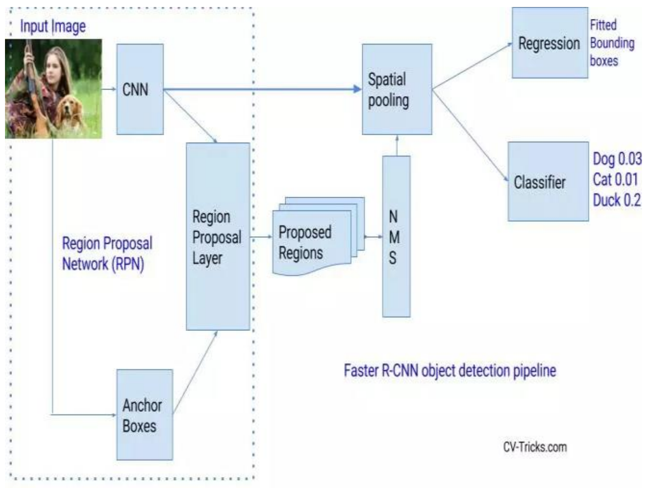
Object Detection: Locate the presence of objects with a bounding box and types or classes of the located objects in an image.
Input: An image with one or more objects, such as a photograph.
Output: One or more bounding boxes (e.g. defined by a point, width, and height), and a class label for each bounding box.



1 Introduction:

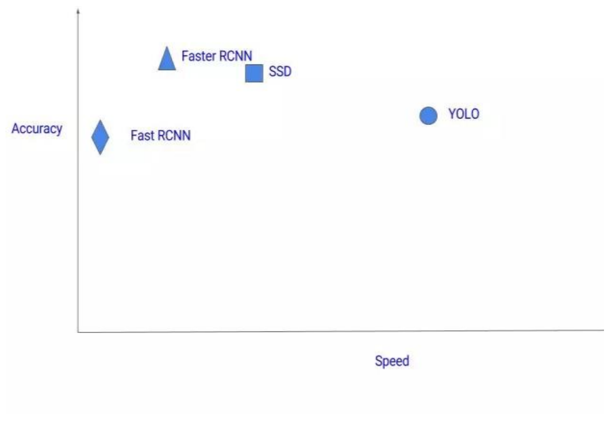
In the process of teaching a machine, one of the most basic things is classification and detection. The machine should be able to identify and distinguish between things like, good or bad, cat or dog etc. It is one of the most fundamental things we do. Thus, we need to train the machine such that it classifies/detects things just like we do or even better than we do.

There are numerous techniques developed for classification and detection. We are about to use **Faster R-CNN**, which is widely used in image processing.

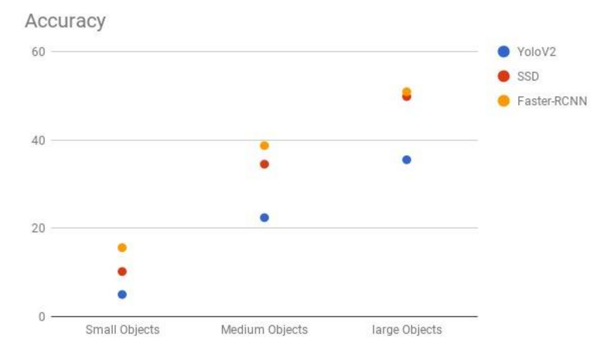


Faster R-CNN is an object detection architecture presented by Ross Girshick, Shaoqing Ren, Kaiming He and Jian Sun in 2015, and is one of the famous object detection architectures that uses Convolutional Neural Networks like YOLO (You Look Only Once) and SSD (Single Shot Detector).

Simple R-CNN takes about 50 seconds to process one image which in real time, has a huge impact in time constraint for the user. To overcome this, many researchers developed Fast R-CNN and then later Faster R-CNN was developed and now it is more popular and efficient than ever.



In terms of accuracy, Faster R-CNN performs better than the Convolution Neural Networks like YOLO and SSD.



2 Related Work

2.1 Region Proposal Based Algorithms

The most representative region proposal based mode is Faster R-CNN, which originates from R-CNN, and fast R-CNN.

