Topics in Artificial Intelligence
Convolution Neural Network
CS-256 - Section 2

Prof. Mashhour Solh

Milestone 2.5

Group B

Joel Alvares	-	013714415
Rakesh Nagaraju	-	014279304
Charulata Lodha	-	014521182
Vidish Naik	-	014515358

Fall 2019

I. Summarization of the Project

Through the literature review, we analyzed the most dominant aspects for improvement of Pedestrian Detection namely Centre-Point Estimation, Bounding Box Alignment, Center and Scale Prediction, and Repulsion Loss. We determined that Repulsion loss, a bounding box regression technique, could be used to improve the localization accuracy of the predicted box as shown in **Figure 1**. This paper aims to minimize the overlap of the predicted box with other ground-truths as shown in **Figure 1**, using a loss function:

$$\mathbf{L = L_{Attr} + \alpha * L_{Rep\ GT} + \beta * L_{Rep\ Box}}$$

Attraction Term (L_{Attr}): It uses Smooth_{L1} to narrow the gap between the predicted box and ground-truth.
Repulsion Term (L_{RepGT}): It penalizes the prediction box for shifting to other ground truth objects.
The term is meant to keep the proposal away from the neighboring non-truth ground-truth objects.
Repulsion Term (L_{RepBox}): It tries to keep the predictions of all ground truths separated from one another.

$$L_{\text{Attr}} = \frac{\sum_{P \in \mathcal{P}_+} \text{Smooth}_{L1}(B^P, G^P_{Attr})}{|\mathcal{P}_+|}. \qquad L_{\text{RepGT}} = \frac{\sum_{P \in \mathcal{P}_+} \text{Smooth}_{ln}\left(IoG(B^P, G^P_{Rep})\right)}{|\mathcal{P}_+|}.$$

$$L_{\text{RepBox}} = \frac{\sum_{i \neq j} \text{Smooth}_{ln}\left(IoU(B^{P_i}, B^{P_j})\right)}{\sum_{i \neq j} \mathbb{1}[IoU(B^{P_i}, B^{P_j}) > 0] + \epsilon},$$

Furthermore, IoG (Intersection over Ground truth) is adopted in this approach because the bounding box regression might enlarge the bounding box size, to increase the denominator in IoU thereby reducing the loss. This approach will aid in detecting occluded pedestrians, in an image as there is a reduction in overlap of bounding boxes due to proper localization and subsequently improves the accuracy, as they do not get suppressed during NMS.

Our new approach aims at using Faster RCNN model along with Repulsion Loss to improve the accuracy of detection of pedestrians on VOC 2007 dataset.

II. Chosen Approach & Implementation Details

The chosen approach 'Repulsion Loss' aims at decreasing the occlusion effect in pedestrian detection. In this project, we executed the "Repulsion Loss: Detecting Pedestrians in a Crowd" on Google Colab for 10k images of the VOC2007 dataset using an Nvidia Tesla K80 GPU, PyTorch 0.3.1 and TorchVision 0.2.0. We executed the code provided in the paper to understand the overall flow as shown in the flowchart in **Figure 2**.

Training:

The model currently supports only VOC and COCO dataset which should be passed as an argument. Before the training phase, training parameters like batch size, weight decay, momentum, gamma and learning rate are set otherwise default values are used. Training starts by loading the total number of classes including a class for the background to initialize the SSD.

Single Shot Multibox Detector architecture is composed of a base VGG network followed by the added multi-box convolution layer. While creating the SSD network, we compute prior-box coordinates in the center-offset form for each source feature map. 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.

The data loader then combines a dataset, a sampler, 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 Priorboxes, which have jaccard index greater than the threshold parameter (default threshold = 0.5). The localization target is calculated next by encoding(method) variance into offsets of ground truth boxes and their matched priorboxes. Then, 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, Repulsion loss, and Confidence loss. Final loss is a sum of Localization loss, Repulsion loss, and Confidence loss, each of which are returned by the multi-box loss function.

Evaluation:

After completing the training, the next step is to evaluate the model's performance using the test dataset. The model is run on the test dataset and predictions are generated. The evaluation file eval.py 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 of either ground truth box or predicted box and minimum of x, y coordinates of bottom right corner of either boxes giving us the intersection area. The area for union is calculated similarly and then divide these values 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.

