Computer Vision Project - Pedestrian Detection

Kimi Cai 4597214 Zhaoyang Cheng 4620585

July 2, 2017

1 Introduction

Pedestrian detection (PD) is an active research field with considerable challenges. Considering accurate pedestrian detection would be applied into various real-life applications directly such as robotics, self-driving car, video surveillance system, it has been investigated extensively in the past decade. Generally, the problem of PD can be broken down into two stages: 1) recognizing the humans correctly; 2) localizing them accurately. Nevertheless, due to a wide range of pose variants and cluttered backgrounds in reality, it still remains several challenging tasks.

The main techniques to solve the problem of PD can be generally categorized into two types. On the one hand, researchers focus on designing a pipeline of feature extraction manually and training on the high-performance classifier. As introduced in the lecture, the representative works include Viola and Jones detectors, Dalal and Triggs detector which extracts HOG features and train on SVM classifier and Deformable Parts Model(DPM) [10],[1],[3].

Nevertheless, the hand-crafted features introduce above commonly suffer the problem of high-dimensional and expensive computational cost. On the other hand, the evolutionary development of deep learning based algorithms have gained considerable attention and have greatly advanced the performance of the state-of-the-art. Some representative works can be found in [5], [4], [6].

Based on the tasks discuss above, the main research goal of the project is to develop an effective, fast and accurate pedestrian detection system based on Convolutional neural network(CNN). We build the deep CNN with the help of the open source library Tensor Flow. In terms of effective, our system shows outstanding performance on detecting difficult scenarios including detecting people at rather small pixels, different poses and overlapping situation. Furthermore, we test our system on the popular benchmark PASCAL dataset, experiment results suggest our system is close to the state-of-art, with the mean average precision (mAP) of 78 percent. In addition, our system demonstrates the ability to process the real-time video, achieved 10 frames per second (fps).

2 Algorithm Overview

We made a visualization of how Faster RCNN works as shown in Figure1. There are 4 parts implemented by ourselves in Faster RCNN . 1, Convolution layers, Faster RCNN use VGG16[8] to extract the feature map from an image, the feature map is used in Regional Proposal Network(RPN) and fully connected layer ; 2, RPN is used to generate proposals, first we generate some anchors(we will give an explanation of anchors later) which to be classified as foreground or background in softmax function, then we use bounding box regression to transform the anchors to accurate proposals ; 3, Region of Interest Pooling, this layer receives feature map and proposals , outputs the proposals feature which is classified in fully connected layer; 4, Classification, classifying the proposal feature and using bounding box regression again to locate the bounding box accurately. Our code is mainly under directory ./cvProject1/lib/rpn_msr , ./cvProject1/lib/networks and ./cvProject1/faster_rcnn, in those directories, if the python function does not have a license, then it is written by us.

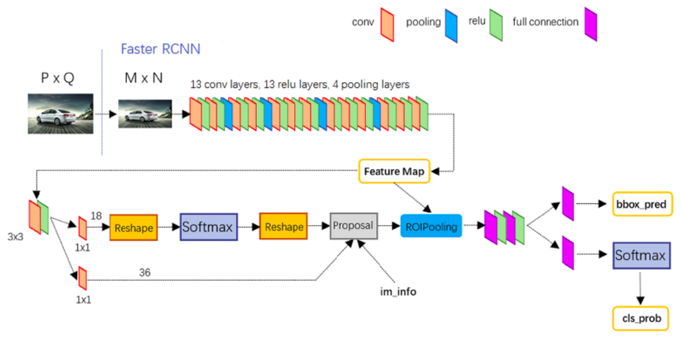

Figure 1: Faster RCNN pipeline

2.1 Region Proposal Network

The feature map from VGG16 is feed to a RPN. We have explained why traditional methods in generating proposals is time-consuming, while Faster RCNN uses a RPN which has great improvement in generating proposals. As shown in Figure 2 There are 2 pipelines in RPN. Upper pipeline

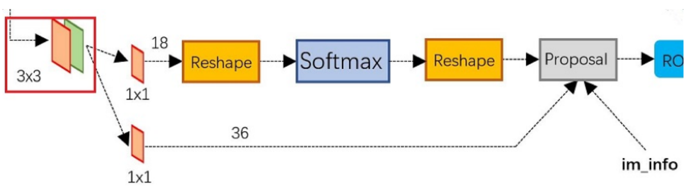

Figure 2: Faster RCNN pipeline

classifies the anchors to acquire foreground and background by softmax,and in proposal step we only need those foreground objects. Lower pipeline calculates the transform parameters which map anchors generated by ourselves to Ground Truth box.

