UNIVERSITY OF CALIFORNIA
RIVERSIDE

Pedestrian Detection Based on Deep Learning

A Thesis submitted in partial satisfaction
of the requirements for the degree of

Master of Science

in

Electrical Engineering

by

Li Chen

September 2017

Thesis Committee:
Dr. Qi Zhu, Chairperson
Dr. Hyoseung Kim
Dr. Daniel Wong

The Thesis of Li Chen is approved:

_____

_____

_____

Committee Chairperson

University of California, Riverside

Table of Content

List of Figures

Figure

Chapter 1 Introduction

1.1 Background

According to the preliminary data from Governs Highway Safety Association, the number of pedestrian fatalities in the United States increased significantly recent years. About 6,000 pedestrian fatalities are estimated to have occurred in 2016, which could make 2016 the first year in more than two decades with more than 6,000 pedestrian deaths [1]. On the other hand, intelligence techniques, such as self-driving technology, develop rapidly to benefit people's daily life. Therefore, automotive safety has been an important issue that people are concerned about. In order to guarantee their safety, the accuracy of pedestrian detection is much more significant.

Besides the automotive safety, pedestrian detection has also been widely used in many other applications, such as robotics and surveillance. These critical applications attract great attention to the researchers. They come up with different methods to improve the accuracy for pedestrian detection. From designing the algorithms for the base features of pedestrians to improving the learning algorithms, researchers have made great contributions to this area.

However, pedestrian detection is a challenging task due to its complexity background as well as various body sizes and postures. Traditionally, researchers focus more on capturing low-level feature extraction of pedestrians by manually designing some algorithms. Such hand-crafted method has two steps to compute features, selecting the region of interest in the image, like the corners, and then using a descriptor to calculate the characteristics of the region. The descriptor can distinguish the characteristics from

others. However, using hand-crafted method has to find a good trade-off between the accuracy and efficiency of computation.

Recently, with the development of deep learning technology, researchers take advantage of deep neural network, especially the convolution neural network to automatically extract the features from original images. Instead of extracting the low-level feature as traditional methods, deep learning method extracts high-level feature due to deep layers of convolution neural network. For the first several layers, convolution neural network just learns the low-level features, such as edge, dots and colors, while the later layers will learn to recognize the general shape of the objects and get a high-level representation of the image. So, deep learning method can discover multiple levels of representation of the images through multiple layers and can be considered as feature extractor. These features extracted by convolution neural network are directly learn from the data, which are different from the traditional hand-crafted methods that the features are designed by the experts.

Besides feature extraction for the pedestrians, classification and detection of the region is also an important part that researchers are focus on. For hand-crafted methods, typically it should combine with the classifiers to classify and detect the features that are extracted before. For the deep learning methods, it can be trained from end-to-end, extracting features from the original image and classifying the result from the last layer of convolution neural network, which means that convolution neural network can be considered as both feature extractor and classifier.

1.2 Objectives

In order to get higher efficiency and accuracy on detecting pedestrians, utilize deep learning method could be an effective way. In the previous work, researchers combine the hand-crafted methods with deep neural network, leading good results of pedestrian detection, but hand-crafted methods are still slower than the efficient network, since these are implemented on CPU while the region-based CNN can be implemented on GPU.

Recently, Faster RCNN, proposed by Shaoqing Ren[2] can purely use convolution neural network to achieve the detection of objects without any hand-crafted method. The result has shown that it has great performance on object detection. Pedestrian detection, a specific category of objects, can also utilize part of network in Faster RCNN to implement the process of detection in deep neural network.

Chapter 2 Literature Overview

2.1 Pipeline for Pedestrian Detection

Pedestrian Detection has been an important research area in computer vision for decades of years and researchers propose a lot of methods. Typically, they solve such problem have similar pipeline: region proposals, feature extraction and region classification.

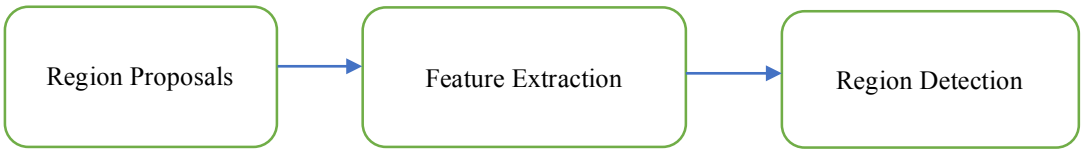

Figure 1: pedestrian detection pipeline

At the first stage, regions of interest are selected in the images. Researchers come up with some methods to select the regions as candidates, some of them include parts of pedestrians, others do not.

Then the features of these candidate regions will be extracted. These features can be considered as representations of the whole image and feed to the classifiers.

Finally, train the classifiers to recognize these features. These classifiers can provide a binary flag to indicate whether pedestrians exist in the candidate regions.

2.2 Traditional Methods

2.21 Region Proposal Methods:

For region proposals, traditionally researchers use hand-crafted method, selecting the region of interests in the image based on the low-level features. Sliding window is the most common method that people use in the research. In general, the detected image is

fixed and the sliding window with different scales will traverse the whole image from the top left to the bottom right, this method is simple but time-consuming.

EdgeBox [3] is another method based on sliding windows. Instead of traversing every location of the image, this method accelerates the process of proposing the candidate region. The authors use edges information to generate bounding box for the objects. The main idea is that they observed the number of contours that totally enclosed in the bounding box can indicate the probability that the bounding box includes objects. The edges correspond to the boundaries of the objects and when all the pixels of the edges are in the bounding box, the edges are thought to be wholly enclosed in it. Since the number of the bounding box could be so large, the authors came up with a method that they must score these candidates to select the best one. Firstly, they utilized the Structured Edge detector proposed in [4] and [5] to get the initial edge-map. After that they grouped the neighbor pixels with the similar orientations as the edge groups and calculated the affinity for the neighboring groups, shown on the third image of the Figure 2. According to the affinity, the boundary and the score of the box can be calculated. They used sliding window method to find the potential candidate bounding boxes with different scale and aspect ratio. Figure 1 shows the process of edge box method that find the example of correct bounding box.

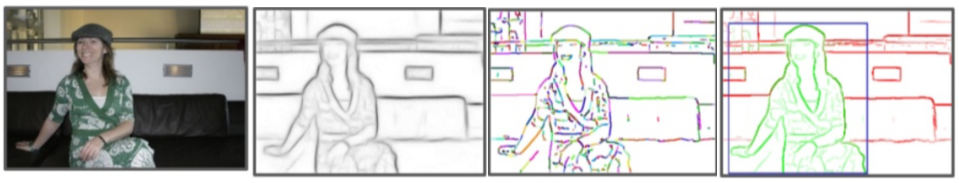

Figure 2: from left to right: Original image, initial edge map used Structured Edges, edge groups and example of correct bounding box [3].

Selective search [6] is a hand-crafted method that based on super-pixels. It generally groups the pixels of similar colors together and generates the bounding box around these regions. There are three main design considerations they proposed in the paper: capture all scales, diversification and fast to compute [6]. For capturing all scales, they proposed the hierarchical grouping [6] that combined with segmentation method, generating all locations with different scales and grouping these regions until the single region is left. The way to group the regions is the following: 1) use the segmentation method to create the initial image. 2) calculate the similarity between all these neighboring regions 3) group the two most similar regions and calculate the new similarity between this new region and its neighboring regions. 4) repeat the process until the whole image becomes a single region. The steps from 2) to 4) are considered as greedy algorithm. For diversification, the authors also proposed several strategies: complementary color space, complementary similarity measures and complementary starting regions[6]. For the fast to computer, the selective search generated fewer regions than the exhausted method that generates all the regions from the bottom left to the bottom right in the image. In [7], the authors utilized the selective search on Daimler Pedestrian Monocular Dataset. From the Figure 3, we can see that the image including a pedestrian, cars, ground, trees and sky are segmented and grouped by the similar pixels. The bounding boxes also take into grouping these regions together. In a word, using selective search method can improve the computational efficiency and reduce the cost of computation compared to exhausted search method.

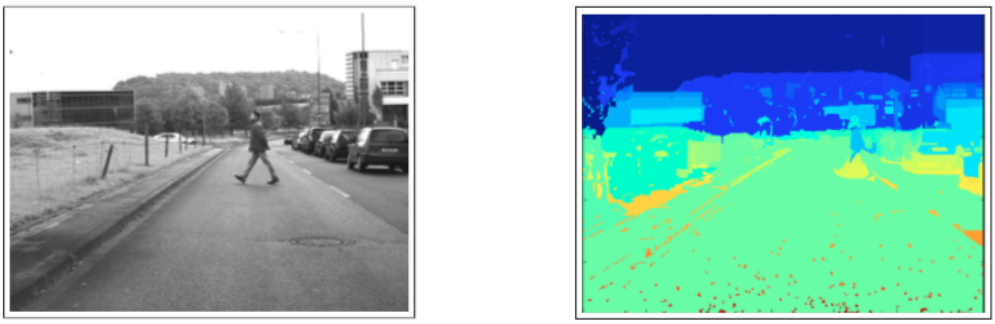

Figure 3: Left is the original image and the right one is the image that used selective search.[7]

2.22 Feature Extraction Methods

Histogram of Gradients (HOG) and Aggregate Channel Feature Algorithm (ACF) are two typical methods to extract the features.

For the Histogram of Gradients, it is a feature descriptor that covert the image to a feature vector. The authors proposed that utilize the distribution of local intensity gradient or the edge directions can represent the shape and appearance of the local objects

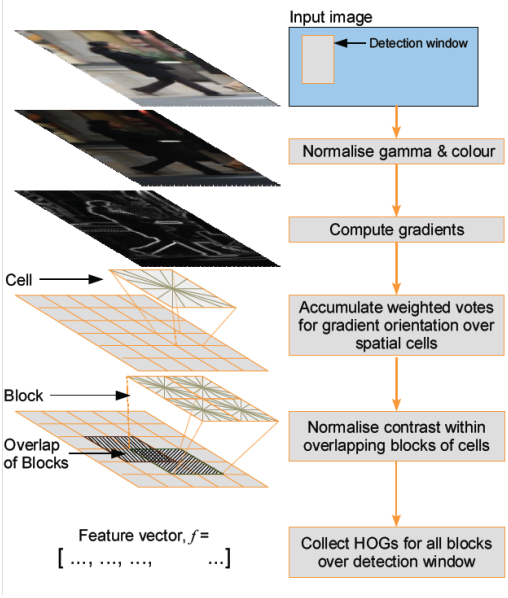

Figure 4: HOG Descriptor [8].

well, since the direction gradients of the edge or corners can be so large, they can contain large information about the shape and appearance of objects in the image. In order to gain a better performance on the images, gamma and color normalization should be preprocessed at the very beginning. For the color image, turn it to the gray one. For the image with uneven lightness, use the gamma correction can raise or reduce the contrast of the whole image. Then, calculate the gradient of each pixel, getting the appearance of the pedestrians. After that, accumulate histogram of direction gradients for each cell, where the cell is divided by the fixed square sliding window. Because of the diversity background and other conditions, the range of the gradient changing will be large, normalize the features becomes so important. Set the 2x2 cell as one block and each block has overlap, which helps the result of detection. Normalize each block and collect the HOGs for all blocks over detection window. These HOGs are the features of the image.

