Topics in AI:
Convolution Neural Network

by

Prof. Mashhour Solh

Pedestrian Detection

Milestone 3

Group B :

Vidish Naik
Charulata Lodha
Rakesh Nagaraju
Joel Alvares

Contents

Idea of the Project

- After Literature survey, we chose to move forward with Repulsion Loss as our project.
- We saw that Repulsion Loss uses Single Shot Detector for Object detection.
- We found that Faster RCNN has a better accuracy and we want to try to see the effects of repulsion loss as its loss function.
- Our aim is to improve the accuracy of pedestrian detection using Faster RCNN and repulsion loss.

Recap: Repulsion Loss

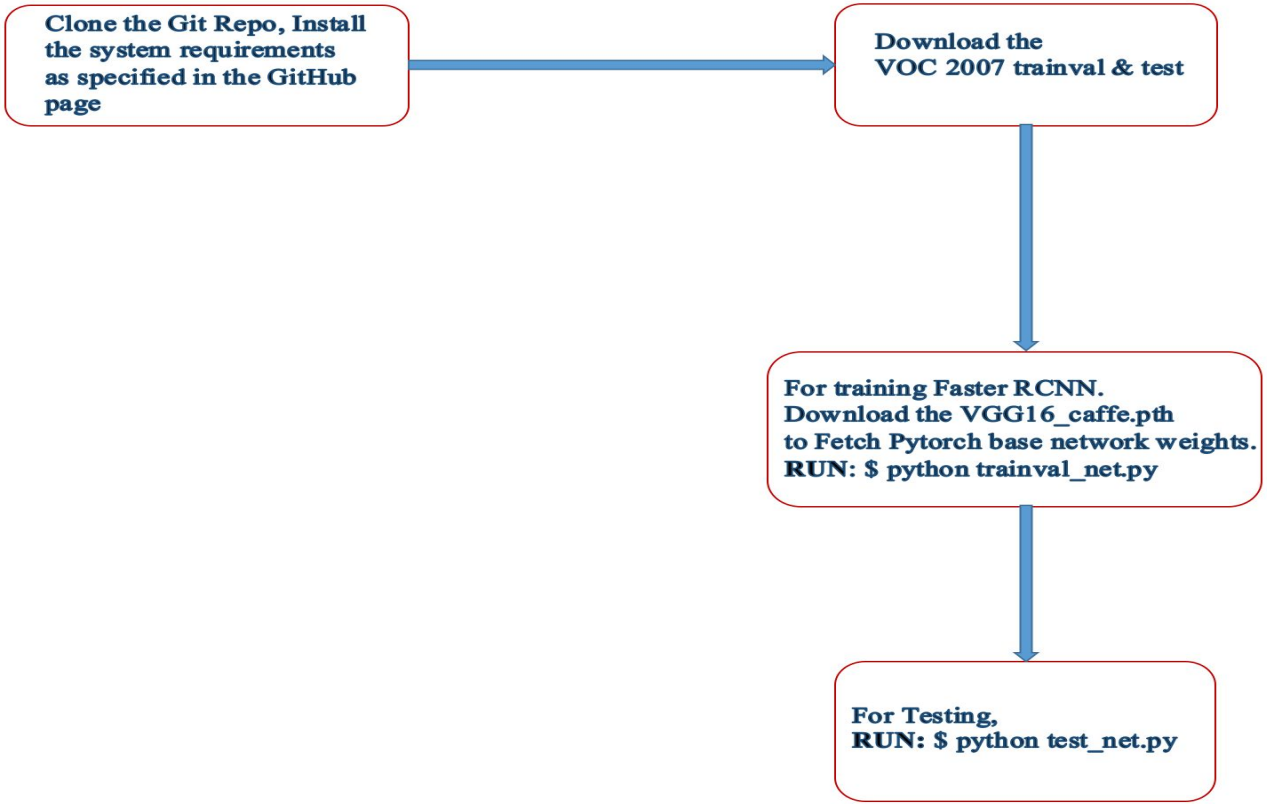
- The idea of repulsion comes from the attraction and repulsion properties of magnets.
- The predicted bounding boxes and ground truth act as magnets attracting each other.
- All of the predicted boxes together act as magnets that are trying to repel each other.
- Repulsion Loss is given by the formula

$$L = L_{Attr} + \alpha * L_{RepGT} + \beta * L_{RepBox}$$

➔ L_{Attr} is the attraction term.