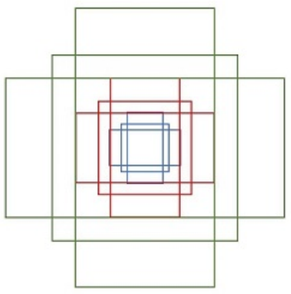
Figure 3: 3*3 anchors

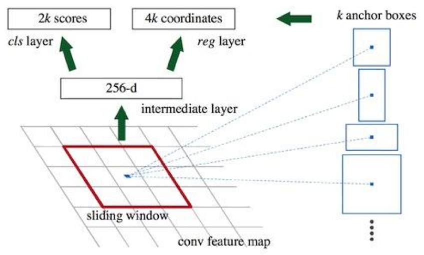

Figure 4: Region Proposal Network[6]

2.1.1 Anchors

an anchor is an vector [x1,y1,x2,y2], which represents 4 coordinates of top left and lower right point, these anchors have 3 sizes and 3 ratios, k=9 rectangles in total which are shown in Figure 3, the center of all k rectangles is a point in feature map, for a size of W*H feature map, there are W*H*k anchors in total , this is the idea of multiple scales detection. In Figure 4 which comes from Faster RCNN paper, author used ZF net[11] which last convolution layer is 256d, meaning every point in the feature map is 256d , however , as we use VGG16 net , every point in our feature map is 512d, the intermediate layer is used to combine information of the point and its 3*3 neighbors. Assuming there are k anchors on every point, the output of intermediate layer is rpn_cls_score of size [1,H,W,2*k] assigned to background and foreground, and rpn_bbox_deltas of size [1,H,W,4*k] assigned to bounding boxes regression parameter $[t_x, t_y, t_w, t_h]$, in Figure2, the two reshape functions before and after softmax is just a handle of TensorFlow, we denote the output after the second reshape function *rpn_cls_prob_reshape* of size[1,H,W,2*k], which is feed to proposal generating function following them.

2.1.2 Bounding Boxes Regression

In Figure 5 red box is our anchor, green box is GT box, we know, a set of generated anchor can not approximate the Ground Truth(GT) box accurately, we need a box regression to transform our anchor to a prediction box which is more close to GT box. a box is basically represented by

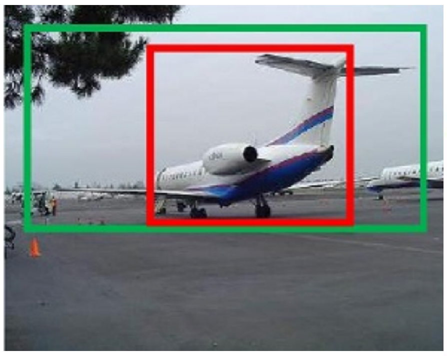
Figure 5: red anchor and green ground truth box

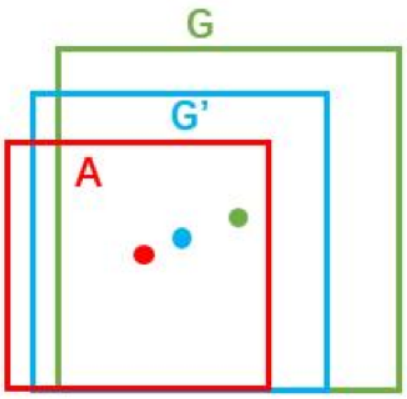

Figure 6: Box Regression

a vector(x,y,w,h), (x,y) is coordinates of window center point, w and h is width and height. in the Figure6, Red box A is our Anchor, green box G is GT box, we need a transformation to map A to a prediction box G' which is more close to Ground Truth box . i.e. $T(A_x, A_y, A_w, A_h) = (G'_x, G'_y, G'_w, G'_h), and \quad (G'_x, G'_y, G'_w, G'_h) \approx (G_x, G_y, G_w, G_h)$. Assume T is $[t_x, t_y, t_w, t_h]$

$$\begin{array}{rll} \textbf{Translation} & G^{'}_x = A_w * t_x + A_x & G^{'}_y = A_h * t_y + A_y \\ \textbf{Zoom} & G^{'}_w = A_w * exp(t_w) & G^{'}_h = A_h * exp(t_h) \end{array}$$

hence, the transformation parameters are :

$$t_x = (G^{'}_x - A_x)/A_w \qquad t_y = (G^{'}_y - A_y)/A_h$$
$$t_w = log(G^{'}_w/A_w) \qquad t_h = log(G^{'}_h/A_h)$$