R-CNN, Region-based Convolutional Neural Network, or R-CNN, was proposed not long after the emergence of CNN and it greatly improved the detection performance in terms of mean average precision (mAP) compared to models without deep CNNs. The system consists of three parts. The first part generates region proposals using selective search . The second part is feature extraction through CNN, in each proposed region. The third part is classification by SVMs. Although R-CNN is an innovative and effective model, it has many defects: ad hoc training objectives, expensive training both in space and time, and worst, long detection time.

Fast R-CNN, builds on the work of R-CNN and improves training and testing speed while increasing the detection accuracy. The speed-up comes from sharing CNN computation among all region proposals. In Fast R-CNN, an image is put into a CNN to create a convolution feature map first and then a Region of Interest (RoI) pooling layer extracts a feature vector for each region proposal. The feature vectors are fed into fully-connected layers and finally the model produces soft-max class probability estimates and bounding boxes for each detected objects. Fast R-CNN significantly reduces training and testing time, but the region proposals are still made by traditional methods, which costs a long time for pre-processing.

Faster R-CNN. To solve the bottleneck problem of region proposals in Fast R-CNN, Faster R-CNN is proposed, which makes region proposals by neural network instead. It is composed of two modules. The first module is a Region Proposal Network(RPN), which proposes regions for the second module, Fast R-CNN detector, to inspect. In RPN module, a small network slides over the convolution feature map with multiple anchors at each sliding window location. The RPN outputs a bounding box (region proposals) and predicted class as Fast R-CNN module does. A four-step alternating training is adopted to share features for both modules.

2.2 Regression Based Algorithms

YOLO and SSD. You Only Look Once (YOLO) reframe the detection problem as a regression problem, so as Single Shot Multi-Box Detector (SSD), and they do not have region proposals involved. Instead, they are based on pre-defined grid and use a CNN to simultaneously predicts bounding boxes and class confidence, making it extremely fast compared to methods based on region proposals.

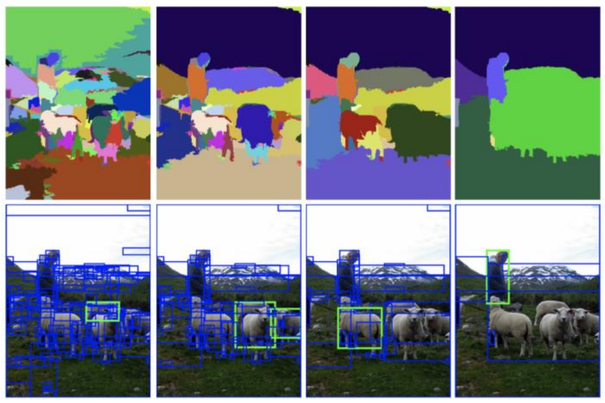
2.3 Iterative Refinement

As reported in Refine-Net, iterative refinement on the proposed regions can significantly improve the mAP of the detection result. Refine-Net is based on ZF Net and achieved close performance to Faster R-CNN, while running ten times faster. Inspired by the paper, we propose iterative refinement based on Fast R-CNN and adding LSTM to it, achieved further improvement on its performance in object detecting task.

3 Project Description:

So, we will be implementing the Faster R-CNN method using Keras for custom data from Google's Open Images Dataset V4 where we will be considering three classes namely "car", "person", and "mobile phone". Every class contains around 1,000 images.

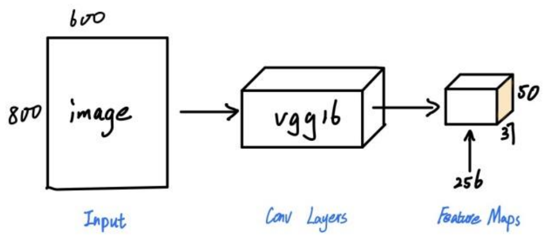
R-CNN is the first step for Faster R-CNN. It uses **search selective** to find out the regions of interests and passes them to a Conv-Net. It tries to find out the areas that might be an object by combining similar pixels and textures into several rectangular boxes. The R-CNN paper uses 2,000 proposed areas (rectangular boxes) from search selective. Then, these 2,000 areas are passed to a pre-trained CNN model. Finally, the outputs (feature maps) are passed to a SVM for classification. The regression between predicted bounding boxes (b-boxes) and ground-truth b-boxes are computed.