Aggregate Channel Feature Algorithm is proposed by Piotr Dollar in [9]. The main idea is aggregating channel features with fast feature pyramids. As Figure 5(1) shows, when given an image, we need to create the feature pyramids which are multi-scale representations of the image and the features includes normalized gradient magnitude, histogram of oriented gradients and LUV color channels. Instead of calculating channels at each scale, fast feature pyramids just need to calculate only a sparse set of s, for example, once for every eight scales. The pipeline is shown in Figure 5(2). The feature channels of other scales are calculated by directly scaling based on the nearest scale which we have calculated, the vision of the pipeline is shown in Figure 5(3).

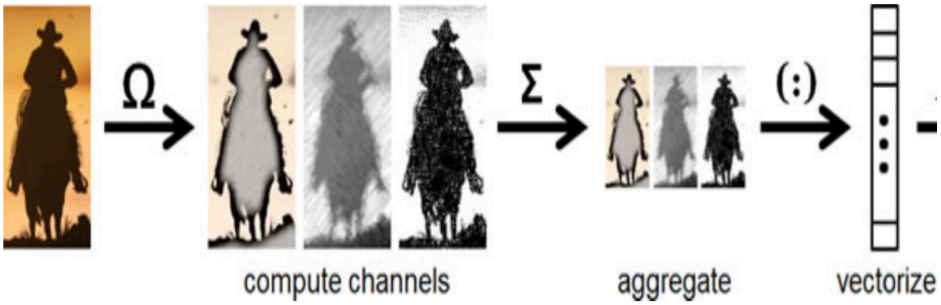

Figure 5(1). Overview of ACF detector [9]

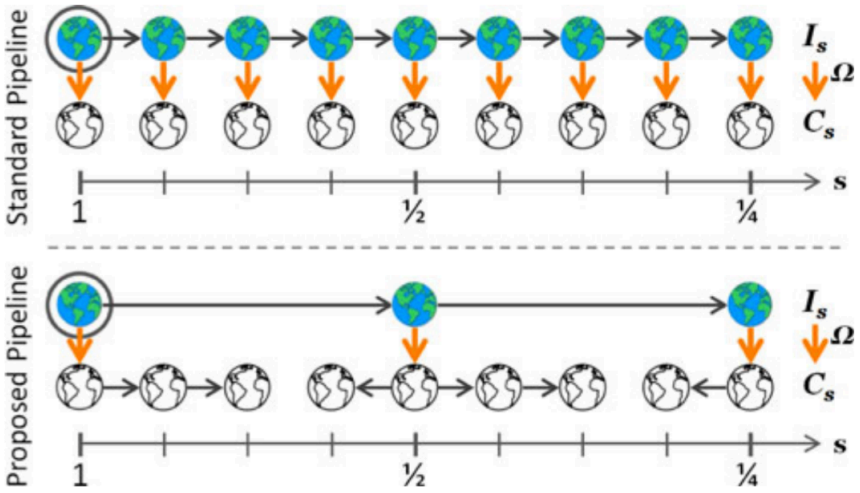

Figure 5(2): Fast feature pyramids

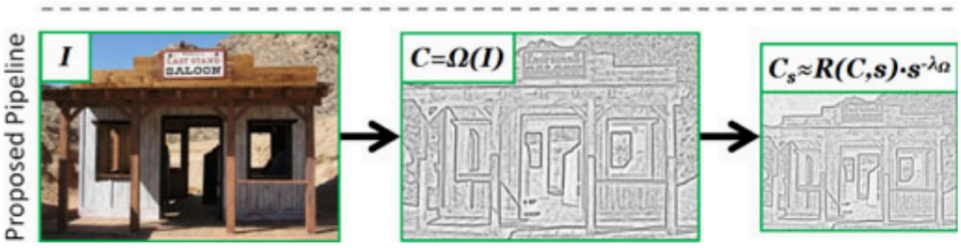

Figure 5(3): Feature channel scaling[9]

2.23 Region Classification Methods

Adaptive Boosting is an algorithm that can combine a sequence of other learning algorithms into a weighted sum that represent a final output of the boosted classifier. The process is similar as the Figure 6 shown below. It starts to assign the equal weights to each observation. D1, the decision stump, represents the first learning algorithm, and we can see that three plus are predicted wrong in box 1. Therefore, we assign the larger weights to the wrong predicted plus in box and use a second learning algorithm represented by D2, which is applied to predict these wrong observation correctly. Again, in Box2, three observations (minus) are predicted wrong, so in Box3, three minus labels are assigned with higher weights for the third learning algorithm to learn. Continue this process, adding new algorithms until reach the accuracy limitation. Adaptive Boosting is like the Box4 that combine all the learning algorithms to form a strong prediction that perform better than any individual learning algorithm.

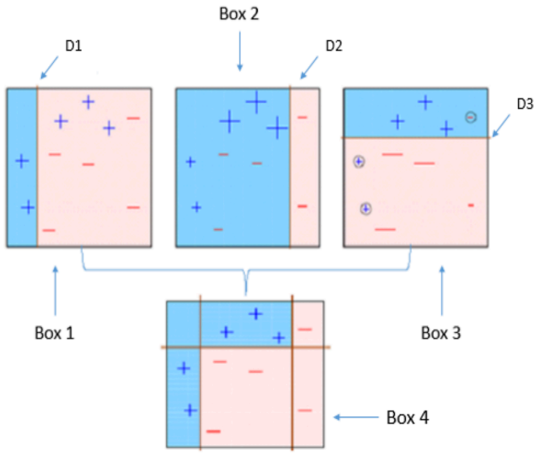

Figure 6: Diagram that explains the Adaptive Boosting [10]

Support Vehicle Machine is a supervised machine learning algorithm that mostly are used for classification. As shown in Figure 7, numbers of data items are plotted with their corresponding value in the two-dimensional space, if there are n number of features, it should be plotted in n-dimensional space. The support vectors are the co-ordinates of each observation. The main idea is to find the hyperplane to differentiate two different classes. A good hyperplane should set the accuracy as the priority and it can separate the classes better, choosing the one which can maximum the distance between the hyperplane and the nearest data point. Besides the linear problem, SVM can also deal with non-linear problem. It has a technique called kernel, which are some functions that can transform the low dimensional of the input space to a higher dimensional space. It helps to convert the non-separate problem to be a separate problem, classifying the different regions.

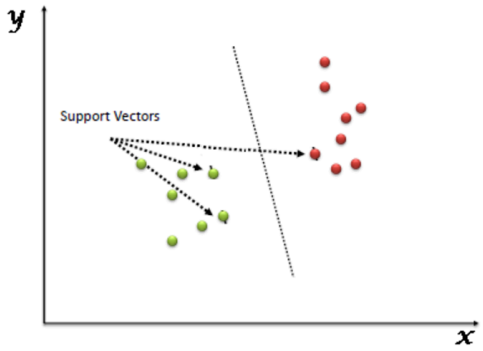

Figure 7: SVM [11]

2.3 Deep Net Methods

Besides the traditional methods are illustrated before, deep net is a totally different method to extract feature and classify the regions and it has been an important method in recent research area due to its high efficiency.

For feature extraction, instead of designing feature by expert beforehand, deep neural network can automatically extract the feature from the region proposals of the image, especially, the convolutional neural network. It is a supervised learning network with multiple layers, including the input layer, output layer and several hidden layers. Figure 8 shows the simple order of the layers in CNN network. After the image is fed into the

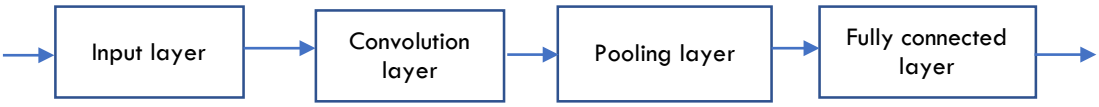

Figure 8: Layers of convolution neural network

input layer, it will enter the convolution layer. In convolution layer, the features of the image can be extracted by the convolution operation and several feature maps would be produced. These several feature maps correspond to different kinds of features in the image. During the convolution operation, the weights and bias are learned by the neurons and are adjusted through the backpropagation. These weights and bias are shared between the same feature map, but for the different feature maps, the weights and bias are different. In addition, each neuron in the feature map is related to the local receptive field of the previous layers. Pooling layer is very useful to reduce the dimension of the features, improving the efficiency of the parameter learning. It operates independently on

each feature map and remains the most important value in the sliding window, for example, the max operation. For fully connected layer, neurons are fully connected to all neurons in the previous layers by the activations and these activations are computed by the matrix multiplication plus a bias offset.

In addition, CNN network can also classify the feature by itself by the end of fully connected layer. Through calculating the loss function between the ground-truth labels and predicted output value, adjust the weights and bias to minimum the loss value. After training these parameters, the network can automatically classify the image whether includes the objects. Figure 9 is an example that shows how the pedestrians in the image can be detected through convolution neural network.

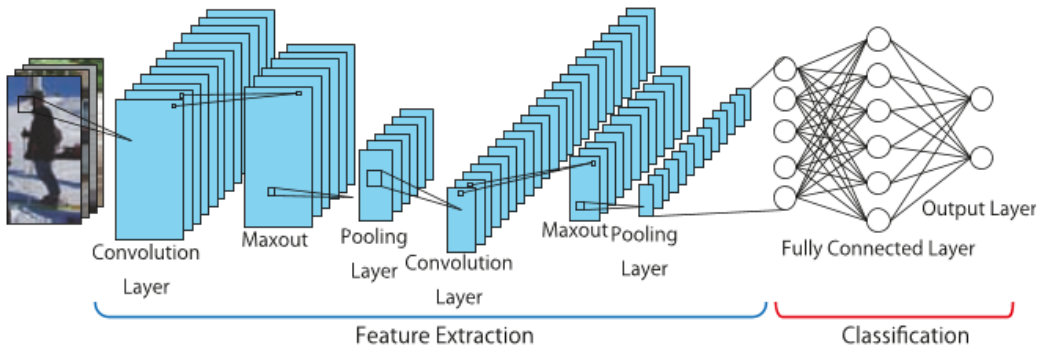

Figure 9: Example of pedestrian detection based on CNN network [12]

Chapter 3 Related Work

3.1 Object Detection Methods

Recently, the algorithms for object detection have made great breakthrough on both the computation efficiency and accuracy, especially, the three related algorithms: RCNN, Fast R-CNN and Faster R-CNN, they are all based on deep learning. The difference is that Faster RCNN can implement all the processes in deep nets: feature extraction, region proposals and region classification as well as bounding box regression, other two still need to implement some of the processes by hand-crafted. These differences are drawn in Figure 14.