Utility Modules:

- **ssd.py:**
  The network is composed of a base VGG network followed by the added multi-box convolution layers. Each multi-box layer branches into 1) conv2d for class confidence scores, 2) conv2d for localization predictions, 3) associated prior box layer to produce default bounding and 4) boxes specific to the layer's feature map size.
  It takes input as a batch of images as arguments and returns values depending on the phase. In the case of test, a variable(tensor) of output class prediction, confidence score, and corresponding coordinates of prediction for each object is returned.
  And while in training, a list of outputs from confidence layers, shape [batch*num_priors, num_classes], localization layers, shape [batch, num_priors*4], priorbox layers, shape [2, num_priors*4] is returned.

- **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:**
  This is the final layer of SSD and is called during testing. It decodes location predictions and applies non-maximum suppression(NMS) based on configuration scores and 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 R-CNN:

We have executed the base code of Faster R-CNN on Google Colab. There were two pre-trained models VGG16 and ResNet101. The models use PyTorch versions 1.0+ and support the latest version of PyTorch. We evaluated the detection performance of a pre-trained VGG16 model using PyTorch's pre-trained model feature on the Pascal VOC2007 test set to obtain the final detection results as shown in **Figure 4**. As next steps we plan to integrate repulsion loss with Faster R-CNN and compare its performance with SSD as an object detector.

III. Qualitative and Quantitative Results

In **Figure 3** the ground truth is the green boxes while the blue boxes are the predictions. As seen in the figure, the IoU is > 0.5 for the predictions as there is a large overlap.

PR Curves (Precision recall curves): Precision measures the ratio of true positives over all positive predictions which includes false positives. Recall is the same as true positive rate which is the ratio of the number of positives predicted accurately.

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

where TP = True Positive, FP = False Positive and FN = False Negative.

The metrics required to calculate precision and recall such as True Positives, False Positives and False Negatives are calculated using Intersection over Union (IoU) which is given by:

$$IoU = \frac{Area\ of\ overlap}{Area\ of\ Union}$$

The metrics used to calculate precision and accuracy are as follows:

- A True positive would have the correct classification with an IoU > 0.5.
- A False positive could be possible in two cases where either would have the correct classification with an IoU < 0.5 or a duplicate bounding box for an object already classified.
- A False negative prediction would have an IoU > 0.5 but the wrong classification.

In **Figure 5** we have calculated the precision and recall for the 21 classes as discussed above to generate the PR curve.

mAP (Mean average prediction): mAP is used to predict the accuracy of an object detector. It is the average of the AP calculated for all the classes. The AP is calculated by taking the area under the PR curve which is done by segmenting the curve evenly into 11 parts {0, 0.1, 0.2, ….., 1} which is given by:

$$AP = \frac{1}{11} \sum_{r \in \{0,0.1,0.2,\ldots,1\}} p\_interp(r)$$

where linear interpolation is used to calculate p_interp by selecting the maximum precision measured for a given recall value.

mAP is the average of the AP calculated for all the classes.

- mAP for repulsion loss with SSD = 74.58 (with batch size 32)
- mAP over evaluation set for Faster RCNN = 69.2 (with batch size 24)

IV. Next Steps Towards Final Project

Our current implementation uses SSD (single-shot object detector) along with repulsion loss for prediction with real-time processing speed. However, it faces issues while detecting objects that are too close even after applying repulsion loss and at times fails to detect objects that are too small.

As the next steps, we plan on using Faster R-CNN as the object detector using repulsion loss and compare the accuracy with that of the SSD based network. Faster R-CNN is a sliding window-based approach which has a dedicated region proposal network followed by max-pooling and a classifier to classify the object in the proposed region. Faster R-CNN with repulsion loss should overcome the issues with SSD although it will require more compute resources.

Other than this we also plan on utilizing 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.

V. References

1. RepulsionLoss: Detecting Pedestrians in a Crowd (link: https://arxiv.org/pdf/1711.07752v2.pdf)
2. Faster R-CNN (link: https://arxiv.org/abs/1506.01497)
3. https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1
4. https://medium.com/datadriveninvestor/overview-of-different-optimizers-for-neural-networks-e0ed119440c3
5. https://medium.com/@jonathan_hui/what-do-we-learn-from-single-shot-object-detectors-ssd-yolo-fpn-focal-loss-3888677c5f4d
6. https://medium.com/@jonathan_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359
7. https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52