Example of search selective

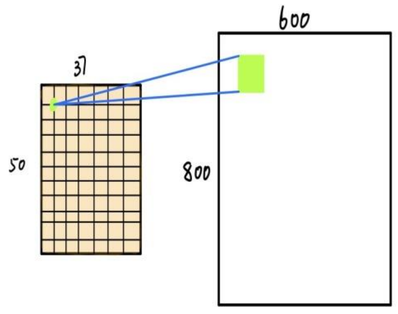
Fast R-CNN moves one step forward. Instead of applying 2,000 times CNN to proposed areas, it only passes the original image to a pre-trained CNN model once. Search selective algorithm is computed based on the output feature map of the previous step. Then, ROI pooling layer is used to ensure the standard and pre-defined output size. These valid outputs are passed to a fully connected layer as inputs. Finally, two output vectors are used to predict the observed object with a soft-max classifier and adapt bounding box localizations with a linear regressor.

Faster R-CNN (frcnn for short) makes further progress than Fast R-CNN. Search selective process is replaced by Region Proposal Network (RPN). As the name revealed, RPN is a network to propose regions. For instance, after getting the output feature map from a pre-trained model (VGG-16), if the input image has 600x800x3 dimensions, the output feature map would be 37x50x256 dimensions.



*First step of **Faster R-CNN***

Each point in 37x50 is considered as an anchor. We need to define specific ratios and sizes for each anchor (1:1, 1:2, 2:1 for three ratios and 128^2, 256^2, 512^2 for three sizes in the original image).



One anchor projected to the original image

Anchor centers throughout the original image

3.1 Difficulties involved in solving the problem:

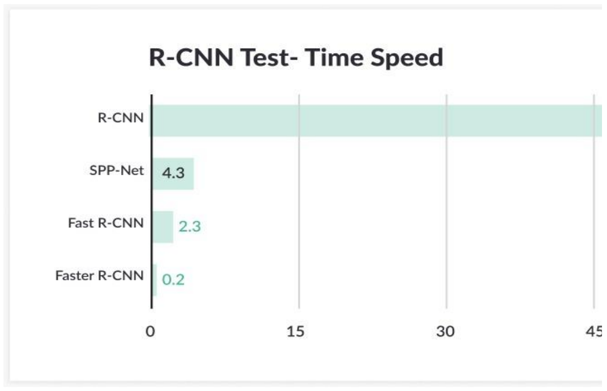
The main problem with R-CNN is that it is very slow to run. It can take 47 seconds to process one image on a standard deep learning machine, making it unusable for real-time image processing scenarios.

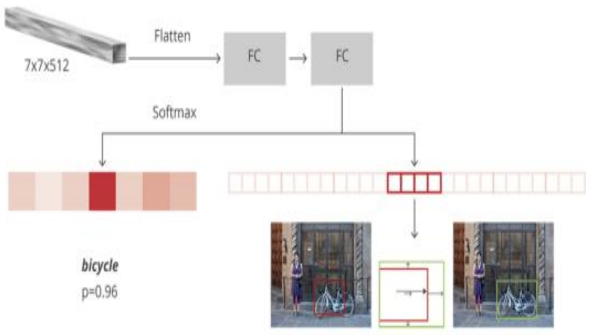
The main thing that slows down R-CNN is the Selective Search mechanism that proposes many possible regions and requires classifying all of them. In addition, the region selection process is not “deep” and there is no learning involved, limiting its accuracy. In 2015 Girshik proposed an improved algorithm called Fast R-CNN, but it still relied on Selective Search, limiting its performance.