For GT boxes, we also have $[t^*_x, t^*_y, t^*_w, t^*_h]$ by boxes regression. We use a smooth L1 loss function[4], Hence, the loss is

$$L_reg(t, t^*) = \sum_{i \in x,y,w,h} smooth_{L_1}(t_i - t^*_i)$$

2.2 Generating Proposals

Proposals Generating Algorithm:

```
Input:rpn_cls_prob_reshape[1,H,W,2*k]
Input:rpn_bbox_deltas[1,H,W,4*k]
downsample ratio feat_stride from feat map to image =16
For each location i in (H,W) :
    generate k anchors centered on i
    shifted_anchor=map anchor to original image(centered on pixel)
    proposal=regression(rpn_bbox_deltas,shifted_anchors)
sort proposals by rpn_cls_prob_reshape
```

clip out_edge proposal
remove proposals either height or width < thresholde.g. 16
adopt Pre_nms_topN(e.g. 6000) proposals
apply NMS with threshold 0.7 to remaining proposals
adopt After_NMS_topN proposals(e.g. 500) after NMS

2.3 Classification

The proposals are set to fixed length by RoI pooling[4] , then we can do classification, as shown in Figure7, we apply bounding box regression to every proposal again to acquire more accurate prediction bounding, and softmax function will output which class the proposal belongs to.

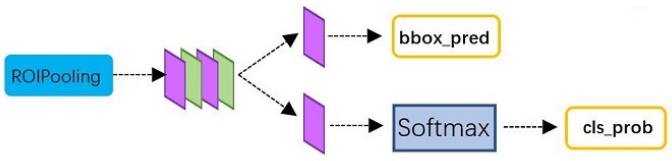

Figure 7: classification

3 Train Stage

Figure8 shows a work pipeline of Faster RCNN train stage, at left part of Figure8, we compute Loss of RPN by formula given in [6]

$$L(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{1}$$

In formula(1), i is index of anchor, p_i is probability of foreground softmax prediction, p_i^* is Ground Truth, when an anchor has an IoU greater the 0.7 with Ground Truth, then it is judged as foreground, those lower than 0.3 are judged as background, otherwise it is dropped. L_{cls} is a cross_entropy loss for computing probability of being foreground or background, L_{reg} is a $smooth_{L_1}$ loss for bounding boxes regression. In practice, N_{reg} is much larger than N_{cls}, so we use a λ to balance them. At the right part of Figure8, we compute the loss of RCNN , we use the proposals from Proposal Generating Algorithm as region of interests(rois) which is marked by a blue rectangle in Figure8, the foreground probability is used as bbox_inside_weights ,comparing N_{reg} and N_{cls} to get λ which is used as bbox_outside_weights. Then the loss function is the same as it in rpn.
The total loss is sum of rpn_loss and RCNN_loss. We use a Momentum Gradient Descent method[7] to optimize the loss.

4 Result

According to the results of some early tests, we will focus on the model trained on Pascal VOC2007 dataset in this section.

4.1 Detection under complex scenarios

Before we do any quantitative test on our system, we first used some special qualitative tests to evaluate the system's performance on complex scenarios. The scenarios we tested includes small pixels and overlapping situations.

Figure 9 shows the performance of our system in overlapping situations. As we can see in the figure, overlapping occurs many times, especially near both Vertical edges. Our system succeeded in detecting every pedestrian when the threshold of confidence score is set to 0.7. Also, no false