R-CNN is the earliest one among these three algorithms, proposed by Ross Girshick. The authors used selective search to get the regions of interest in the image, resized these regions and put these into the convolution neural network to extract features. At the end, these features will be fed into the SVM classifier to judge what kind of objects they are. The main disadvantage is that all the regions must feed into the CNN one by one which will bring large computation costs. In addition, the input size of CNN is fixed and all the region proposals must be resized. The process of R-CNN is shown in Figure 10.

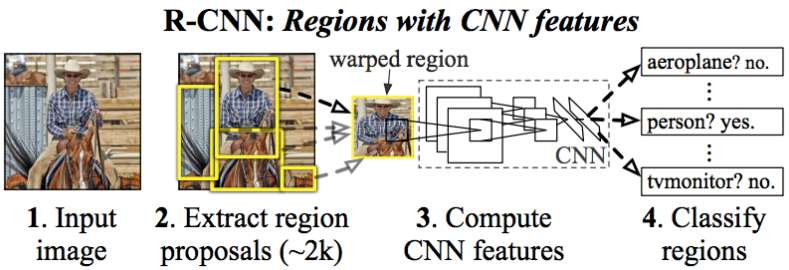

Figure 10: R-CNN [13]

Instead of feeding region proposals into CNN one by one, Fast R-CNN input the whole image to convolution neural network. Firstly, it still uses the selective search to get the region proposals in the original image. After getting the feature map through CNN, the region proposals in the original image can be mapped on this feature map. Since the size of the input images are different, we need to get the fixed size of the feature maps from the original images and feed them into the fully-connected layer, the ROI pooling layer is used to fix this problem, making each feature map has the same fix-length representation after this layer, see the details in Figure 11. This layer performs max pooling on each window of feature map and produces a small feature map of fixed size. Finally, the outputs of the CNN are fed into two sibling layers: the classification layer and bounding box regression layer. Therefore, the whole system can be trained from end to end, except the selective search for region proposals. Figure 12 shows the whole process that how Fast R-CNN works.

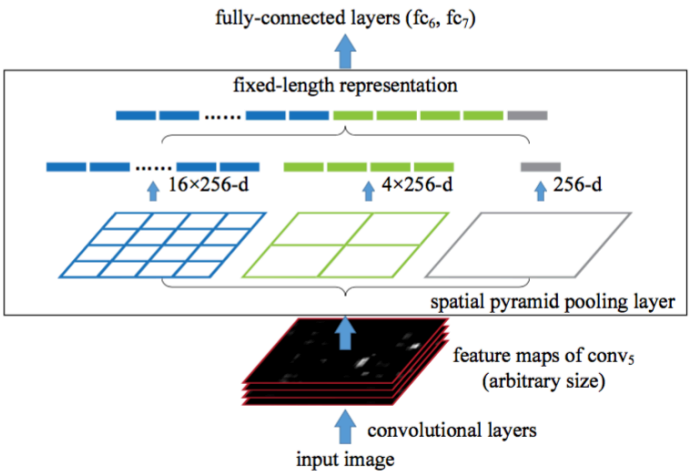

Figure 11: A network structure with a spatial pyramid pooling layer [14]

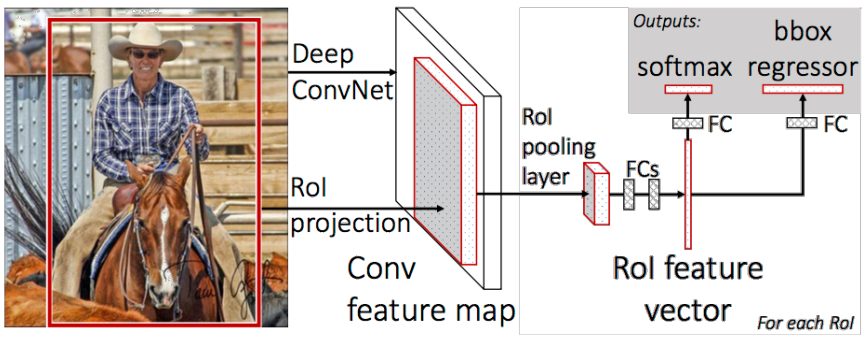

Figure 12: Fast R-CNN [15]

Faster R-CNN, a much more efficient algorithm that combines all these processes in the pure deep net without any hand-crafted methods. This algorithm composes two main network, Region Proposal Network and Fast R-CNN. Region Proposal Network(RPN) is used to propose the candidate regions, telling which module should be looked. The output of RPN are used as the input to the Fast R-CNN, detecting what these candidate regions are. Figure 13 illustrates the general flow path of Faster R-CNN.

Since Faster R-CNN improves accuracy on several multi-category benchmarks and has high efficiency that totally implements the end-to-end training in the convolution neural network, it also can be considered to apply for pedestrian detection, a specific part of the objects.

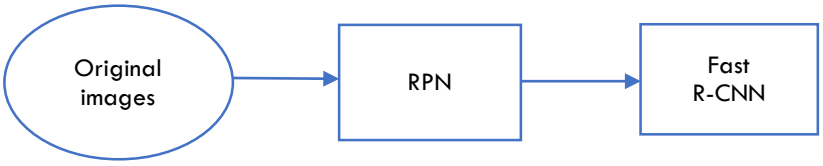

Figure 13: Process of Faster R-CNN

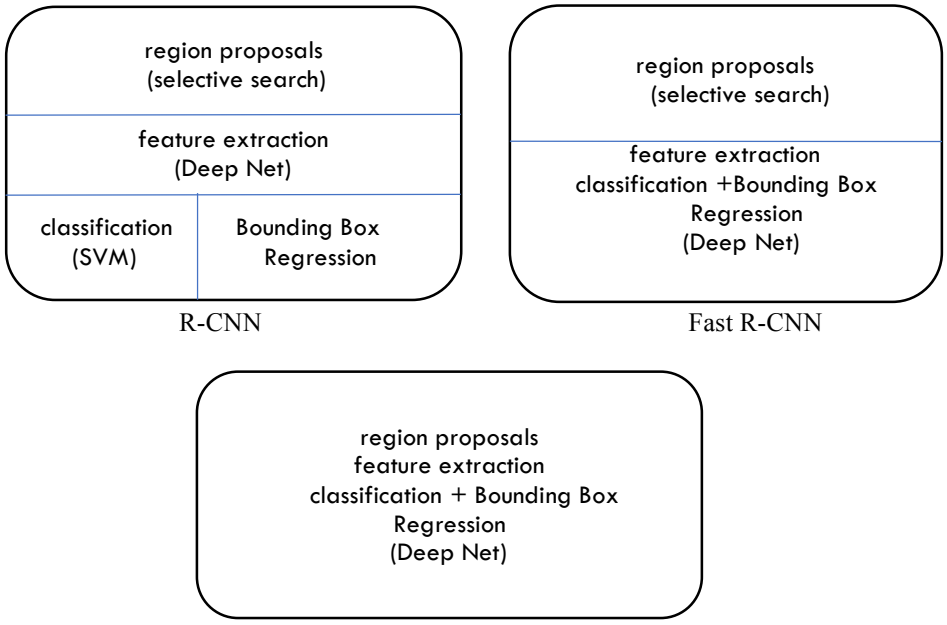

Figure 14: Methods comparison among R-CNN, Fast R-CNN and Faster R-CNN

Chapter 4 Methodology

4.1 Framework

Instead of using all the deep networks in Faster R-CNN, RPN can classify the pedestrians and the performance will be downgrade if the following Fast R-CNN is implemented, according to the result from [17]. When detecting the pedestrians, there are two objects in total, the pedestrians and background. Utilize the RPN could detect whether the candidate region is a pedestrian or not. Since pedestrian has its special shape, some changes have been made when implement the RPN. The framework of the methodology is shown in Figure 15.

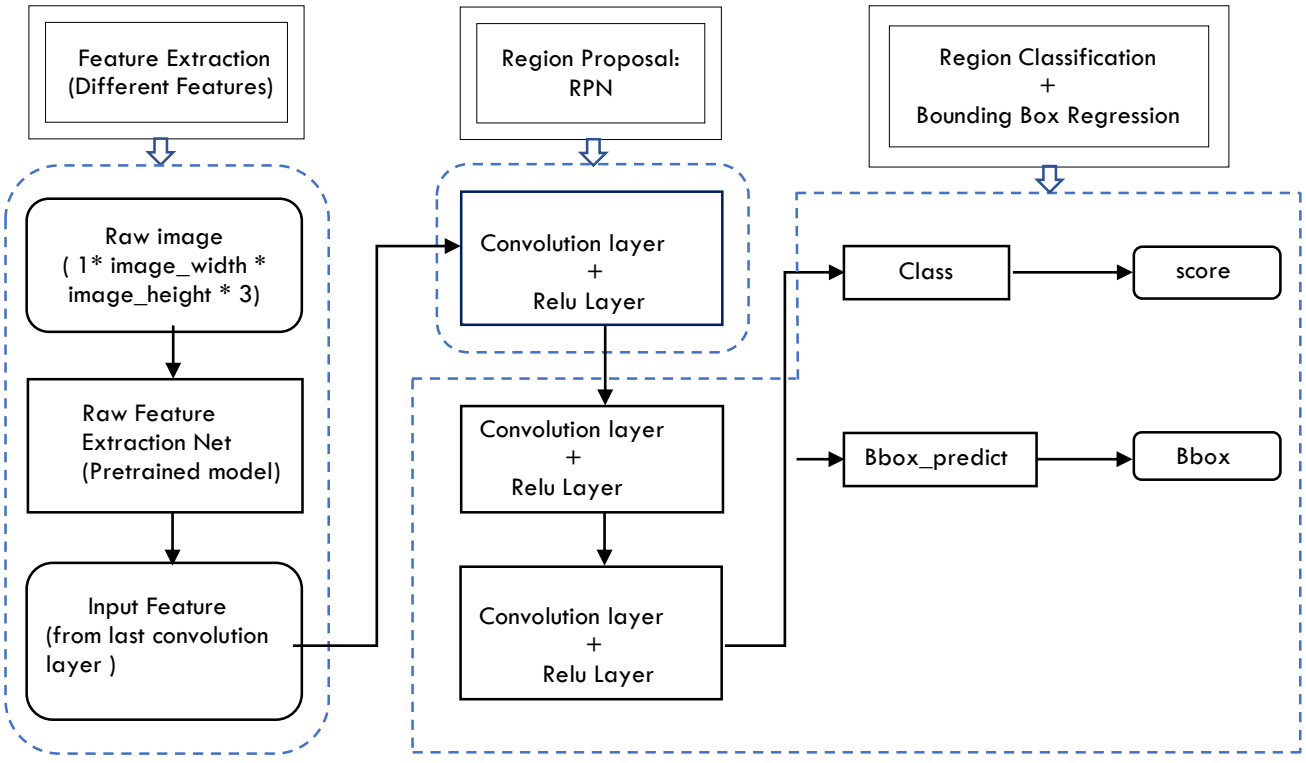

Figure 15: Framework of methodology

The left side in Figure 15 is to extract feature. Input any size of original image and get the raw feature from the last convolution layer of the pretrained model. Here I use a different pretrained model from the model in Faster R-CNN based on my training set, so the extracted features are totally different. After that, use Region Proposal Network to get the region proposals over the feature map. Since pedestrians have special shapes than the general objects, I change the scales of the anchor boxes that can fit the shape of pedestrians better. Basically, this architecture is implemented with a convolution and a Rectified Linear Unit layer. The third part is used for region classification and bounding box regression. This part will be trained and the parameters will learn to get a good performance. In order to get a higher efficiency and train the parameters better, I add two more layer modules and the output is fed to two sibling fully-connected layers: region classification layer and bounding box regression layer. For classification layer, it will predict the probability of each anchor box including a pedestrian. For bounding box regression layer, it will predict four offset coordinates for each anchor box. Through the backpropagation, the performance will be optimized via calculating the gradient of the loss function, minimizing the loss and adjusting the parameters, like the weights and bias.