Appendix

Figure 1: Repulsion Loss

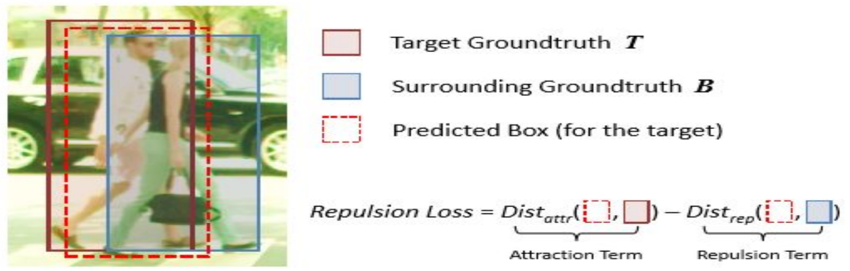
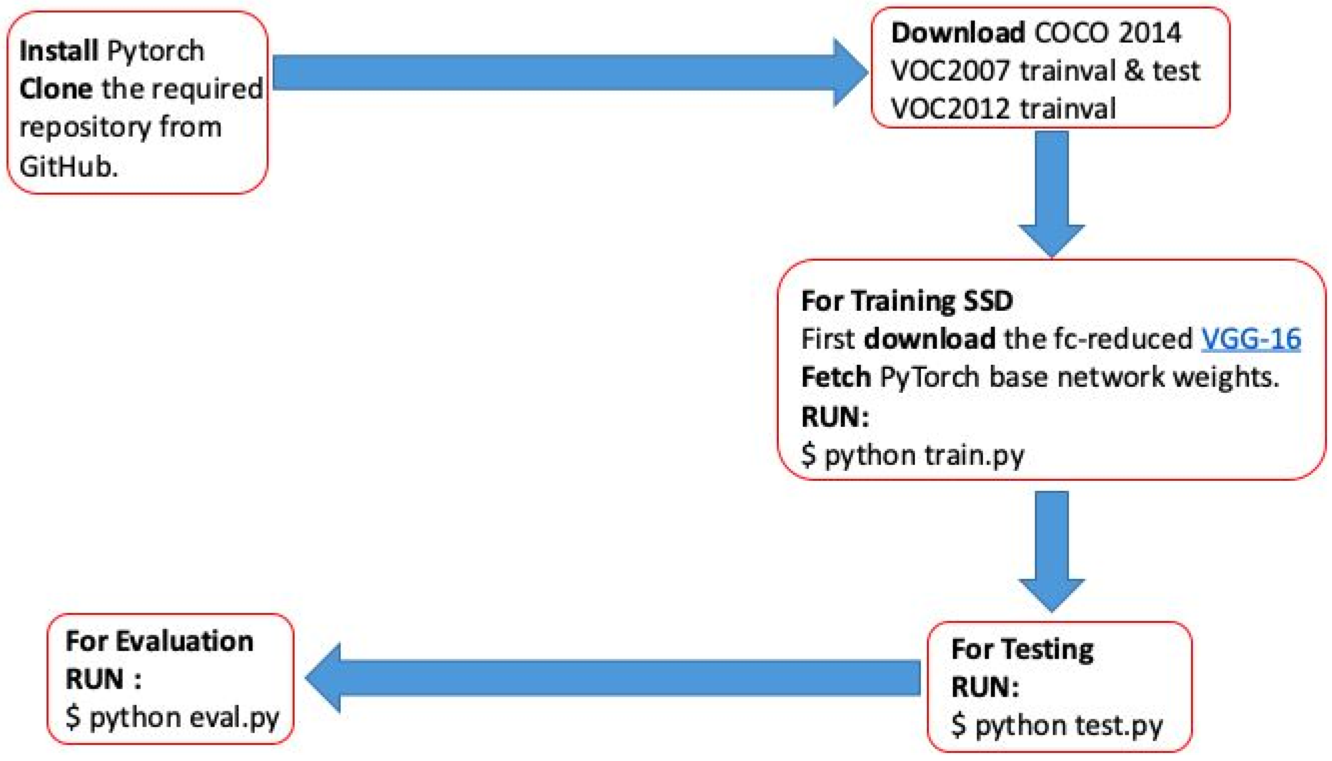

🟥 Target Groundtruth *T*
🟦 Surrounding Groundtruth *B*
⬚ Predicted Box (for the target)
Repulsion Loss = $\underbrace{Dist_{attr}(⬚, 🟥)}_{\text{Attraction Term}} - \underbrace{Dist_{rep}(⬚, 🟦)}_{\text{Repulsion Term}}$

Figure 1. Illustration of our proposed repulsion loss. The repulsion loss consists of two parts: the attraction term to narrow the gap between a proposal and its designated target, as well as the repulsion term to distance it from the surrounding non-target objects.