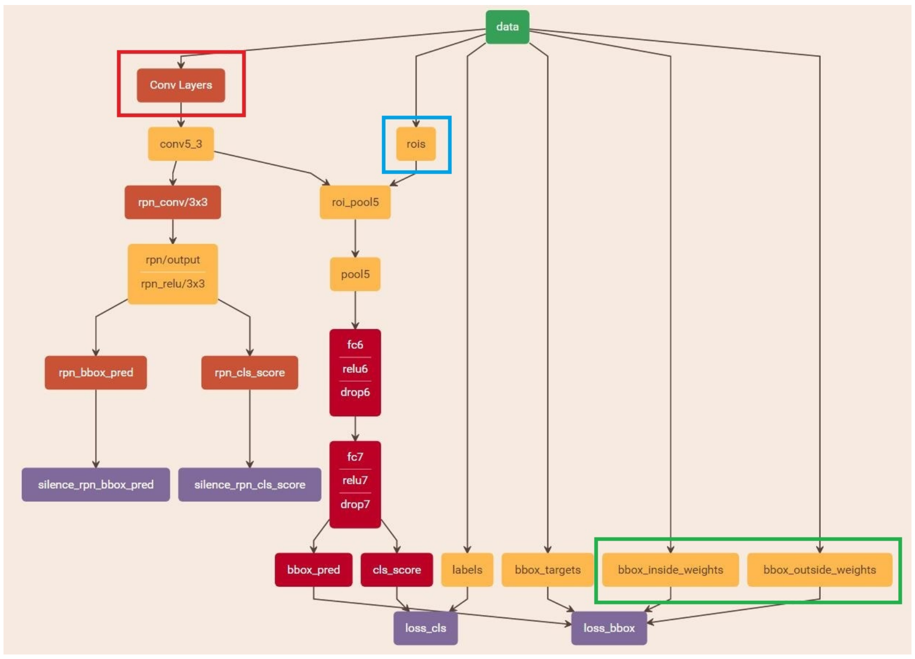

Figure 8: Train Stage

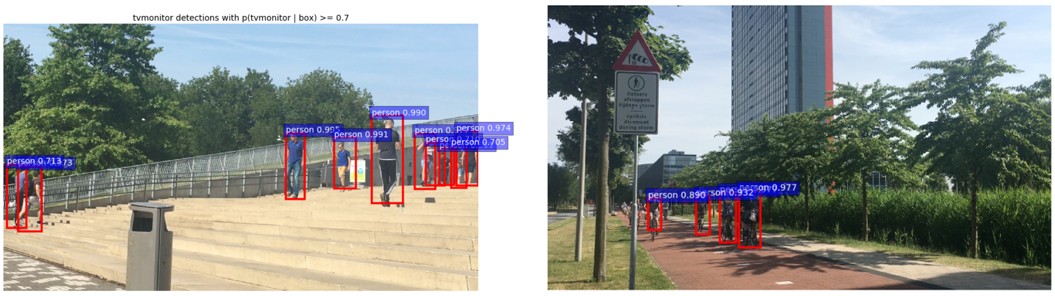

Figure 9: Results on overlapping scenarios

positive bounding box is given out.

Figure 10 shows the results given by our system. The original pictures were taken in the campus. For small pixes, according to the information of bounding box in Figure ? whose resolution is 856×502, the scale of pedestrian we detected is 16×8 pixels. The ratio meets the requirement of medium scale(between 30-80 pixels in 640×480 resolution)[2] which is essential for real world application. for example, automotive applications. It can be computed that An 80 pixel person whose height is around 1.7m is just 1.5s away at an urban speed of 55 km/h. So it is quite reasonable to ask a detection system to meet this requirement.

4.2 Detection Accuracy

The quantitative evaluation is based on mean average precision (mAP) . As mentioned above, our system is trained on Pascal VOC2007 dataset, so we use the method provided by the organizer of Pascal VOC to evaluate our system. Another reason why we use Pascal VOC for evaluation is that we can easily get many benchmarks of this popular dataset. Although the online evaluation server for 2007 dataset is no longer available, the Matlab toolbox 'VOCdevik' can be downloaded from Pascal VOC's website. We studied the relevant document and mastered the method of use.