Therefore, this methodology can also implement the pedestrian detections by using the purely convolution neural network, rather than combine with other hand-crafted methods.

4.11 Feature Extraction

For feature extraction, use pretrained models is an efficient way since these models have been trained on very large dataset and we can get a good starting point, saving more

effort than training from scratch based on much less dataset. Therefore, select a good pretrained model is very important. In my methodology, I choose the Alexnet model, which has been trained under the ImageNet, with 15 million annotated images, over 22,000 categories. As we can see in Figure 16, it has 5 sharable convolutional layers, max-pooling layers, dropout layers and 3 fully connected layers. For the first convolution layer, applies 96 kernels with size of 11x 11 over the original image and then use pooling and local response normalization layers. The Relu layer is used for the nonlinearity function. Actually, it does a max operation, which is faster than other activations like tanh activation, improving the efficiency of training. After that, 256 filters with size of 5 x 5 are applied over 96 feature maps, still followed by the pooling and local normalization unit layers. Then applies 384 kernels with size of 3 x 3 over 256 feature maps for the later three convolution layers and do the pooling layers. Finally, three fully connected layers are applied.

The reason to choose Alexnet model as the pretrained model is that the number of layers is suitable for the training data that I choose. The number of images in my training dataset is not large enough, once other complex model with more layers like VGG16, it probably causes overfitting. In addition, Alexnet model has been trained with many images from multiple object categories, including pedestrians.

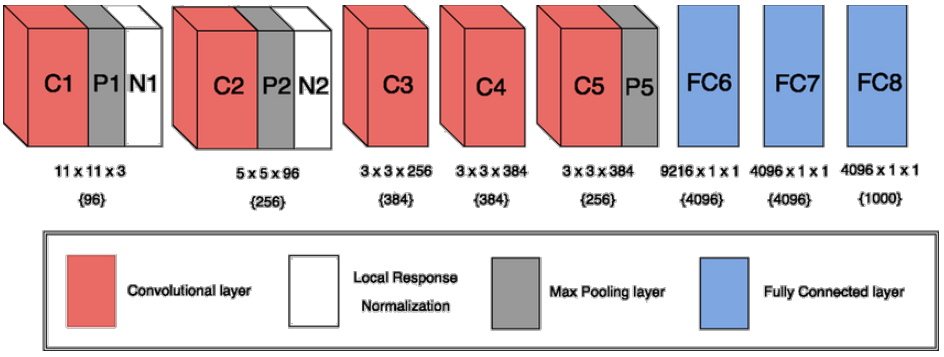

Figure 16: Alexnet model [18]

4.12 Region Proposal

After getting the feature map from the last convolution layer of the pretrained model, generating region proposals can be implemented over this feature map. Slide a 3 x3 window over the feature map and each sliding window is mapped to a 256-dimensional feature. Instead of directly feeding the feature to the two sibling fully-connected layer, these features are fed into the following two convolutions and Relu layers and then the output of Relu layers will be fed into classification layers and bounding box regression layer by the end.

At each sliding window, k region proposals are predicted. So for the classification layer, it outputs 2k scores, indicating the probability of object or not object in each region proposal. For the regression layer, there are 4k output since each region proposal is associated with four coordinates. According to the general shapes of pedestrians, here k is set to be 21. Heights are set as 45, 67.5, 101, 151, 227, 410, 738 and the ratio of width to height is 0.25, 0.41 and 0.6, which generates 21 anchor boxes for each sliding window.

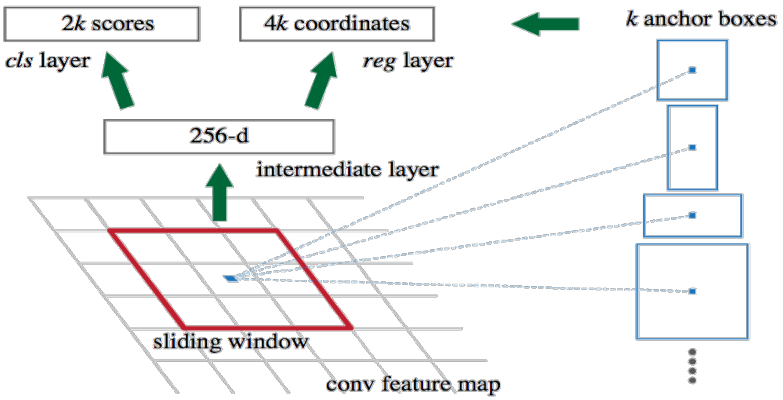

Figure 17: RPN in Faster R-CNN [16]

The way to get the corresponding region proposals on the original image is the following. First, find the center point of the sliding window and map to the point on the original image. Based on the point on the original image, draw the corresponding different scales of the region proposals. So on the right side of Figure 17, these anchor boxes are on the original image. This anchor-based method is built on a pyramid of anchors, which is more efficient than the traditional methods, like the way based on image or feature pyramids. The way based on the image or feature pyramids has to resize the images or features at different scales, useful but time-consuming. The anchor-based method only relies on the single scale of the image and uses sliding window of a single size.

4.13 Region classification and bounding box regression

Region classification and bounding box regression are implemented by the last two layers: classification and bounding box regression layer respectively. Based on the given

labels, the parameters in the convolution neural networks can learn by themselves to minimize the loss function between the predicted value and true values.

4.2 Deep learning model

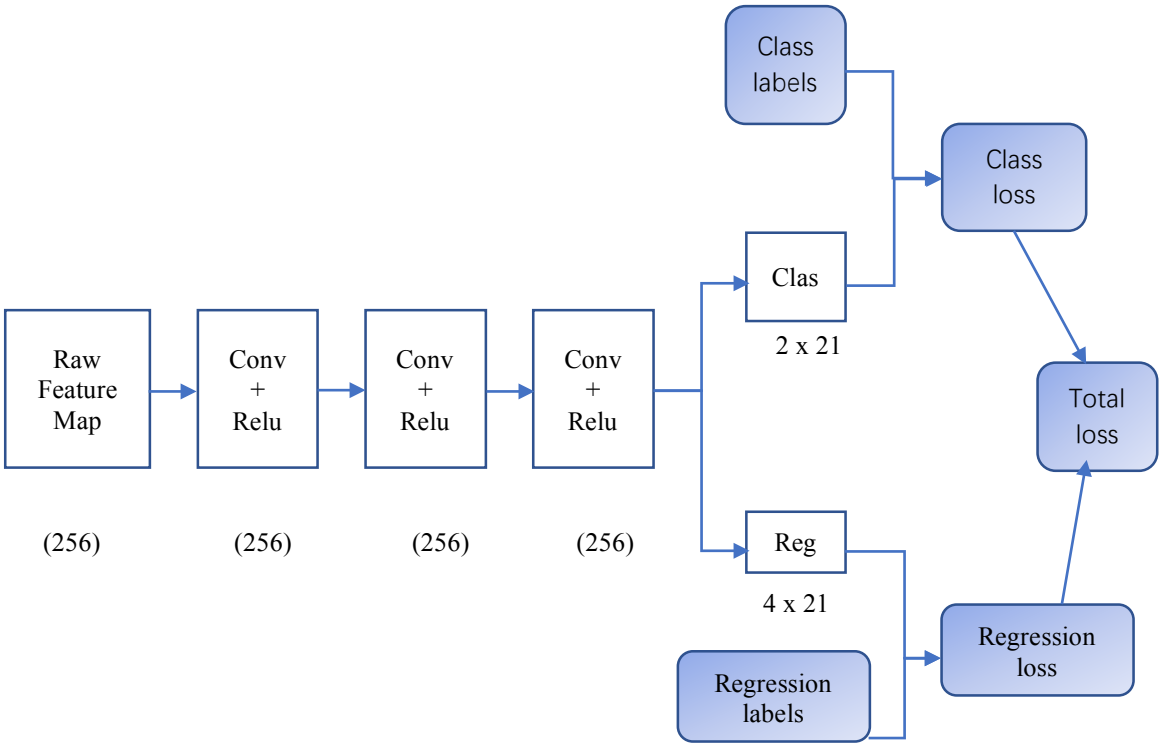

Figure 18: Training model

Figure 18 illustrates the diagram of training model. I trained the training dataset and test dataset based on this model. The input of the model is the raw feature maps that extracted by the pretrained Alexnet model. Actually, we get 256 feature maps totally, and these feature maps are fed to the following three convolution and Relu layers. Each

convolution layer has the same kernel size which is 3 x3 on each feature map. Finally, there are two fully-connected layers with two outputs from this model, one is the score that indicates the probability of the anchor box has pedestrians or not. Since we have 21 anchor boxes for each sliding window and each one has two scores, the number of score output is 21 x 2 which is 42. The other one is regression layer which has 4 x 21 outputs, encoding the coordinates of 21 anchor boxes. By the end, we need to calculate the total loss for these two outputs, using the backpropagation to adjust the weight of each neuron and making the loss function become minimum.

4.21 Loss function

As for the total loss function, it is composed with two parts: the loss function for the classification plus the loss function for the bounding box regression.

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \quad [16]$$

When training RPN, the anchor box is assigned as positive or negative label based on the IOU overlap with the ground-truth box. When the anchor box is positive, the ground-truth label p_i^* is 1, where i is the index of the anchor box, and p_i^* is 0 when the anchor box is negative. p_i is predicted probability for the i_{th} anchor box including a pedestrian. t_i is a vector with predicted parameterized coordinates for the predicted anchor boxes while t_i^* is the ground-truth label for the positive anchor box.

For classification loss L_{cls}, it is the log loss over the two classes, pedestrian or not pedestrian. For the regression layer, L_{reg} is the loss function.

$$L_{reg}(t_i,\ t_i^*) = \ smooth_{L1}(t_i - \ t_i^*) = \begin{cases} 0.5x^2, & if\ |x| < 1 \\ |x| - 0.5, & otherwise \end{cases}$$

Figure 19. Smooth L1 loss function

The reason to choose the smooth L1 loss is that it is less sensitive to outliers, than other loss functions like L2 loss. In Figure 19, we can see that the transition point changes from quadratic function to linear function is 0. In addition, the regression loss is only applied for the positive anchor box since when p_i^* is zero, the loss will be zero.

