COL780: Computer Vision

Assignment 3: Pedestrian Detection

We present here the details for each of the methods we implemented for Pedestrian Detection. We also provide the details on results obtained.

1. Pre-trained HOG

Method: This part of the assignment uses a pre-trained HoG detector, specifically, it uses the HOGDescriptor function from OpenCV. Using this, we get a set of bounding boxes, but a lot of them overlap with each other. To resolve this problem we perform non-maximal suppression on the set of boxes obtained and improve our result.

Results:

Average Precision = 0.056

Average Recall @ 1 detection per image = 0.063

Average Recall @ 10 detections per image = 0.121

2. Custom HOG

Method: For custom HOG, we have first generated a labelled set of training data - positive and negative samples. Positive samples are directly taken from the ground-truth values while negative samples are generated by considering a random portion of images that do not overlap with the positive samples. Then we have extracted the HOG features in the training samples using the scikit-image HoG feature extractor with 9 orientation bins that the gradients of the pixels of each cell will split up in the histogram and 8 pixels of each row and column per cell over each gradient the histogram is computed. Thereafter, we have trained a linear SVM classifier on the extracted features. Then we have implemented the sliding window algorithm (with step size = 10 and window size = (100,264)) combined with the Gaussian image pyramid (downscale factor=1.5) to get all the subregions of the frame.

For each subregion, we have extracted its HoG features and then used the SVM classifier to detect pedestrians in the subregion. After scanning all subregions of the image using the sliding window, we have applied non-maximum suppression to remove redundant and overlapping bounding boxes. We experimented our model with different window sizes but observed the best accuracy for window size (100, 264). It is worth mentioning that while the step size of 10 does make our algorithm slow, it significantly improves the accuracy at the same time. Other values of step size (eg. 15, 20) we experimented with, also managed to give a decent accuracy without significantly slowing down the algorithm.

The trained SVM model can be found here.

Results:

Average Precision = 0.1720

Average Recall @ 1 detection per image = 0.125

Average Recall @ 10 detections per image = 0.2657

3. Faster RCNN

Method: This part of the assignment uses a pre-trained faster RCNN model to perform pedestrian detection. The model used has been pre-trained on the COCO dataset to detect multiple object categories, but we are specifically concerned with the category "person" for the task of pedestrian detection. We have designed our custom dataset class to work with the provided dataset. We have used the default data loader from PyTorch but with a custom collate function to accommodate images with varying sizes in the data without scaling them.

Results:

Average Precision = 0.7600

Average Recall @ 1 detection per image = 0.2932

Average Recall @ 10 detections per image = 0.8185

Comparison of Models:

Method	Average Precision	Average Recall	Miss Rate
Pre-Trained HOG	0.056	0.063	0.937
Custom HoG	0.1720	0.125	0.875
Faster-RCNN	0.7600	0.2932	0.7068

We observe that faster-RCNN performs the best, all while being significantly better than Custom HOG and pre-trained HOG. Pre-trained HOG performs the worst, followed by custom HOG which remains better than the former but still has worse performance compared to faster-RCNN. We discuss some of reasons responsible for these observations:

- In pre-trained HOG, for detecting pedestrians, we tune the parameters (winStride, padding and scale) to get better results. However, since this tuning is image specific, there is no guarantee that the exact same parameters will work from image-to-image. Due to this, we falsely detect pedestrians and/or miss pedestrians entirely, simply due to poor parameter choices on a per-image basis. Leading to a bad performance of the model.
- As the pre-trained HOG is trained only on the INRIA dataset, its performance on the PennFudan dataset is not so good. Since we have trained Custom Hog on the same PennFudan Dataset, it gives better results as compared to pre-trained hog. Pre-trained faster RCNN is trained on the Pascal VOC dataset which consists of a large variety of images, so it gives good results on PennFudan dataset as well.
- Faster RCNN uses a selective search algorithm on the feature map to identify the region proposals. Selective search uses local cues like texture, intensity, color and/or a measure of insideness etc to generate all the possible locations of the object leading to significantly better performance.
- Faster RCNN uses a neural network and considers a wide variety of features to detect objects but custom hog considers only HoG features for detection. The deep

convolutional neural network uses many layers of convolution filter sets that learn a hierarchical representation of the input image data, where lower level convolutional layers will learn to detect simple features such as lines and textures, while higher level convolutional layers will learn features that are combinations of the lower level features. Overall making faster RCNN use a diverse set of features which helps it detect objects (pedestrians) even with significant occlusion, deformation.