Shoqing Ren et. al. proposed an improved algorithm called Faster R-CNN, which does away with Selective Search altogether and lets the network learn the region proposals directly. Faster R-CNN takes the source image and inputs it to a CNN called a Region Proposal Network (RPN). It considers a large number of possible regions, even more than in the original R-CNN algorithm, and uses an efficient deep learning method to predict which regions are most likely to be objects of interest.

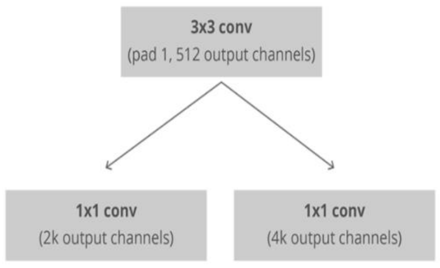
The predicted region proposals are then reshaped using a Region of Interest (RoI) pooling layer. This layer itself is used to classify the images within each region and predict the offset values for the bounding boxes.

The image below shows the huge performance gains that Faster R-CNN achieves compared to the original R-CNN and Fast R-CNN proposed by Girshik's team.





For example, RPN is connected to a Conv layer with 3x3 filters, 1 padding, 512 output channels. The output is connected to two 1x1 convolutional layer for classification and box-regression (Note that the classification here is to determine if the box is an object or not).



Convolutional implementation of an RPN architecture, where k is the number of anchors. (k=9 here)

In this case, every anchor has 3x3 = 9 corresponding boxes in the original image, which means there are 37x50x9 = 16650 boxes in the original image. We just choose 256 of these 16650 boxes as a mini batch which contains 128 foregrounds (positive) and 128 backgrounds (negative). At the same time, non-maximum suppression is applied to make sure there is no overlapping for the proposed regions.

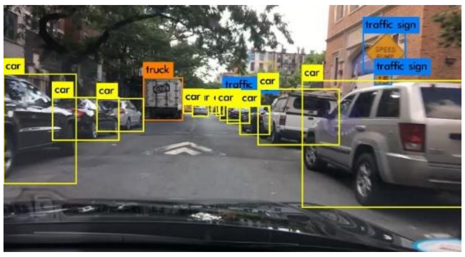
RPN is finished after going through the above steps. Then we go to the second stage of frcnn. Similar to Fast R-CNN, ROI pooling is used for these proposed regions (ROIs). The output is 7x7x512. Then, we flatten this layer with some fully connected layers. The final step is a soft-max function for classification and linear regression to fix the boxes' location.

3.2 Applications of real time object detection:

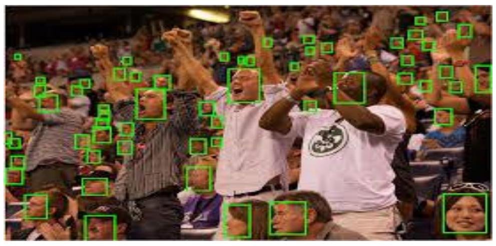
Object detection is being widely used in the industry right now. The use cases of object detection range from personal security to automated vehicle system.

Some of the real time applications are,

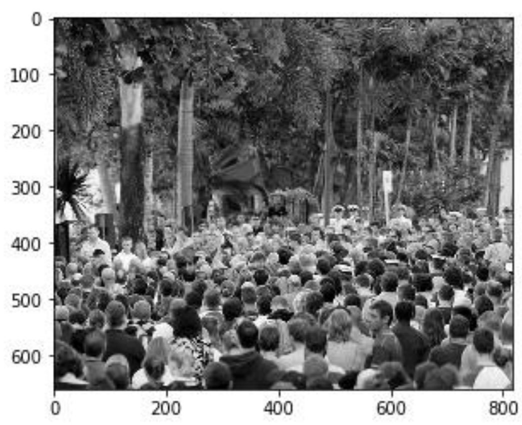
Self-driving cars, which are designed in such a way that they are capable of moving by themselves with little or no human guidance. So, for the car to make it's next step, it must know the location of all objects around it for which object detection is helpful.



Face detection and face recognition, which is one of the most widely used applications.