After calculating classification loss and bounding box regression loss for each anchor box, we need to sum them up separately and normalize these two terms by N_{reg} and N_{clas}, where N_{clas} is the number of anchor box of each mini-batch that use for training and N_{reg} is the number of positive anchor box. Finally, use a parameter λ to make these two losses balance.

4.22 Labels

As we discuss the loss function before, labels for the classification layer and bounding box regression need to be defined.

When define the label of the anchor box, we need to calculate the IOU overlap with the ground-truth box. In the definition of IOU, A represents the area of anchor box and G represents the area of ground-truth box.

$$IOU = \frac{A \cap G}{A \cup G}$$

If anchor box has an IOU overlap higher than 0.7 or has the highest overlap with ground-truth box, we assign a positive label to this anchor box. One thing needs to be noticed is that a ground-truth box may assign positive labels to multiple anchor boxes. If anchor box has an IOU overlap which is less than 0.3, the negative label is assigned to this anchor box. Those anchor boxes have the IOU overlap between 0.3 and 0.7 are ignored and are not contributed to the training.

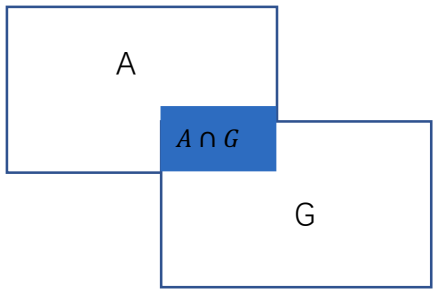

Figure 20: IOU overlap between anchor box and ground-truth box, A represents anchor box and G represents ground-truth box

When define the labels for bounding box regression, we use the parametrization of the coordinates. $(x_a,\ y_a)$, w_a, w_h represent the center coordinates, width and height of the anchor box respectively while $(x^*,\ y^*)$, w^*, h^* represents the center coordinates, width and heights of the ground-truth box respectively. So we adopt t_x^* , t_y^* , t_w^* and

t_h^* as our labels for each positive anchor box when calculating the loss function for the bounding box regression. The formulas for the parameterizations of coordinates are shown below.

$$t_x^* = (x^* - x_a) / w_a$$

$$t_y^* = (y^* - y_a) / w_h$$

$$t_w^* = \log(\frac{w^*}{w_a})$$

$$t_h^* = \log(\frac{h^*}{w_h})$$

reference: from [16]

Chapter 5 Implementation

5.1 Deep learning framework and dataset

Tensorflow is an open-source software library for machine intelligence. Due to its well-organized interface and easy implementation in python, I choose it as my deep learning framework. In addition, it has better computational graph visualizations than other framework, like caffe or torch. Also, it could be trained on big GPU clusters, even on the TPU recently. In Tensorflow, it uses data flow graphs, where the node represents a computation and edge represents the flow of computation, from one node to another in tensor form.

The dataset I choose to train is INRIA Dataset. Different from other datasets which are collected from the videos, INRIA Dataset has the images that are collected from different backgrounds, including the cities, beaches and mountains. It has high quality annotations of pedestrians. The diverse setting has great advantage in training.

In INRIA Dataset, it has training dataset, validation dataset and annotations. In the training dataset, there are 1828 images, where 1214 of them are positive images, including pedestrians and 614 are the negative images, without any pedestrians. In the validation dataset, it includes 741 images, 288 of them are positive and 453 of them are negative images. In annotations, the center point coordinates and the width as well as heights of ground-truth regions are provided for both training data and validation data.

Besides the training dataset and validation set, testing dataset is also needed. I select 457 images from the validation dataset as validation part, including 250 positive images and 206 negative images, and the rest of them are used for testing, which includes 285

images, 38 for positive images and 247 for negative images. Training dataset are totally used for training.

5.2 Data Training

When train the model in tensorflow, we need to choose the proper optimizer. Firstly, I tried to use the simplest one: Gradient Descent Optimizer and let the learning rate stay the same all the time, I found that the process of training is slower and the training loss cannot decrease a lot. The fixed learning rate is one reason for the poor performance, since some parts of weight matrix may change fast, but others stay as constant. Getting stuck in local minimum is probably another weakness. After that, I tended to select the Momentum Optimizer, which can avoid the gradients stuck in local minimum and can accelerate the training. However, it is too fast for training set to learn, causing the training loss becomes infinite.

After several trials, I adopt the AdamOptimizer, since it brings better performance than others on the training loss. It makes each weight have different learning rate and an exponentially decay average of the previous gradients. In addition, it does not need to keep a history of anything, excepts the rolling average, which make the memory efficient. The starting learning rate of the AdamOptimizer is set to be 0.0001, too large or too small cannot converge the training loss.

For training, 1828 images from training set in INRIA dataset are selected to be trained and each image are trained one by one, which means for one iteration, each batch of training is one image. The epoch is set to be 70 and each epoch includes the whole training set, which includes 1828 images. When selecting the anchor boxes for computing

the loss function, 1200 negative samples will be randomly selected from each negative image. For each positive image, the number of sampled positive anchor boxes is equal to the number of sampled negative anchor boxes, and the number of sampled positive samples are all the positive anchor boxes in the images.

5.3 Data Validation

To avoid underfitting or overfitting when training images, we use cross validation to keep tracking the training process. When the model fits the training data very well but does not fit the validation data, it is overfitting. When the model performs poorly on the training data, it is underfitting since it cannot capture the relationship between the input data and the target values. Figure 21 shows each of them in diagram. In Figure 22, we can see that the test error will decrease for a while and then will increase. To avoid the overfitting, we need to stop training at the point when the test error has the tendency to increase.

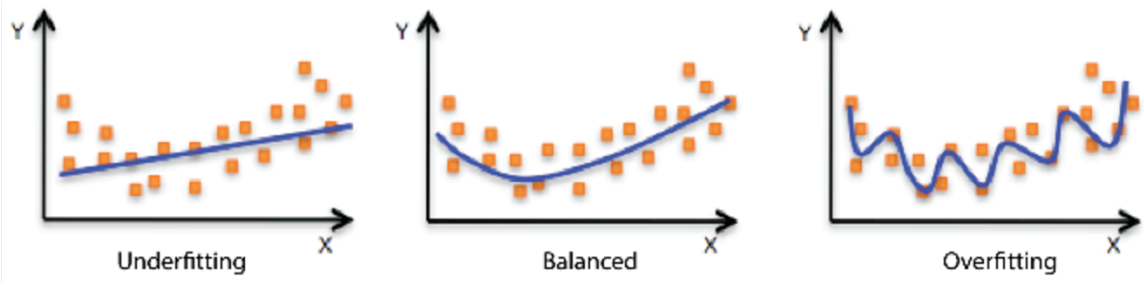

Figure 21: Underfitting, balanced and overfitting of the model [19]

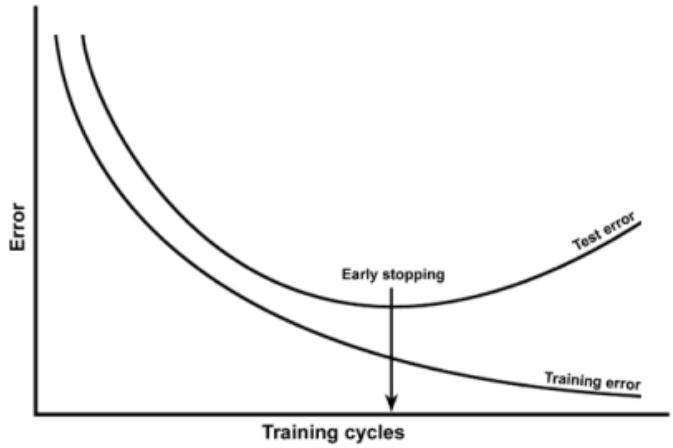

Figure 22: The process of training data and validation data

As discussed before, 456 images are selected from the 741 images in validation dataset to use as validation data. It includes 250 positive images and 206 negative images. The target selections for validation data is the same as training data when computing the loss function, which the ratio of the sampled positive and negative is 1 for each positive image and 1200 negative samples for each negative image. For the validation steps, one image will be validated after training every four images.

5.4 Data Testing

Testing data is composed with two part: one from the validation data, including 38 positive images and 247 negative images, the others are the pictures that we collected from daily life.

To test how the model works, I let the predicted bounding boxes show on the image, which can compare with the ground-truth boxes. Firstly, we need to know the center coordinates and the height as well as the width of the predicted bounding box. The output

of the bounding box regression layer is the parameterization coordinates (t_x, t_y, t_w, t_h), and according to the following formula, the needed information (x, y), w and h can be calculated, which represents the center pointer, width and height of the predicted bounding box. x_a,y_a, w_a, and h_a presents the coordinates of the anchor box associated with the predicted position.

$$\boldsymbol{t_x = (x - x_a)/w_a} \qquad \boldsymbol{t_y = (y - y_a)/h_a} \qquad \boldsymbol{t_w = log(\frac{w}{w_a})} \qquad \boldsymbol{t_h = log(\frac{h}{h_a})}$$

When showing the proposals, one pedestrian probably has several bounding boxes with corresponding probability due to the method of sliding window and there is no necessary to keep all of them. So, I adopt Non Maximum Suppression(NMS) to keep the most optimized bounding box for each pedestrian. Firstly, sort all the bounding boxes by the probabilities that includes pedestrians and select the bounding box with the highest score. Secondly, calculate the overlap between the selected one and its surrounding bounding boxes and delete the bounding boxes which have exceed the overlap threshold. Repeat the process until the last bounding box. In my own implementation, I set the threshold of the overlap to be zero, which keeps only one bounding box for each possible pedestrian. Figure 23 shows the sample results between the original images and the images after NMS.

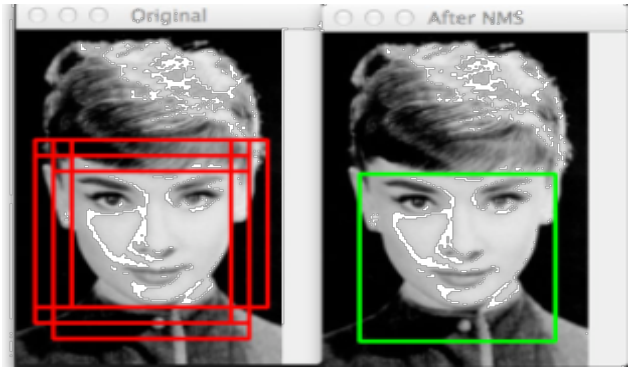

Figure 23: samples [20]

5.5 Evaluation methods

When evaluating the result, true positive, false positive, true negative and false negative for each image are recorded. The relationship between these four terms are illustrated in Figure 24. If both the predict outcome and the true value are positive, we can consider the output is true positive. If the predict value is the same negative as the actual value, the result is true negative. It is false positive when the predict output value is positive but the actual value is negative and the predict outcome is false negative when the predict outcome is negative but the true value is positive.