➔ L_{RepGT} and L_{RepBox} are the repulsion terms.

Flow of Execution



Tech Specs(To be edited) :

- Technical Specs :
  - GPU : Tesla K80
  - PyTorch : 0.3.1
  - Torch Vision : 0.2.0
- Data Set :
  - VOC 2007
- Weights:
  - VGG16_caffe.pth PyTorch base network weights.
- Total classes : 21
  - Person, Bird, Bicycle, Horse, etc.

XML Annotations

```
<annotation>
        <folder>VOC2012</folder>
        <filename>2007_001558.jpg</filename>
        <source>
                <database>The VOC2007 Database</database>
                <annotation>PASCAL VOC2007</annotation>
                <image>flickr</image>
        </source>
        <size>
                <width>500</width>
                <height>375</height>
                <depth>3</depth>
        </size>
        <segmented>0</segmented>
        <object>
                <name>person</name>
                <pose>Unspecified</pose>
                <truncated>1</truncated>
                <difficult>0</difficult>
                <part>
                        <name>head</name>
                        <bndbox>
                                <xmin>77</xmin>
                                <ymin>63</ymin>
                                <xmax>206</xmax>
                                <ymax>199</ymax>
                        </bndbox>
                </part>
                <part>
                        <name>hand</name>
                        <bndbox>
                                <xmin>178</xmin>
                                <ymin>136</ymin>
                                <xmax>215</xmax>
                                <ymax>197</ymax>
                        </bndbox>
                </part>
                <part>
                        <name>hand</name>
                        <bndbox>
                                <xmin>65</xmin>
                                <ymin>286</ymin>
                                <xmax>138</xmax>
                                <ymax>328</ymax>
                        </bndbox>
                </part>
                <bndbox>
                        <xmin>1</xmin>
                        <ymin>56</ymin>
                        <xmax>286</xmax>
                        <ymax>329</ymax>
                </bndbox>
        </object>
```

Training

- The model currently supports only VOC 2007 dataset.
- Train VOC dataset: 9,963 images containing 24,640 annotated objects.
- To train Faster RCNN using the train script simply specify the parameters listed in **trainval_net.py** as a flag or manually change them.
- Before starting the training phase, parameters like batch size, weight decay, momentum, gamma and learning rate are set otherwise default values are used.

Utility Files

- **prior.py:**
  It computes the **coordinates of the prior box** in the form of center-offset for each source feature map by calculating and summing different means and returns an output with x, y coordinates for the top left and bottom right corner of the prior box.

- **detection.py:**
  It decodes **location predictions** and **applies non-maximum suppression(NMS)** based on threshold to a top k number of output predictions for both confidence scores and locations.

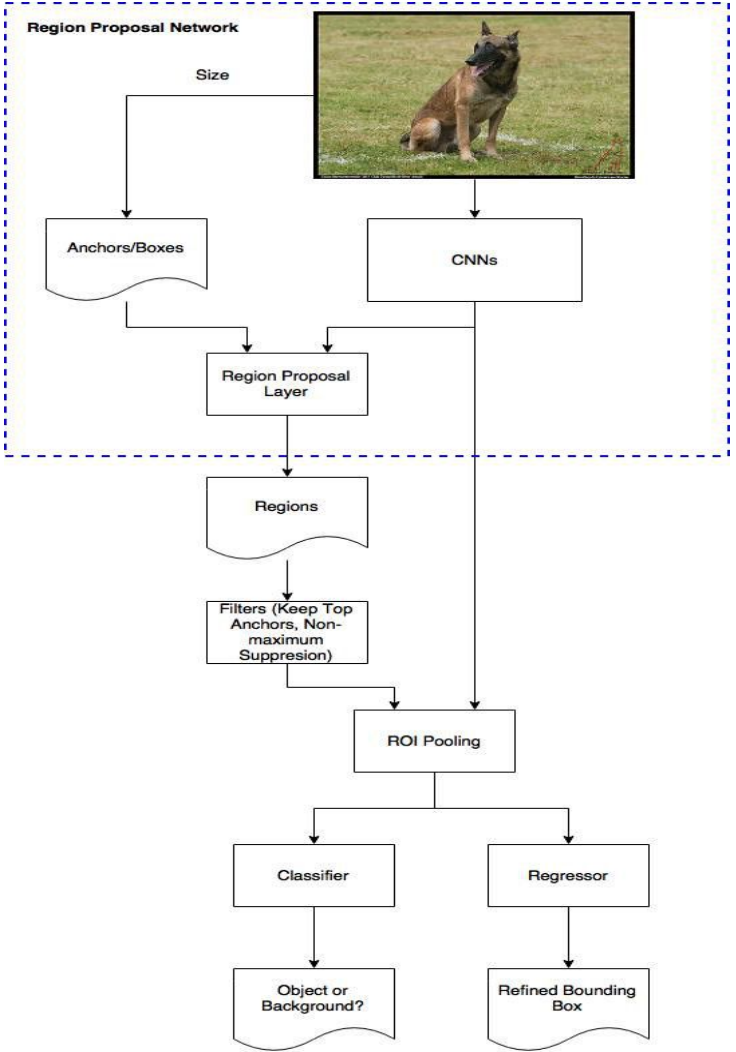
- **repulsion_loss.py:**
  It calculates the repulsion loss as **the sum over IoG values** using location data, ground truth data, and prior data.