Object counting, we can use real time object detection for counting number of objects which is very helpful in many ways such as analyzing the performance of a store, estimating the number of people in a crowd, number of people present in a conference hall etc.

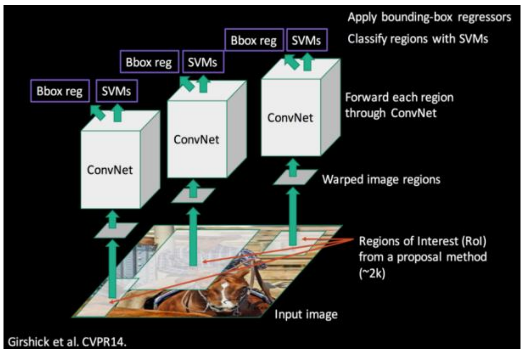


3.3 Approaches used in object detection:

Our object detection system is Faster R-CNN (faster member of R-CNN family), which was published in 2015.

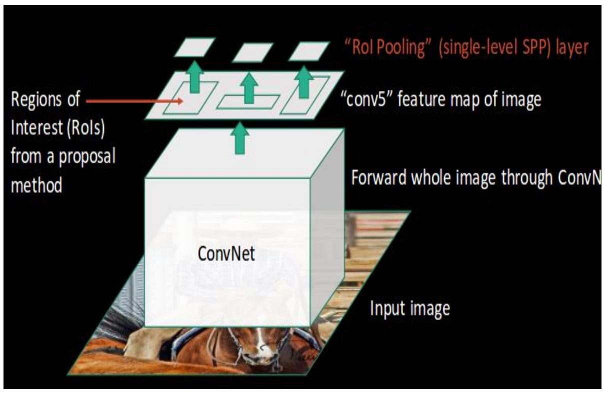
<u>A Brief Overview of the Different R-CNN Algorithms for Object Detection</u>

R-CNN extracts a bunch of regions from the given image using selective search, and then checks if any of these boxes contains an object. We first extract these regions, and for each region, CNN is used to extract specific features. Finally, these features are then used to detect objects. Unfortunately, R-CNN becomes rather slow due to these multiple steps involved in the process.


Apply bounding-box regressors
Classify regions with SVMs
Bbox reg
SVMs
Bbox reg
SVMs
Bbox reg
SVMs
ConvNet
ConvNet
ConvNet
Forward each region through ConvNet
Warped image regions
Regions of Interest (RoI) from a proposal method (~2k)
Input image
Girshick et al. CVPR14.

Fast R-CNN, on the other hand, passes the entire image to ConvNet which generates regions of interest (instead of passing the extracted regions from the image). Also, instead of using three different models (as we saw in R-CNN), it uses a single model which extracts features from the regions, classifies them into different classes, and returns the bounding boxes.

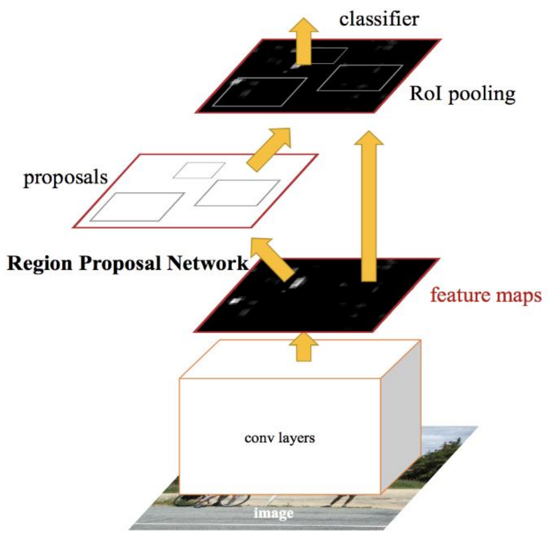
All these steps are done simultaneously, thus making it execute faster as compared to R-CNN. Fast R-CNN is, however, not fast enough when applied on a large dataset as it also uses selective search for extracting the regions.