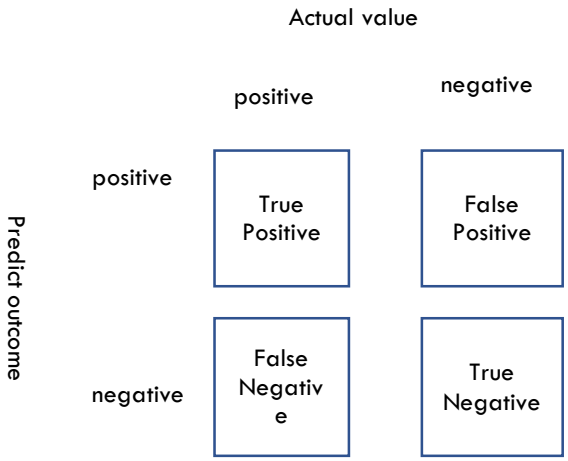

Figure 24. Definition for true positive, false positive, false negative and true negative

Instead of simply calculating the accuracy of the result, I choose to use precision, recall, F1 score and miss rate as the evaluation. Since the number of negative samples is much larger than the number of positive samples, there is no more sense to calculate the accuracy.

Precision is the number of true positives divided by all the selected elements. In pedestrian detection, precision is the fraction of all the predicted pedestrians that are predicted correctly. The formula of precision is:

$$\text{Precision} = \text{True positive} / (\text{True positive} + \text{False positive})$$

For recall, it is the number of true positives over the total number of positives that should be selected. In pedestrian detection, recall is the fraction of all the pedestrians that are successfully predicted. The formula of recall is:

$$\text{Recall} = \text{True positive} / (\text{True positive} + \text{False negative})$$

F1 Score is a method that consider both the precision and recall and find the average weight between these two values to reach a balance, making the contribution of both precision and recall to F1 score equal. It can reach the best value at 1 and the worst value at 0. In pedestrian detection, we mainly focus on this score. The formula of F1 Score is:

$$\text{F1 Score} = 2 * \text{Precision} * \text{Recall} / (\text{Precision} + \text{Recall})$$

For miss rate, it is also the false negative rate and the value of miss rate and recall equals to one. The formula of miss rate is:

$$\text{Miss Rate} = 1 - \text{Recall}$$

In Figure 25, the diagram shows the definition of precision and recall.

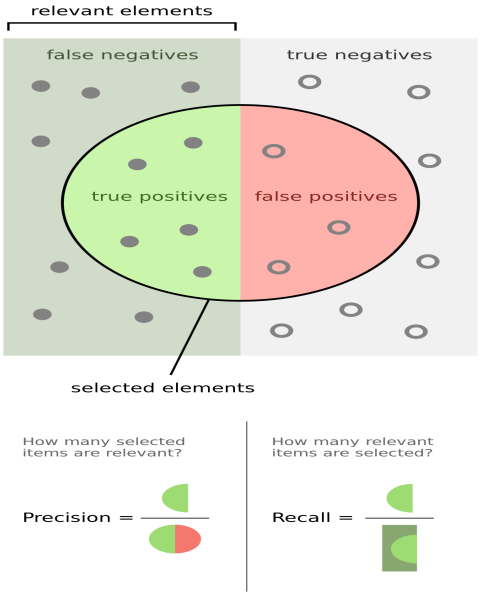

Figure 25. Evaluation Diagram from Wikipedia.

Chapter 6 Result Analysis

6.1 Training Loss and Validation Loss

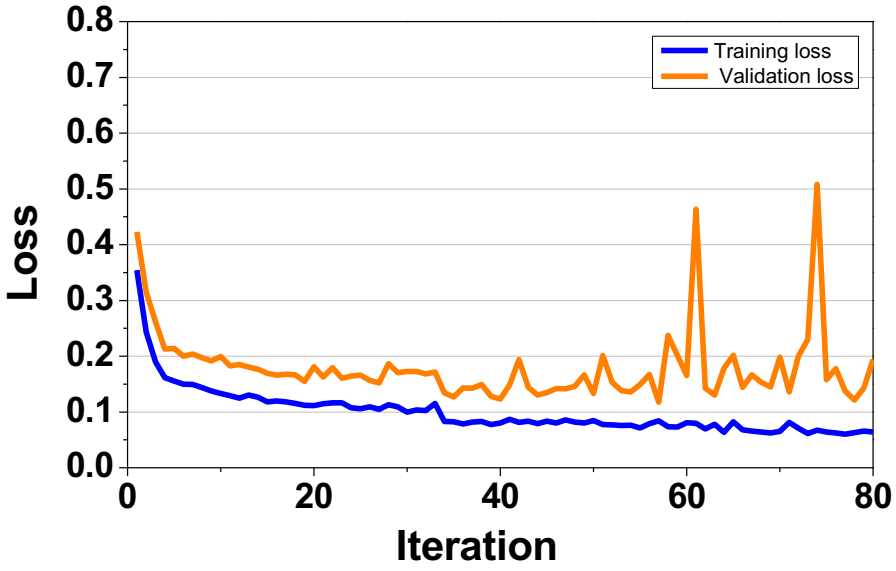

Figure 26. Comparison between training loss and validation loss

In Figure 26, it shows how the loss changes at each iteration in both training data and validation data. The x-axis is the number of iterations and y-axis is the loss value. Each point in the diagram represents the average loss at each iteration. There are 80 iterations in total and every one image is validated when four images are trained. For training loss, it decreases as the number of iterations increases and reach at around 0.06 at the last iteration. In general, the loss of validation data also decreases when the iteration becomes larger, but some parts have sudden changes. The reason is that during these iterations, the model cannot fit the validation data well at some specific steps, causing the average loss

becomes relatively larger than the surrounding points. But the loss still can decrease to the low value after these sudden changes and reach the lowest value at around 0.15. I stop the iteration at 80 because when I continue adding the number of iteration, the loss of validation data becomes larger and more points of the loss have sudden increase, it tends to become overfitting for the model.

6.2 Evaluation Results

For the evaluation of experiments, I mainly compare the precision, recall, F1 score and miss rate between the training data and validation data. And then test the images separately on these evaluations. The training data includes 1828 images, while the validation data includes 457 images. The number of images for testing is 38.

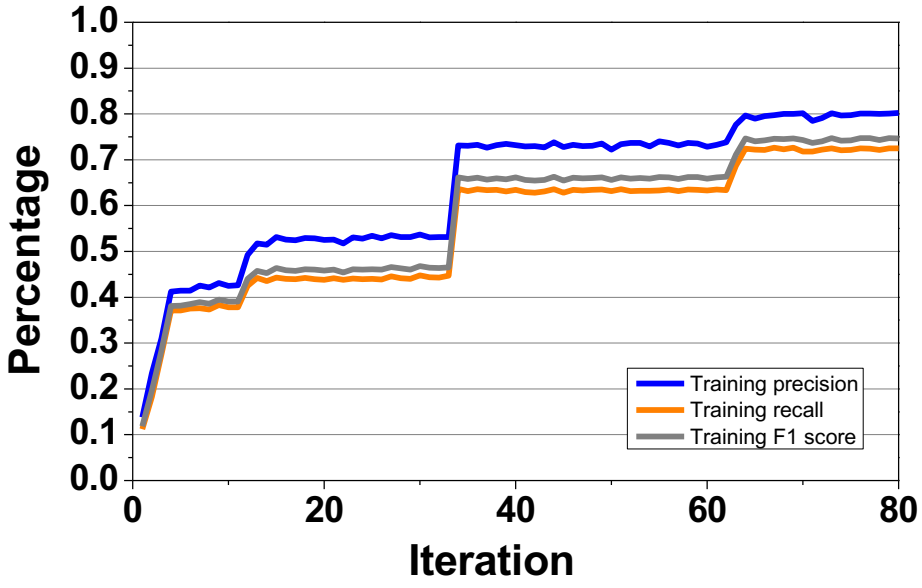

Figure 27. Training precision, recall and F1 score

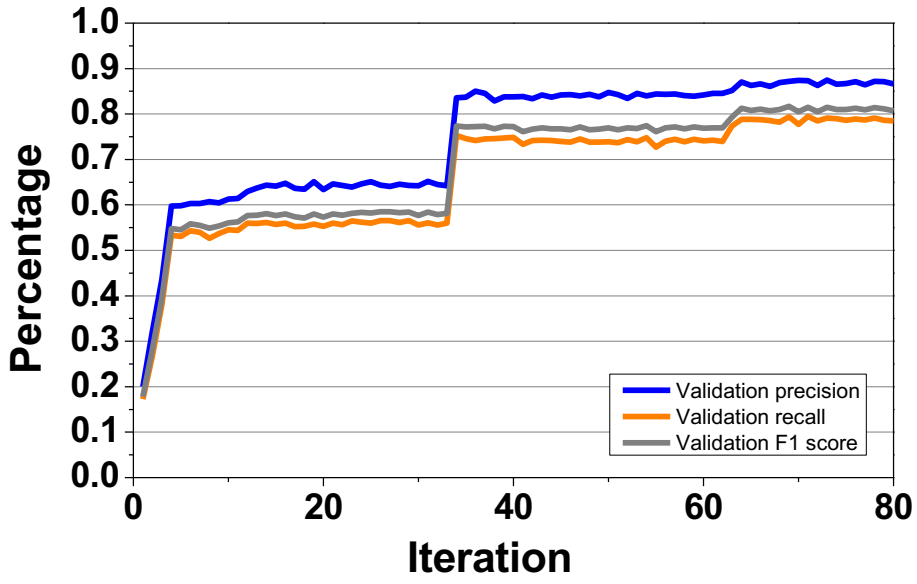

Figure 28. The precision, recall and F1 Score of validation data

The precision, recall and F1 score of training data and validation data are shown in Figure 27 and Figure 28 respectively. For training and validation data, they have the similar tendency on these three evaluations. They both have significant increase at iteration 4, 33 and 65. The F1 score of training data can reach at around 72 percent while for the validation data, it can achieve at 81 percent.