Faster RCNN

- Faster R-CNN is a sliding window-based approach
- It has a dedicated region proposal network followed by max-pooling and a classifier to classify the object in the proposed region.
- Stochastic Gradient Descent (SGD) is adopted to optimize the network with the weight decay parameter as 5e-4, momentum as 0.9, and learning-rate as 1e-3.

Faster R-CNN

- Faster R-CNN is a sliding window-based approach
- It has a dedicated region proposal network followed by max-pooling and a classifier to classify the object in the proposed region.



Training

- The data loader then combines a dataset, and provides an iterable over the given dataset.
- In each batch iteration, the initial learning rate is decayed at specific step intervals.
- A forward pass computes the loss followed by backpropagation to update the weights on the batch of images.
- The multi-box loss function calculates Confidence Target Indices, by matching ground truth boxes with Prior boxes, which have jaccard index greater than the threshold parameter (default threshold = 0.5).

Training Continued...

- Hard negative mining is done to filter out the excessive number of negative examples associated with a large number of default bounding boxes. (default negative: positive ratio 3:1).
- This program returns three-loss values:

  Localization loss (how far away the network's predicted bounding boxes are from the ground truth),

  Repulsion loss (sum over IoU), and

  Confidence loss (how confident the network is).
- Final loss is a sum of Localization loss, Repulsion loss, and Confidence loss, each of which are returned by the multi-box loss function.
- This forward Pass and backpropagation together comprise of the training phase and we train the model for about **2500 epochs**.
- Once, training is done, we now have to test and evaluate the model's performance.

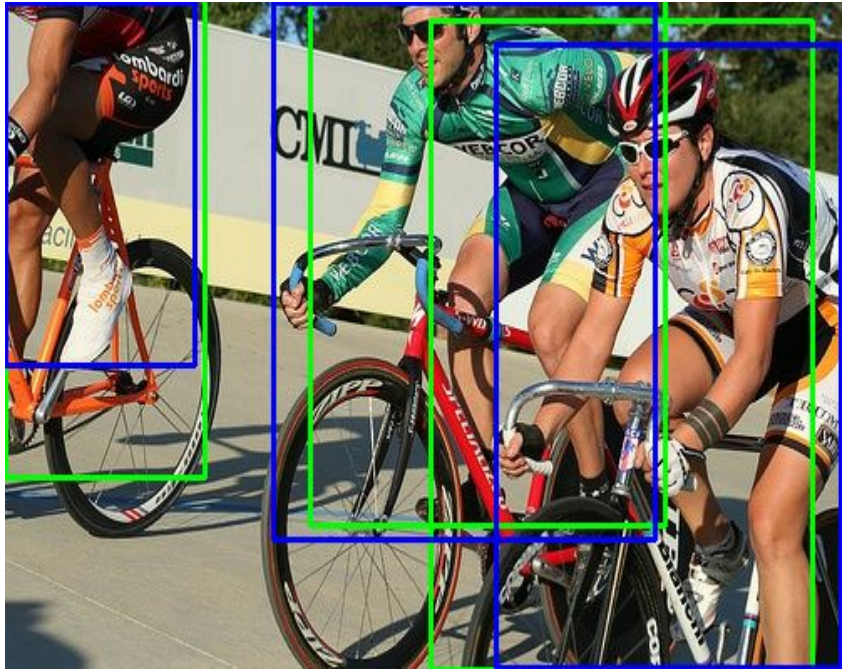
Testing and Evaluation:

- After completing the training, the next step is to evaluate the model's performance using the test dataset.
- "**test_net.py**" is run on the test dataset and predictions(x1,y1,x2,y2) are generated during forward pass and results are stored in a text file.
- This program also generates bounding boxes over its predictions.
- Overlap of predicted bounding boxes and ground truth boxes is then calculated using the maximum of the x, y coordinates of the top left corner and minimum of x, y coordinates of bottom right corner of either ground truth box or predicted box, giving us the intersection area.