"RoI Pooling" (single-level SPP) layer
Regions of Interest (RoIs) from a proposal method
"conv5" feature map of image
Forward whole image through ConvN
ConvNet
Input image

Faster R-CNN fixes the problem of selective search by replacing it with Region Proposal Network (RPN). We first extract feature maps from the input image using ConvNet and then pass those maps through a RPN which returns object proposals. Finally, these maps are classified and the bounding boxes are predicted.

I have summarized below the steps followed by a Faster R-CNN algorithm to detect objects in an image:

1. Take an input image and pass it to the ConvNet which returns feature maps for the image
2. Apply Region Proposal Network (RPN) on these feature maps and get object proposals
3. Apply ROI pooling layer to bring down all the proposals to the same size
4. Finally, pass these proposals to a fully connected layer in order to classify and predict the bounding boxes for the image



4 About our project

4.1 Dataset

Our dataset is extracted from Google's Open Images Dataset V4. Three classes for 'Car', 'Person' and 'Mobile Phone' are chosen. Every class contains around 1000 images. The number of bounding boxes for 'Car', 'Mobile Phone' and 'Person' is 2124, 1573 and 2981 respectively.

Note: We took only three of speed of the training. You can take as many as you want.

4.2 Implementation platform

Google's Colab python3 with Tesla K80 GPU acceleration for training.

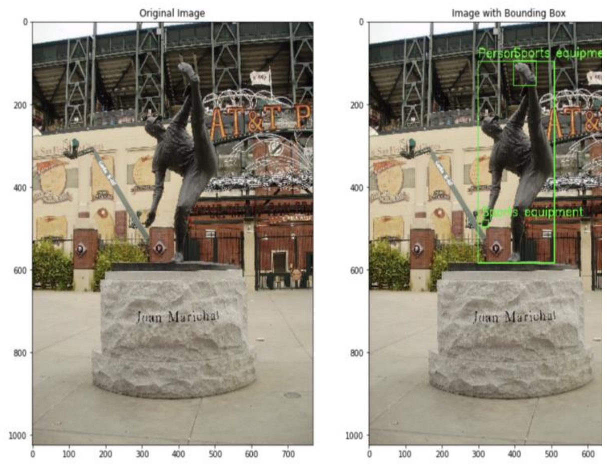
5 Implementation details

5.1 Dataset preparation

The whole dataset of Open Images Dataset V4 which contains 600 classes and we extracted 1,000 images for three classes, 'Person', 'Mobile phone' and 'Car' respectively.

After downloading these 3,000 images, we saved the useful annotation info in a .txt file. Each row has the format like this: file_path, x1, y1, x2, y2, class_name (no space just comma between two values) where filepath is the absolute file path for this image, (x1,y1) and (x2,y2) represent the top left and bottom right real coordinates of the original image, classname is the class name of the current bounding box.

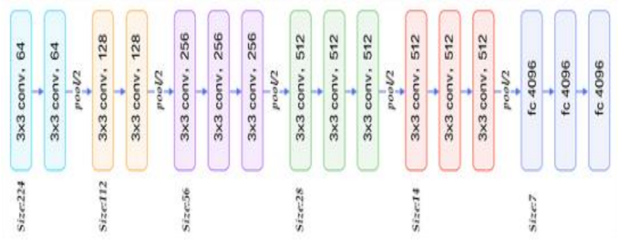
We used 80% images for training and 20% images for testing. The expected number of training images and testing images should be 3x800 -> 2400 and 3x200 -> 600. However, there might be some overlapped images which appear in two or three classes simultaneously. For instance, an image might be a person walking on the street, and there are several cars in the street. So the number of b-boxes for training images is 5388 , and the number of b-boxes for testing images is 1335.



One sample from my extracting data. The left is original image downloaded from given URL and the right is drawn by adding bounding boxes parsed from train-annotations-bbox.csv.

5.2 Faster-RCNN code

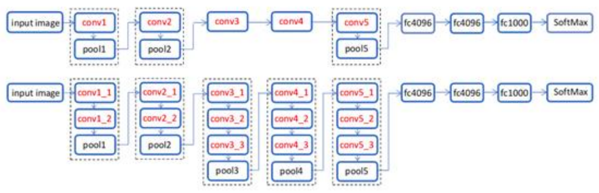
We built a VGG-16 as our base model which is a CNN architecture.