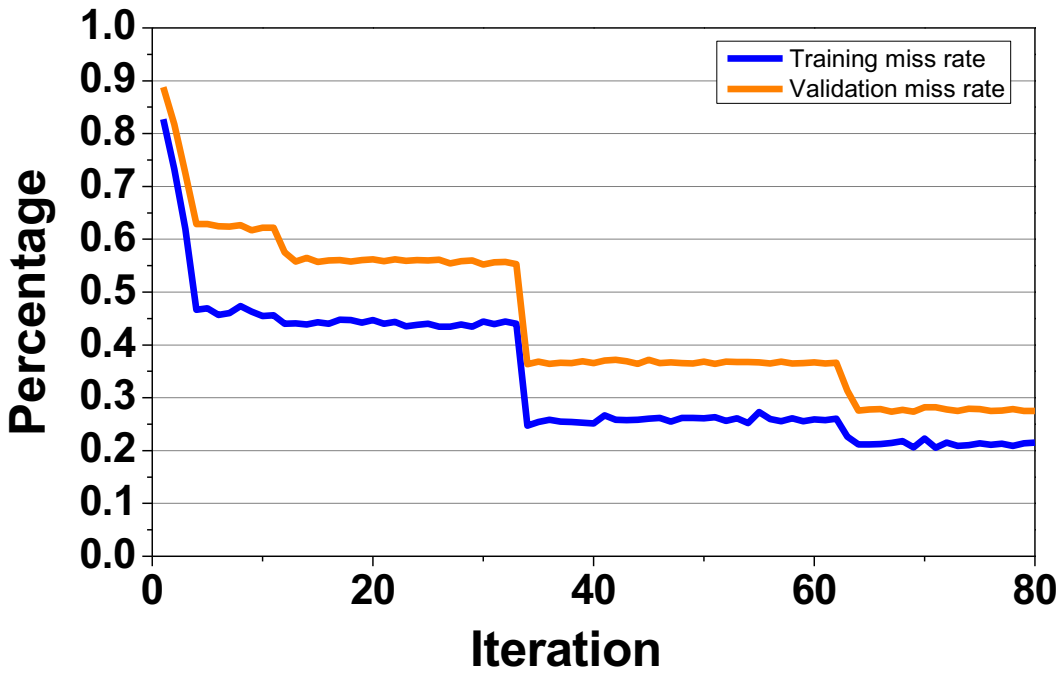

Figure 29. Miss rate of training and validation data

In Figure 29, it shows the miss rate of training data and validation data. It is obvious to notice that the tendency of training miss rate and validation miss rate is also similar. At around 4^{th} , 33^{rd} and 65^{th} iteration, the miss rate has significantly decreased, which can correspond to value of recall in Figure 27. For the training data, miss rate decreases from 0.9 to 0.28 while the miss rate of validation data decreases to 0.2.

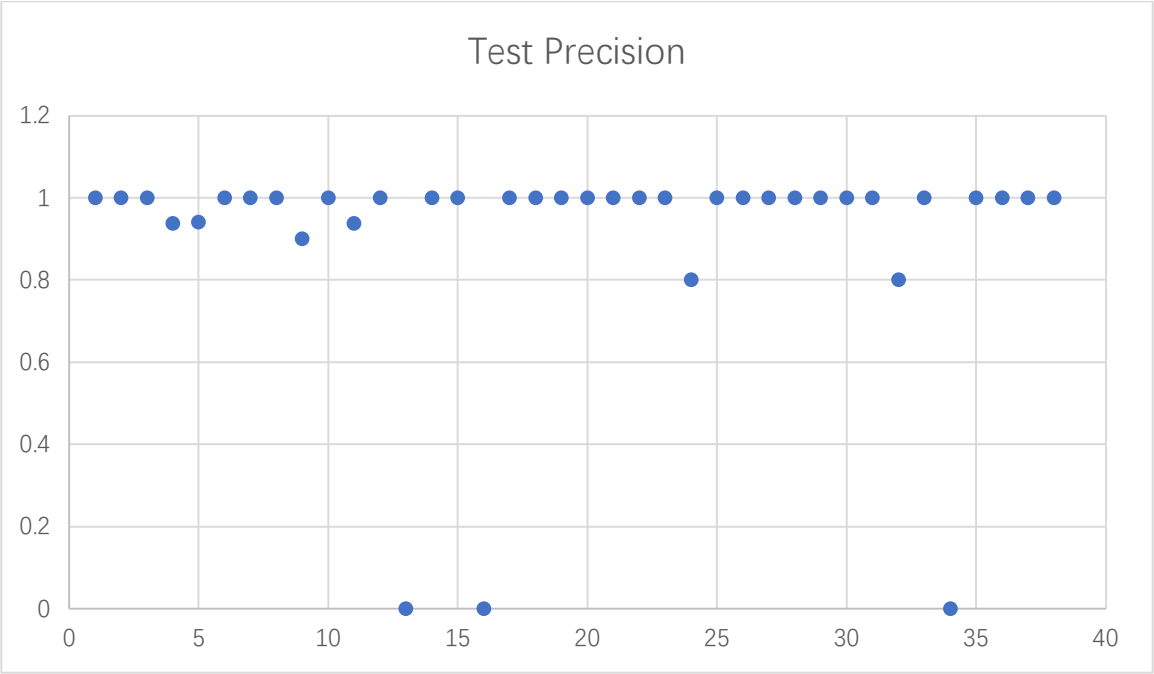

Figure 30. Precision of 38 testing images

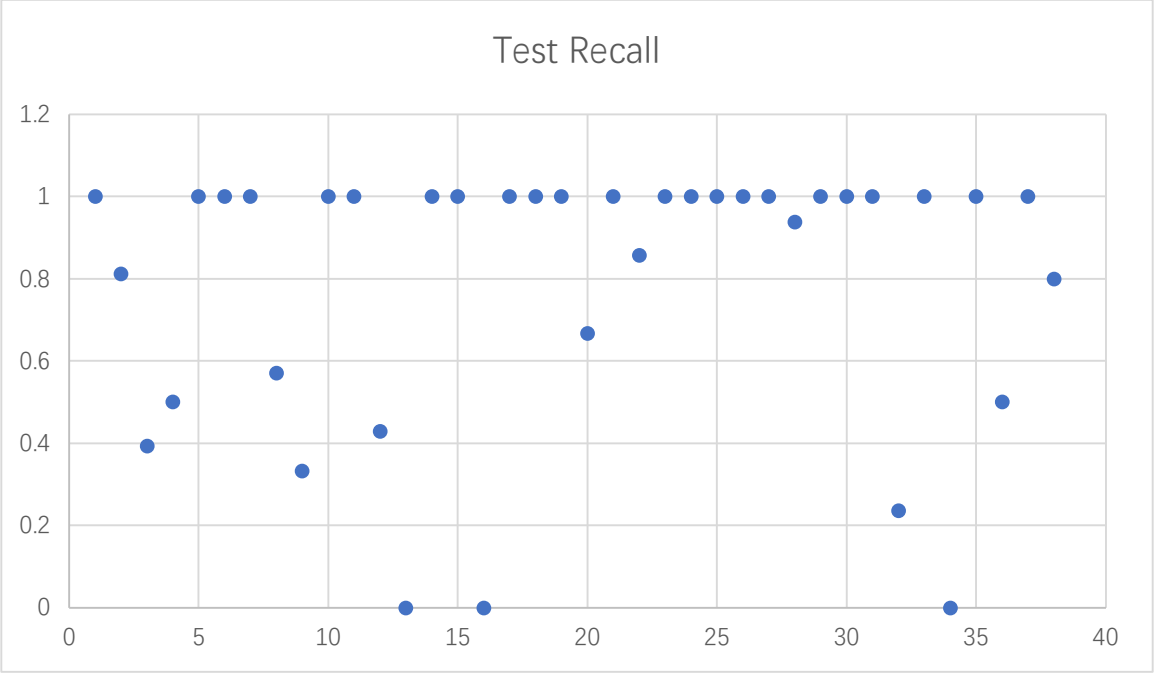

Figure 31. Recall of 38 testing images

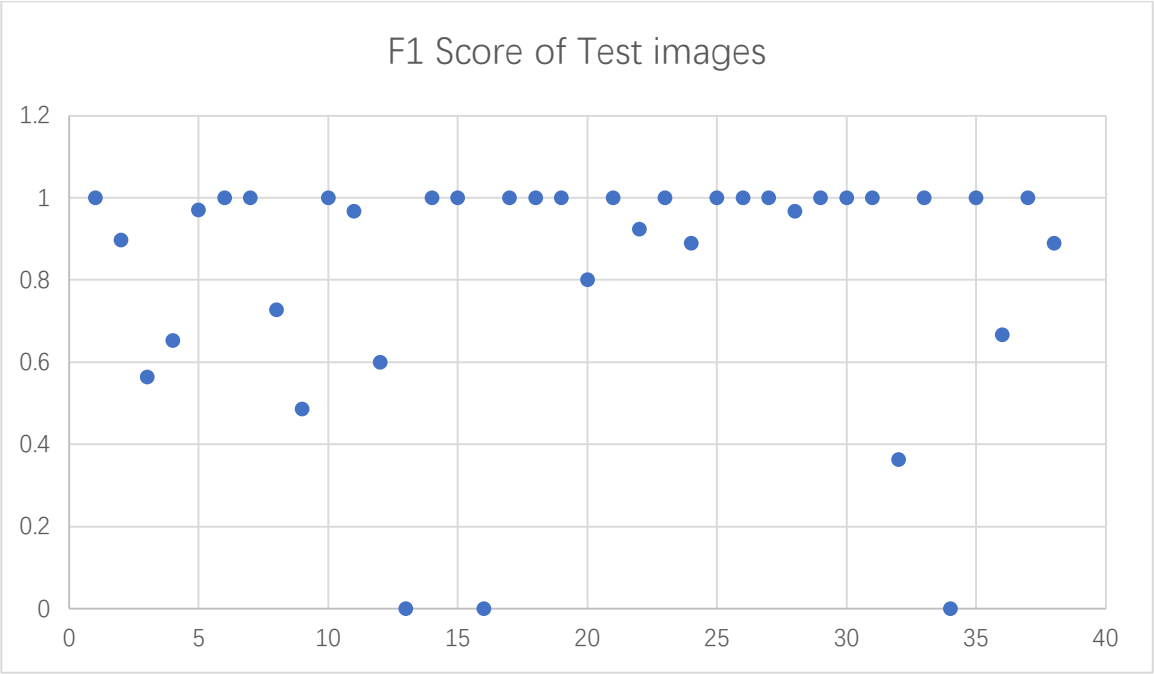

Figure 32. F1 Score of testing images.