Figure 2: Flowchart for Implementation details.



Install Pytorch
Clone the required repository from GitHub.

Download COCO 2014
VOC2007 trainval & test
VOC2012 trainval

For Training SSD
First **download** the fc-reduced VGG-16
Fetch PyTorch base network weights.
RUN:

```
$ python train.py
```

For Testing
RUN:

```
$ python test.py
```

For Evaluation
RUN :

```
$ python eval.py
```

Figure 3: Ground Truth (Green) vs Predicted Bounding Box (Blue) using SSD

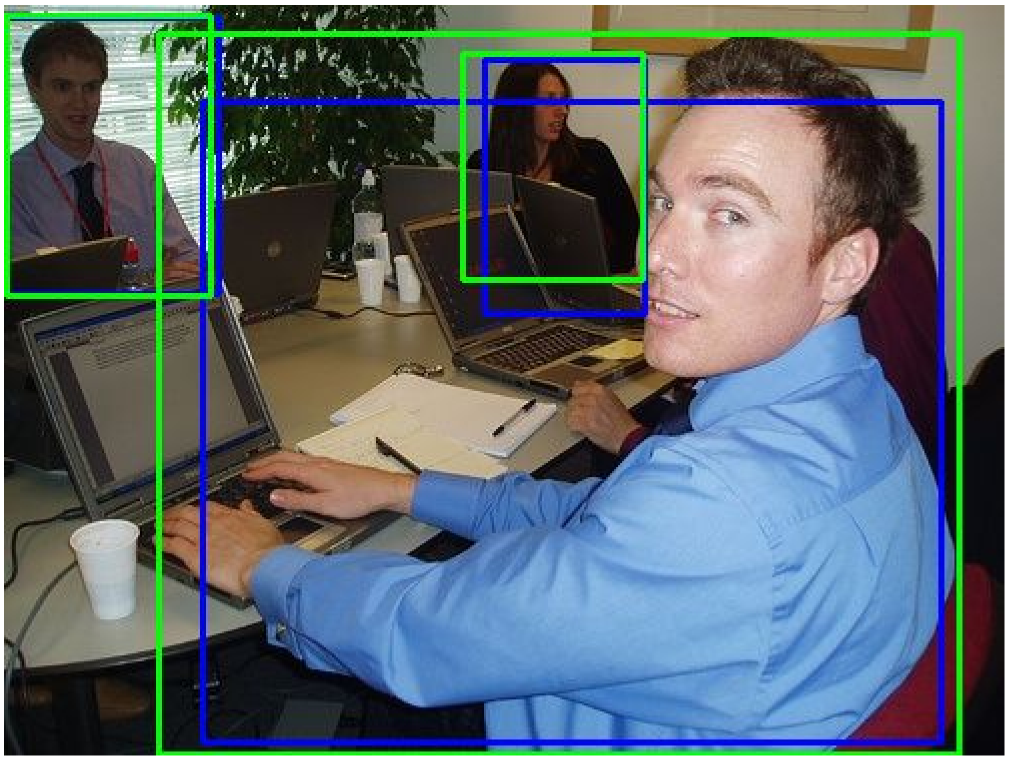


Figure 4: Predictions with Faster RCNN

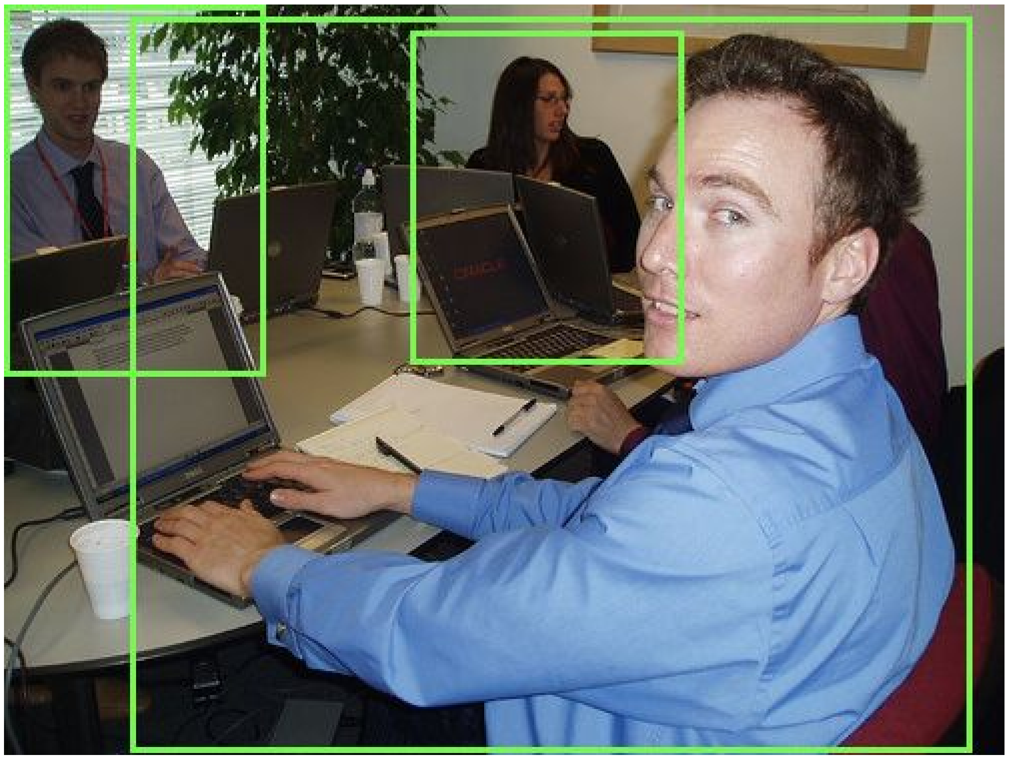