5.2.1 VGG-16 model

VGG is a Convolutional Neural Network architecture, It was proposed by **Karen Simonyan** and **Andrew Zisserman** of **Oxford Robotics Institute** in the year 2014.

With the decades of dedicated research efforts. CNN has made a huge impact after Alex-Net. Here is a rough Architectural comparison of Alex-Net and VGG. For more deep study on VGG than presented here you can visit **Very Deep CNNS for Large-Scale Visual Recognition.**



The Top part is the architecture of Alex-Net and the bottom part is the architecture of VGG-16.

Alex-Net consists of 8 weight including 5 convolutional layers and 3 fully-connected layers, and three max-pooling layers are used following the first, second and fifth convolutional layers**. The first convolutional layer has 96 filters of size 11 X 11 with a stride of 4 pixels and padding with 2 pixels**. The stride and padding of other convolutional layers are set as 1 pixel. The second convolutional layer has 256 filters of size 5 X 5. The third, fourth and fifth convolutional layers have 384, 384 and 256 filters with size of 3 X 3 respectively.

The used VGG 16 is much deeper which consists of 16 weight layers including thirteen convolutional layers with filter size of 3 X 3, and fully-connected layers. The configurations of fully-connected layers in VGG-16 are the same with Alex-Net. The stride and padding of all convolutional layers are fixed to 1 pixel. All convolutional layers are divided into 5 groups and each group is followed by a max-pooling layer.

In contrast to VGG-16, Alex-Net retains more unrelated background information in last convolutional layer, which often disturbs the final prediction. Hence the VGG predicts better than Alex-Net.

Table 2. Size and stride of receptive fields in each layer AlexNet.

layer	**c1**	**p1**	**c2**	**p2**	**c3**	**c4**	**c5**	**p5**
size	11	15	47	55	87	119	151	167
stride	4	8	8	16	16	16	16	32

Table 1. Size and stride of receptive fields in each layer of VGG-16.

layer	**c1_1**	**c1_2**	**p1**	**c2_1**	**c2_2**	**p2**	**c3_1**	**c3_2**	**c3_3**
size	3	5	6	10	14	16	24	32	40
stride	1	1	2	2	2	4	4	4	4
layer	**p3**	**c4_1**	**c4_2**	**c4_3**	**p4**	**c5_1**	**c5_2**	**c5_3**	**p5**
size	44	60	76	92	100	132	164	196	212
stride	8	8	8	8	16	16	16	16	32

5.2.2 Training data

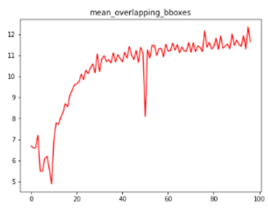
The input data is from annotation.txt file which contains a bunch of images with their bounding boxes information. We used RPN method to create proposed bounding boxes.

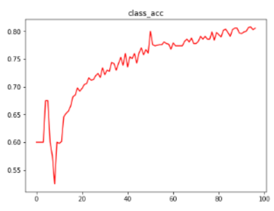
Calculate rpn for each image:

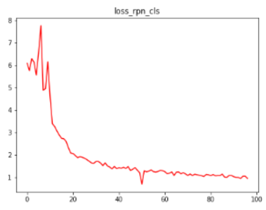
If feature map has shape 18x25=450 and anchor sizes=9, there are 450x9=4050 potential anchors.

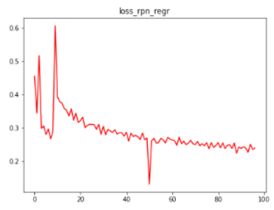
The initial status for each anchor is 'negative'. Then, we set the anchor to positive if the IOU is >0.7. If the IOU is >0.3 and <0.7, it is ambiguous and not included in the objective. One issue is that the RPN has many more negative than positive regions, so we turn off some of the negative regions. We also limit the total number of positive regions and negative regions to 256.