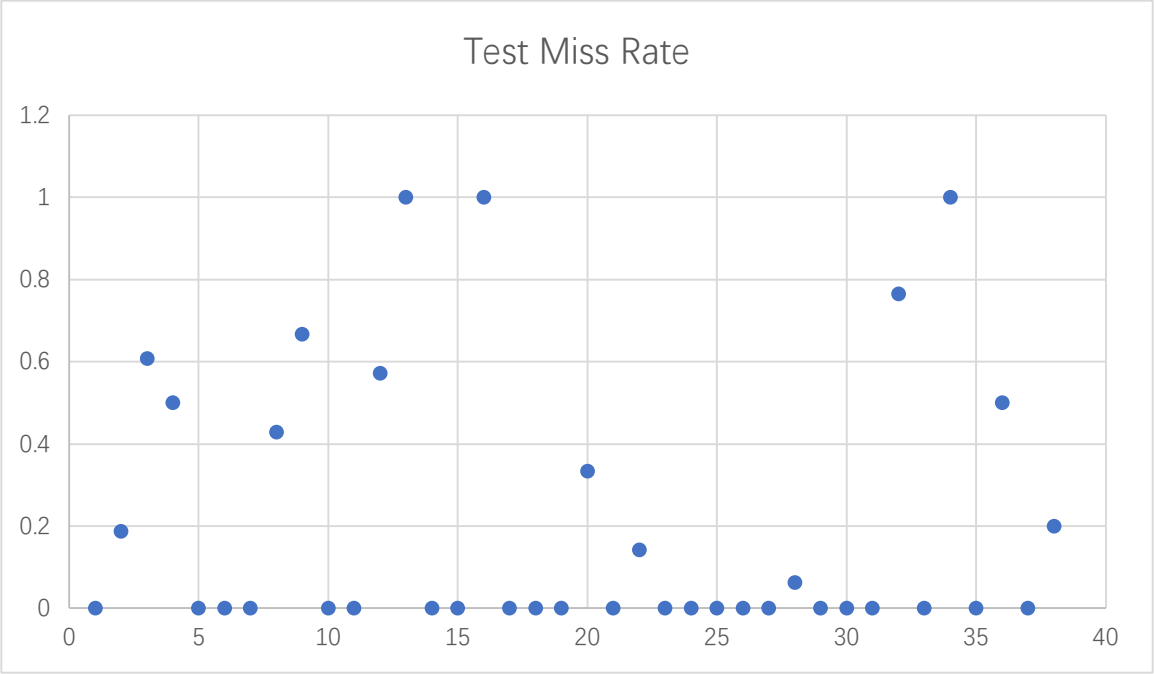

Figure 33. Miss rate of Testing images.

Figure 30 to Figure 33 are the evaluation results for testing images. There are 38 test images in total and the precision, recall, F1 score and miss rate for each image are recorded. In Figure 30, it shows the distribution of precision of all the test images. We can notice that most of the images can reach 100 percent precision, which means all the predicted bounding box are true positives in these test images. The average precision is 90 percent. However, according to Figure 31, The average recall is 0.79, which is much lower than the precision. The reason is that some test images have high precision but lower recall rate, indicating that there are some pedestrians which should be predicted are not be shown with the bounding boxes. Combine with the precision and recall, the F1 score of these test images can reach at around 83 percent. Figure 33 shows the distribution of the miss rate for testing data and the average miss rate is around 0.2.

From the results shown above, we can see that it still exists some limitations about the accuracy. There are several reasons that I think plays important role in the results. Firstly, training data is not large enough that the CNN model cannot have much larger features of pedestrians to learn. Secondly, due to the number of training dataset is relatively small, the pretrained model Alexnet model is selected which are relatively simpler than other models and has fewer layers. It also limits the learning accuracy. Thirdly, instead of fine-tuning the layers from pretrained model, I directly use the feature map from the last convolution layer from Alexnet model without training, since I just can train these images from my laptop which only has CPU.

6.3 Comparison with Faster R-CNN

In [17], the author proposed that Faster R-CNN degrades the accuracy of detecting pedestrians. When feeding the proposals into Fast R-CNN classifier after getting the proposals from RPN, the convolutional feature maps of Fast R-CNN classifier have low solutions for such small objects. And typically, the scenarios of pedestrian detections present small size of pedestrians, causing the performance of detection is degraded.

6.4 Example of Pedestrian Detection Results

In order to observe directly, I choose some images from validation data and show the predicted bounding boxes and ground-truth boxes for each image. The blue one represents the predicted bounding box associated with predicted probability of the pedestrians while the red one represents the ground-truth bounding box of pedestrians. These images are selected according to their various background as well as the number of pedestrians.

Figure 34. Images from validation data.

Besides the images from INRIA dataset, I also choose some pictures that are photographed by ourselves. The backgrounds of these images are roads and beaches at San Diego. The value associated with the blue bounding box is the predicted value of

whether the object is pedestrian. In Figure 35, We can see that the model can predict different sizes of bounding boxes according to the various shapes of pedestrians and predict correctly where the pedestrians are.

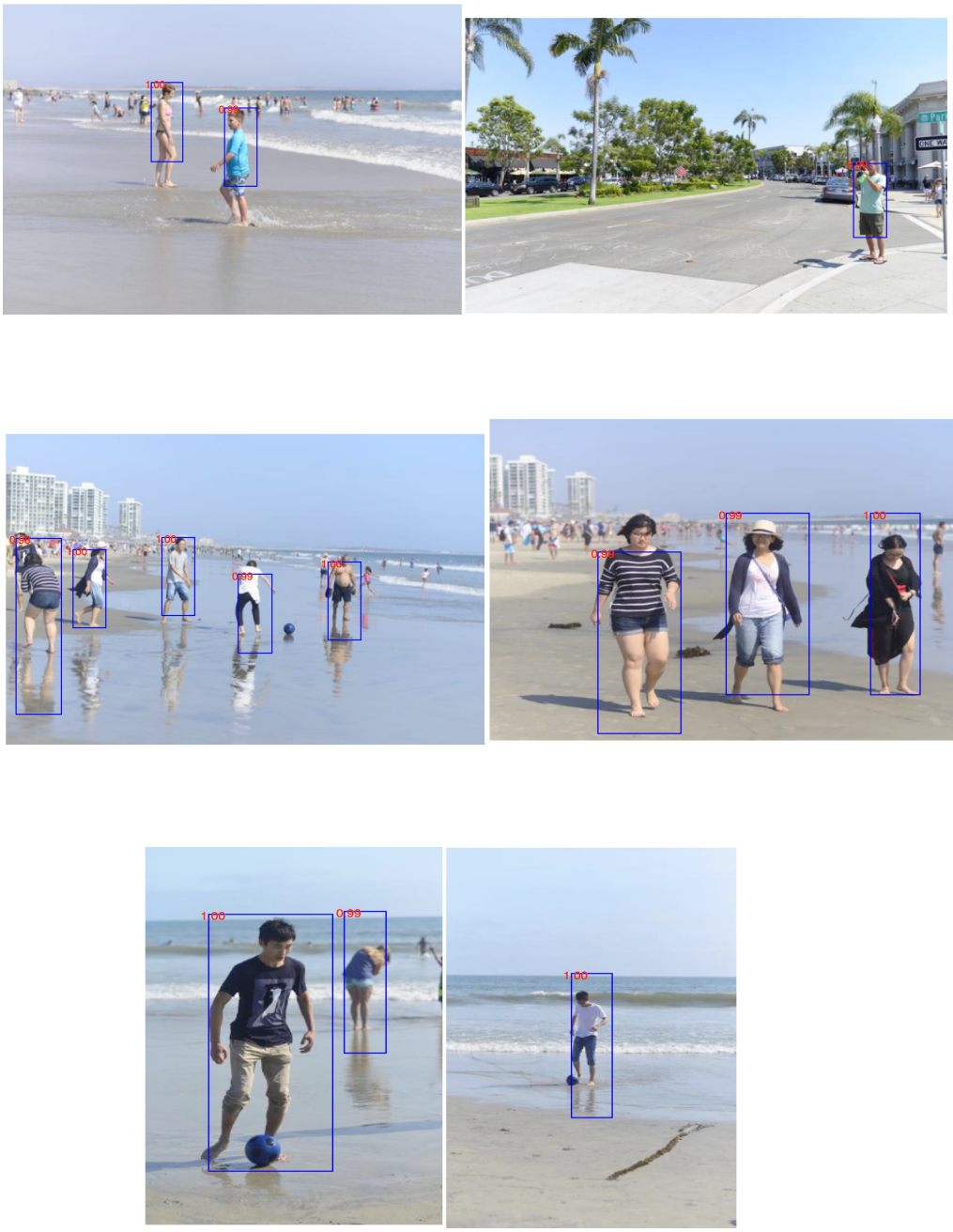

Figure 35. Images from daily life

Chapter 7 Conclusion

The methodology based on Faster R-CNN which leads high accuracy on Object Detection can reach relatively high accuracy. Region Proposal Network, a critical part of Faster R-CNN has been utilized and modified in detecting the pedestrians due to the specific characteristics of pedestrians. Getting the feature maps from pretrained Alexnet model and training in the convolution neural network, we can get both the predicted probability of whether the object is pedestrian and the bounding boxes of the predicted pedestrians.

In order to increase a higher accuracy, there are still some future work to do. Firstly, the training dataset need to be extended, for example, combine the Caltech-USA dataset with INRIA dataset. Secondly, the complexity of the training model should be increased. More layers of the model are needed which can help the training model learn more features from the images. Lastly, since the dataset are trained on CPU with my laptop, the device limits a lot of work. It should be better to fine-tune all the layers of the model, instead of directly using the feature maps from the pretrained model and just training parts of the new layers.

Chapter 8 Reference

[1] Pedestrian Traffic Fatalities by State 2016 preliminary data, prepared for Governors Highway Safety Association by Richard Retting Sam Schwartz Transportation Consultants.

[2] Ren, Shaoqing, et al. "Faster R-CNN: Towards real-time object detection with region proposal networks." *Advances in neural information processing systems*. 2015.

[3] Zitnick, C. Lawrence, and Piotr Dollár. "Edge boxes: Locating object proposals from edges." *European Conference on Computer Vision*. Springer, Cham, 2014.

[4] Doll´ar, P., Zitnick, C.L.: Structured forests for fast edge detection. In: ICCV. (2013)

[5] Doll´ar, P., Zitnick, C.L.: Fast edge detection using structured forests. CoRR abs/1406.5549 (2014)

[6] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *IJCV*, 2013

[7] Ling, Christopher. "Image Detection Techniques on Daimler Pedestrian Monocular Data."

[8] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE, 2005.

[9] Dollár, Piotr, et al. "Fast feature pyramids for object detection." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.8 (2014): 1532-1545.

[10] Blog 'Quick Introduction to Boosting Algorithms in Machine Learning' by SUNIL, RAY

[11] Understanding Support Vector Machine algorithm from examples by Sunil Ray

[12] Fukui, Hiroshi, et al. "Pedestrian detection based on deep convolutional neural network with ensemble inference network." *Intelligent Vehicles Symposium (IV), 2015 IEEE*. IEEE, 2015.

[13] Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014.

[14] He, Kaiming, et al. "Spatial pyramid pooling in deep convolutional networks for visual recognition." *IEEE transactions on pattern analysis and machine intelligence* 37.9 (2015): 1904-1916.

[15] Girshick, Ross. "Fast r-cnn." *Proceedings of the IEEE international conference on computer vision*. 2015.

[16] Ren, Shaoqing, et al. "Faster R-CNN: Towards real-time object detection with region proposal networks." *Advances in neural information processing systems*. 2015.

[17] Zhang, Liliang, et al. "Is faster r-cnn doing well for pedestrian detection?." *European Conference on Computer Vision*. Springer International Publishing, 2016.

[18] Srinivas, Suraj, et al. "A taxonomy of deep convolutional neural nets for computer vision." *arXiv preprint arXiv:1601.06615* (2016).

[19] Picture from Amazon Machine Learning documents.

[20] Non-Maximum Suppression for Object Detection in Python by Adrian Rosebrock on November 17, 2014 in Machine Learning, Tutorials.