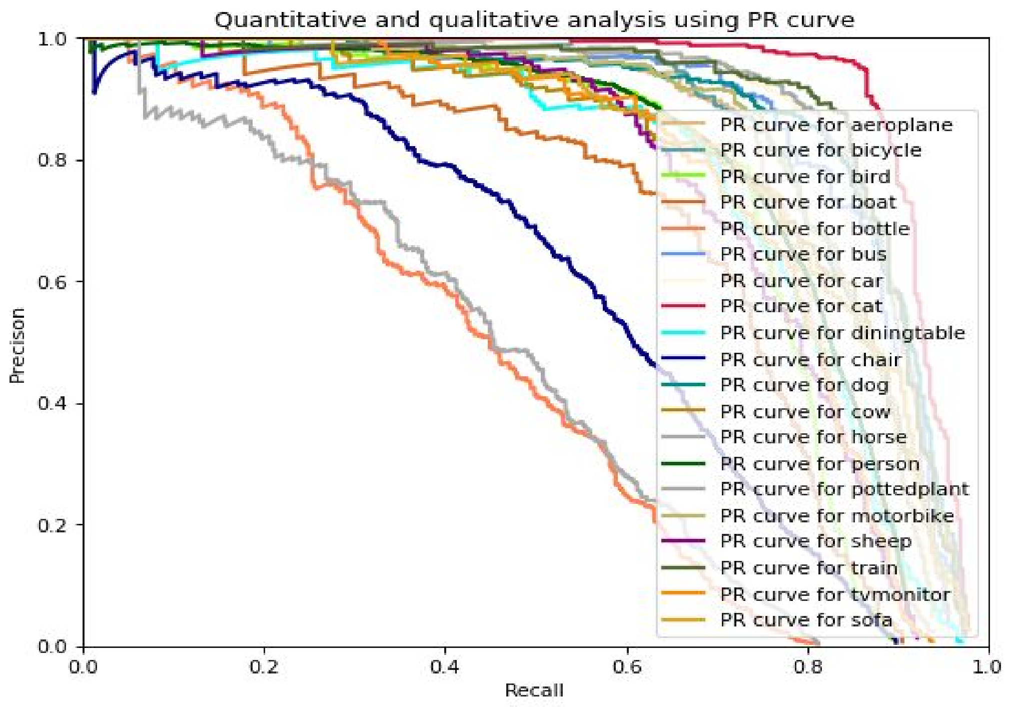


Figure 5: a


Quantitative and qualitative analysis using PR curve
Precison
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
PR curve for aeroplane
PR curve for bicycle
PR curve for bird
PR curve for boat
PR curve for bottle
PR curve for bus
PR curve for car
PR curve for cat
PR curve for diningtable
PR curve for chair
PR curve for dog
PR curve for cow
PR curve for horse
PR curve for person
PR curve for pottedplant
PR curve for motorbike
PR curve for sheep
PR curve for train
PR curve for tvmonitor
PR curve for sofa