- Detection is extremely fast in Faster RCNN as compared to custom HOG as there is no sliding window. forward pass of an entire image through the network is more expensive than extracting a feature vector of an image patch and passing it through an SVM. However, this operation needs to be done exactly once for an entire image, as opposed to the roughly 150 times in the SVM+HOG approach.The sliding window approach is computationally very expensive. To detect objects in an input image, sliding windows at different scales and aspect ratios need to be evaluated at every pixel in the image.

Prediction examples from different models:

The following images show the predicted outputs of different models on the same input images. Moving from left to right, the images are in order: Pre-trained HOG, Custom HOG, Faster-RCNN.

Observation: Pre-trained HOG mistakes a car/tree for a pedestrian. Even the correctly detected pedestrians do not have accurately placed bounding boxes. Bad performance of pre-trained HOG can be reasoned by points 1 and 2 discussed above. Custom HOG aptly detects two pedestrians clearly visible in the frame with comparatively accurate bounding boxes but misses out on the pedestrian in the background with significantly smaller size. This is because of the large window size used in the sliding window algorithm of the model. This forbids the model from detecting the objects (pedestrians) having size significantly

different (in this case smaller) from the used window size. Faster-RCNN performs the best and detects all the pedestrians visible in the frame with accurate bounding boxes. This good performance can be attributed to points 3 and 4 discussed above.

Pre-trained HOG falsely deduces some spurious matches to be pedestrians (points 1 and 2). Custom HOG misses out on one of the occluded pedestrians while accurately detecting another clearly visible one. Because of the limited set of features it uses to perform object detection, it is unable to detect the occluded figures. Faster-RCNN again out-performs both these models by correctly and accurately predicting both the pedestrians present in the frame (Points 3 and 4). We can also see that the bounding boxes are more accurately predicted by faster RCNN as compared to custom HOG, this is because the window sizes do not vary much in the latter. All the windows present in custom HOG are scaled versions of a certain specified window size, whereas this is not the case with faster-RCNN. This provides faster RCNN with better flexibility in terms of bounding box shapes.

Both Pre-trained HOG and custom HOG falsely detect some spurious matches. Even Faster-RCNN fails to predict and locate the significantly occluded pedestrian present. We can explain this behaviour using the same arguments as before. In case of faster RCNN, the

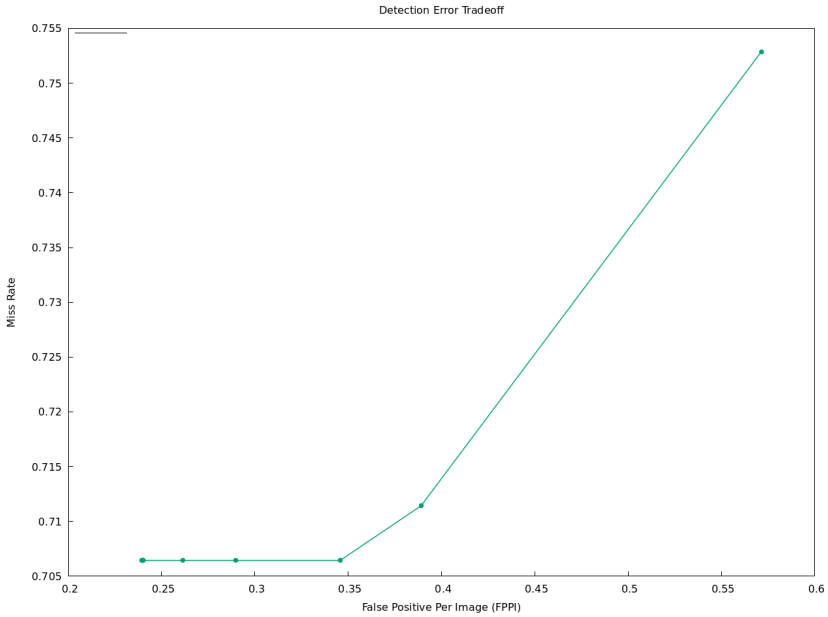
occlusion here becomes so prominent that even the extremely diverse set of features detected by this model fail to detect the occluded person.

RCNN performs significantly better by detecting many occluded pedestrians in the crowd where pre-trained HOG and custom HOG fail to do so. Similar arguments as before explain this case.

Custom HOG does detect the person's reflection in the glass but the bounding box is significantly inaccurate owing to the large window size used while training the model.

Detection Error Tradeoff Graph:

Faster RCNN:



Submitted by:

Rakshita Choudhary, 2019CS10389

Deepak, 2019MT10685