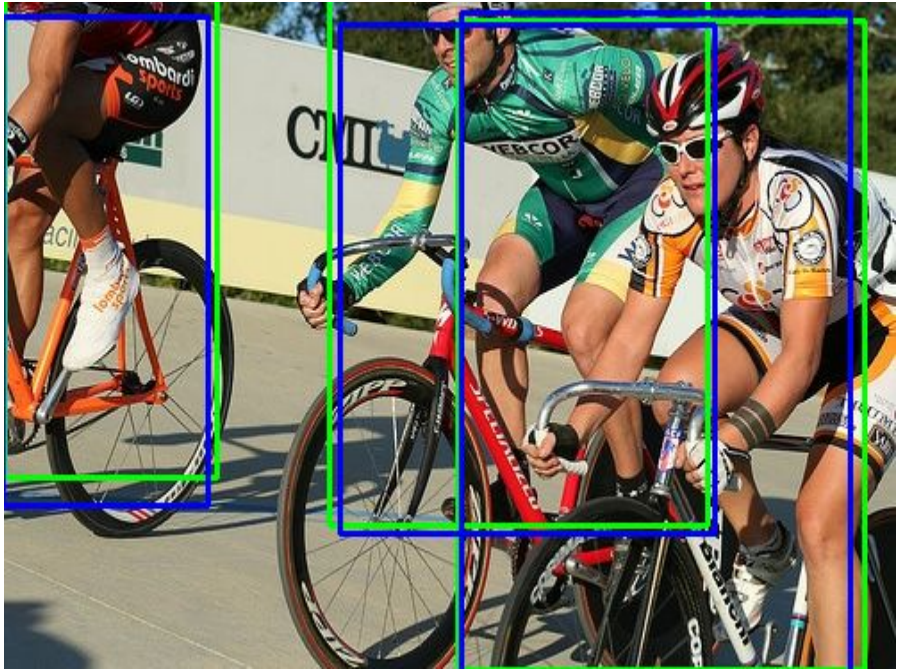
Evaluation:

- The area for union is calculated to get a value which should be greater than threshold value, 0.5 in this case, to qualify as a successful prediction.
- Depending on the prediction, we classify it in true positive(TP) or false positive(FP) and we calculate precision and recall as shown in the next section.

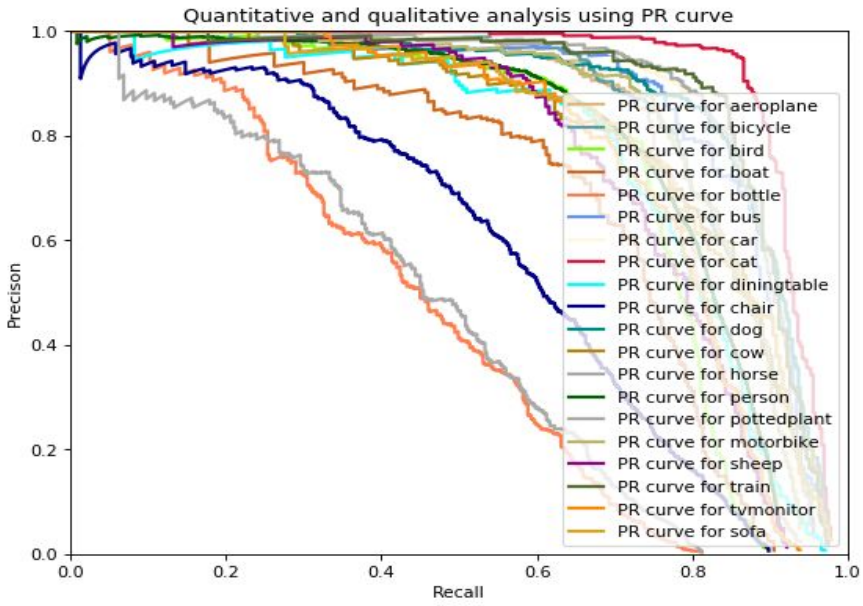
Mean Average Precision Comparison	**mAP**	**batch size**
Repulsion loss with SSD	74.58	32
Repulsion loss with Faster RCNN	79.8	4