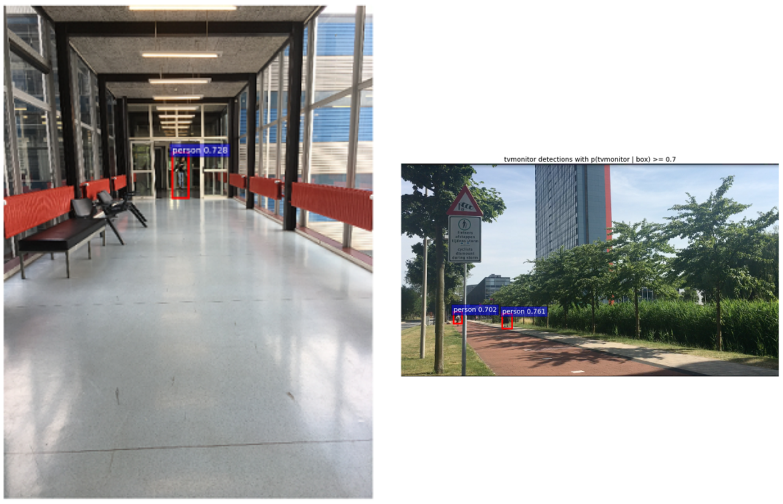

Figure 10: Results on small pixels

4.2.1 Test on Pascal VOC

'VOCdevikt' toolbox reads the detected bounding boxes and confidence scores given by the detection system and give out the precision/recall curve and the final mAP. It is worth to mention that Pasca VOC2007 contains twenty categories of objects. Although we trained all twenty categories, all the pre-processings on the dataset we did is based on 'person' categories, for example, we balanced the positive and negative parts of the dataset based on the document 'trainval_people.txt'. This txt document is provided in the dataset and record whether each picture contains 'object: person'. So our expectation of our model is that the mAP in single category 'person' should outperform other categories. However, we still compute the mAP in all categories as most of benchmarks we found online is based on various categories.

Figure 11 shows the precision/recall curve of our system's result on Pascal VOC2007 dataset. The left one is our first edition after 70 thousand iterations training and the right one is the final edition after 150 thousand iterations training. We found that the accuracy increases with the amount of iterations and tends to be stable when the amount of iterations reach around 120 thousand. The final mAP on category 'person' is 78.4%.

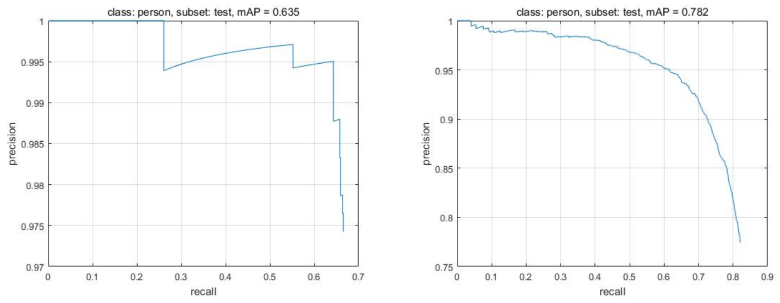

Figure 11: Detection precision/recall curve on Pascal VOC2007 test set (70k/150k iterations)

As we can see in Table 1, our mAP on various categories does not outperform other popular models in object detection. However, it is still acceptable as we focus more on person detection.

4.3 Detection Speed

Another important evaluation criterion is the frame rate(running time) of the detection system. The increasing speed makes it more possible to realize the real-time monitor. Ren et al[6] realized 5fps rate(200ms running time) in their RPN+Fast R-CNN system. Our system has similar performance in frame-rate. The running time of our system keeps in the range between 100ms and

Table 1: Detection results on Pascal VOC2007 test set

training		test		
method	#boxes	method	#proposals	$mAP(\%)$
SS[9]	2k	SS[9]	2k	58.7
EB[12]	2k	EB[12]	2k	58.7
Ren[6]	2k	Ren[6]	300	59.9
Our model	1k	Our model	500	57.6
Our model(People)	1k	Our model(People)	500	78.2

200ms. However, our system use 500 proposals in detection and the size of image is around 2MB, which is much larger than 300 proposals and 500 KB size which is used in Ren's system.

The decrease in proposals used will undoubtedly lead to faster speed of detection, but this will also result the decrease in mAP. So the best way to improve the speed is improving the hardware device used. The GPU we used in this project is GTX1060, whose performance is medium level among the same kind of products. If we have more advanced device, we have the confidence to realize the real-time detection without mAP decrease. We realized a function with what an imported video can be sampled, detected and reconstruct. The result of a sample video is available in https://drive.google.com/file/d/0B3m3OpanadRYLUdpRFQ5Z25WTFk/view?usp=sharing.

5 contribution

Zhaoyang Cheng : 65% **,including code and other work**
Kimi Cai: 35% **including code and other work**

References

[1] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[2] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: A benchmark. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 304–311. IEEE, 2009.

[3] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[4] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[6] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[7] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[8] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[9] R. Uijlings, A. van de Sande, T. Gevers, M. Smeulders, et al. Selective search for object recognition. *International journal of computer vision*, 104(2):154, 2013.

[10] P. Viola, M. J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *null*, page 734. IEEE, 2003.

[11] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *arXiv preprint arXiv:1311.2901*, 2013.

[12] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV (5)*, pages 391–405, 2014.