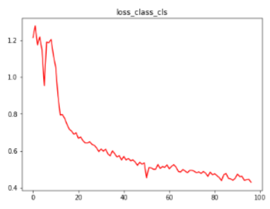
And then Region of Interest(ROI) from the RPN layer is calculated, then passed to **RoIPooling layer and Classifier layer.**

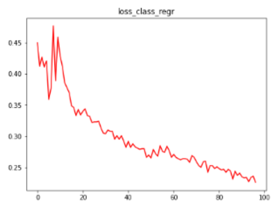


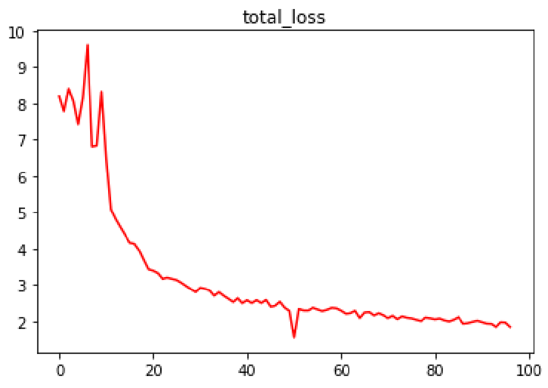






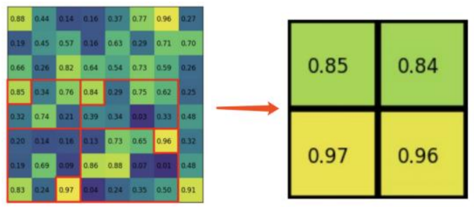






5.2.3 RoI-Pooling layer and Classifier layer

RoI-Pooling layer is the function to process the roi to a specific size output by max pooling. Every input roi is divided into some sub-cells, and we applied max pooling to each sub-cell. The number of sub-cells should be the dimension of the output shape.



Classifier layer is the final layer of the whole model and just behind the RoI-Pooling layer. It's used to predict the class name for each input anchor and the regression of their bounding box.

First, the pooling layer is flattened.
Then, it's followed with two fully connected layer and 0.5 dropout.
Finally, there are two output layers.
1. Soft-max activation function for classifying the class name of the object
2. Linear activation function for b-boxes coordinates regression.

Training images: 2400

Number of epochs is 97 with epoch length of 1000 each.

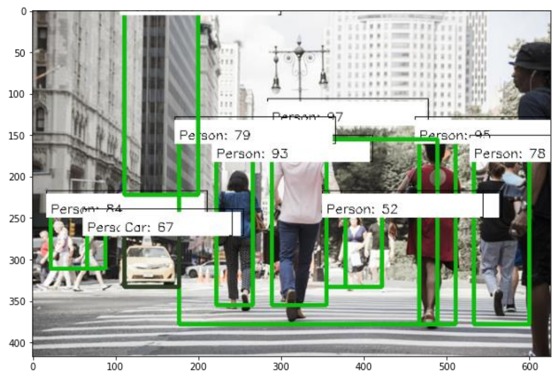
Testing set consists of 800 images.

Images with b-box:

Testing:

After training 97k batches, the mean average precision is 0.722.

On taking a random image from google, we predicted the objects in that image.



6 Result

There are two loss functions which we applied to both the RPN model and Classifier model and as mentioned before, RPN model has two output. One is for classifying whether it's an object and the other is for bounding boxes' coordinates regression.

Performance of Faster RCNN :

Both region proposal generation and objection detection tasks are all done by the same convolutional networks. With such design, object detection is much faster than RCNN and Fast RCNN.

The main advantage of Faster RCNN is that it uses Region Proposal Network as method for generating region proposal.

7 References

1. https://arxiv.org/pdf/1506.01497.pdf

2. https://www.pyimagesearch.com/2018/05/14/a-gentle-guide-to-deep-learning-object-detection/

3.https://www.analyticsvidhya.com/blog/2018/11/implementation-faster-r-cnn-python-object-detection/

4.https://towardsdatascience.com/faster-r-cnn-for-object-detection-a-technical-summary-474c5b857b46