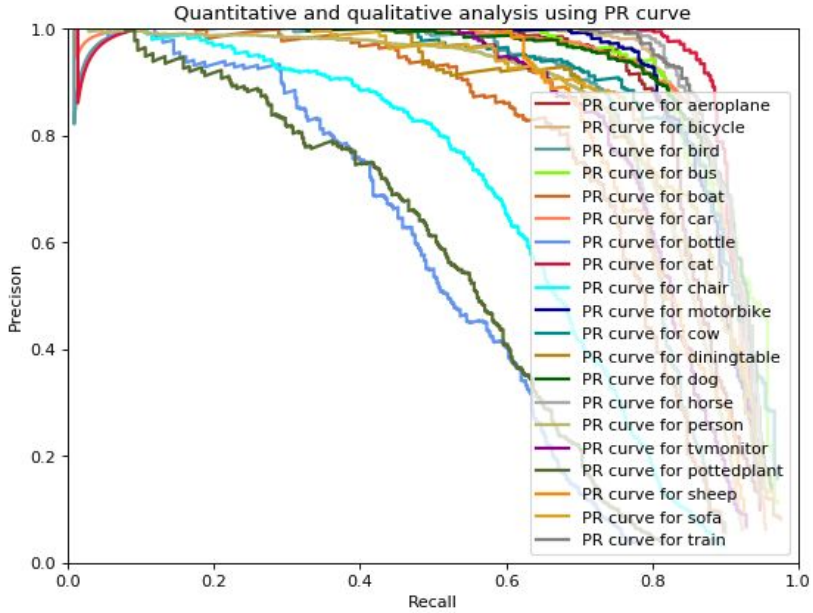
Ground-Truth (Green) vs Predicted Bounding Box (Blue) using SSD with repulsion loss



Ground Truth (Green) vs Predicted Bounding Box (Blue) using Faster R-CNN with repulsion loss



RepLoss+SSD



RepLoss+Faster RCNN

High scores for low false positive rate and low false negative rate show that the classifier is returning accurate results (high precision), as well as returning a majority of all positive results (high recall).

Demo

- Setup
- Faster RCNN + Repulsion Loss : Test Image with Occlusion

Discussion and Future Work

- Our implementation of Repulsion Loss + Fast RCNN performs well, but we think there is still scope for improvement. We can utilize other regression-based loss functions such as soft margin loss and hinge embedding loss, different types of optimizers to update the weights such as Adagrad, AdaDelta, RMSProp, NAG, Adam and various types of decaying learning rates such as time-based decay and step decay and check for improvement inaccuracy of the model.
- Quantization can be utilized to significantly reduce the bandwidth and storage by representing the weights and activations from 32-bit floating-point values to 8-bit integer values. Even though the space required to store the model is reduced ~4 fold but the accuracy would be reduced by a mere 5-8 %. The quantized model will be faster as integer compute is faster than float compute.

Github link

https://github.com/VidishNaik/RepulsionLoss

References

- Repulsion Loss: Detecting Pedestrians in a Crowd (link: https://arxiv.org/pdf/1711.07752v2.pdf)
- Part level CNN for Pedestrian detection (https://arxiv.org/pdf/1810.00689v1.pdf)
- Object as Points (link: https://arxiv.org/pdf/1904.07850v2.pdf)
- Center and Scale Prediction (link: https://arxiv.org/pdf/1904.02948v2.pdf)
- https://medium.com/overture-ai/part-5-object-detection-when-image-classification-just-doesnt-cut-it-b4072fb1a03d
  https://en.wikipedia.org/wiki/Object_detection
- https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1
- https://medium.com/datadriveninvestor/overview-of-different-optimizers-for-neural-networks-e0ed119440c3
- https://medium.com/@jonathan_hui/what-do-we-learn-from-single-shot-object-detectors-ssd-yolo-fpn-focal-loss-3888677c5f4d
- https://medium.com/@jonathan_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359
- https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52
- Faster R-CNN (link: https://arxiv.org/abs/1506.01497)