University of Amsterdam

MSc Artificial Intelligence

Track: Computer Vision

Master Thesis

Pose-RCNN: Joint object detection and pose estimation

by

Yikang Wang

10540288

August 15, 2016

42 EC

September 2015 - July 2016

Academic Supervisor:
Prof. dr. D. M. Gavrila

Local Supervisor:
M.Sc. Fabian Flohr

DAIMLER

Daimler AG
Group Research & MBC Development

Pose-RCNN: Joint object detection and pose estimation

by

Yikang Wang

Abstract

Object detection was seen as a key part for driver assistance systems as well as autonomous cars during the last years. By making use of other information (e.g. pose information) more sophisticated knowledge can be gained. Among all of the object detection, the interaction between the cars and humans is always one of the most important part to study. With having pose information of road users, tracking can be initialized faster and intentions can be analyzed. Therefore we focus our work on road users detection and pose estimation with car-mounted video cameras. In this thesis we propose Pose-RCNN for joint object detection and pose estimation with the following three major contributions. First, we extend the well known Fast R-CNN[1] by a pose layer using Biternion net representation[2] to create a single framework for joint object detection and orientation estimation. Secondly, we propose R-GoogLeNet, which well integrate GoogLeNet[3] into our Pose-RCNN framework, providing more powerful performance. Last, we develop a proposal-box splitting technique for Pose-RCNN, enabling our system to do parts detection and parts pose estimation.

The experiments are conducted using Tsinghua-Daimler Cyclist Benchmark[4] which contains bounding-box and orientation labels of cyclists. The full-object detection and pose estimation performance of different networks are compared and analyzed. The best performance is achieved by our proposed R-GoogLeNet with 0.824 average precision and 0.811 average orientation similarity. The relative small network R-GoogLeNet$_1$ (a substructure of R-GoogLeNet with less layers, see Section 5.5 for detail) is 1.5$\times$ faster than R-GoogLeNet, with only less than 0.01 average precision and average orientation similarity drop. The results of parts detection and orientation estimation are also shown and compared. Based on all the results, we discuss the limitations of the approaches and show the possible directions for further work.

Acknowledgments

It has been a dream of mine since my childhood to work on developing intelligent cars. The dream has finally come true after my coming to Daimler. I would like to thank Prof. Dr. Dariu M. Gavrila and Dr. Kreßel Ulrich for providing me with the chance to work here with a group enthusiastic colleagues. I would express my sincere gratitude to my local supervisor Fabian Flohr for his expert advice and encouragement throughout this thesis project. Whenever I ran into a trouble spot, he consistently helped me and steered me in the right direction. I would like to thank Markus Braun who gave me a lot of help during the work. It has been really great working together with you. I would also like to thank the team I work with in Daimler. It has been a wonderful experience working with such a group of great people.

Besides, my sincere thanks also goes to my lab-mates Sourabh Agrawal, Matteo Bertolucci and Aswin Vijayamohanan Nair. It has been a great time working together and learning from each other. I would also like to thank Mingyu Zhang who provides me with the place of accommodation during the last month of my thesis writing.

Finally, I must express my very profound gratitude to my parents for providing me with unfailing support, continuous encouragement through my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Since the birth of the first modern car in 1886 by German inventor Karl Benz, vehicles has being providing convenience for human life for over an century. At the same time, vehicle accidents gradually became one of the main causes of accidental death in modern civilization. In 2012, road traffic injuries was the leading cause of death among people aged 15-29[5]. In many developing countries the percentage is even higher. The road safety has become a world wide problem. To improve road safety, various of technologies and systems are developed and applied. From the transportation management system's view, intelligence technologies are studied and applied for improving the overall road efficiency, reliability and safety. For example, a intelligent transportation system could collect data from speed cameras, live traffic flow, weather conditions, etc, and then dispatches traffic by controlling traffic signals, variable message signs and so on. Such systems are able to make better use of road resources and thus reduce the risk of accidents and improve the road reliability. From the car perspective, intelligent vehicle systems like advanced driver assistance systems (ADAS) or autonomous cars are developed to automate/adapt/enhance vehicle systems for safety and better driving experience. In our work, we focus on intelligent vehicle systems, or more specifically the road user detection and pose estimation. Based on our work, the behavior and intention of road users could be further analyzed for preventing potential accidents between cars and road users.

1.1 Road Safety

Motor vehicles as one of the key industrial inventions in human development have been manufactured and improved for decades since their birth. In 2015, 68.56 million passenger cars and 22.12 million commercial vehicles were produced[6]. By the end of 2014, the number of worldwide motor vehicles in use is already approaching 1.25 billion[7] and the statics still shows

a growing trend of the total number in the following years. Figure 1-1 shows the number of passenger cars and commercial vehicles in use worldwide from 2006 to 2014.

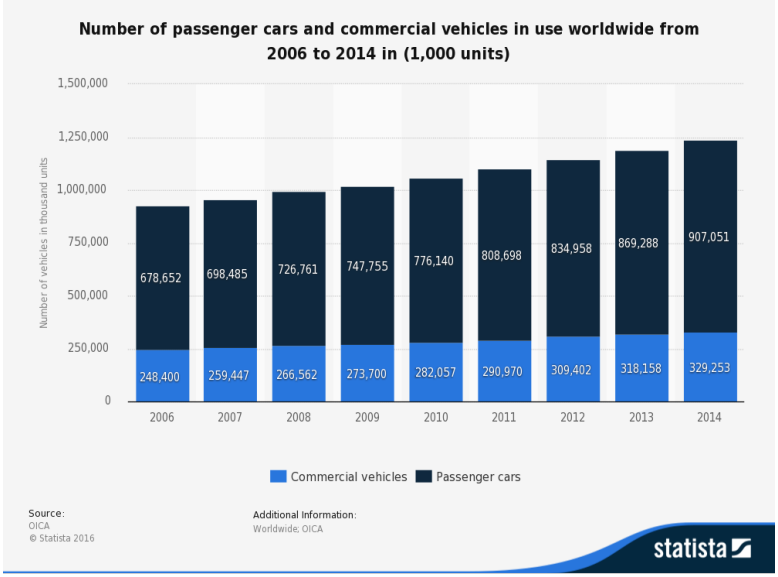


Figure 1-1: Number of passenger cars and commercial vehicles in use worldwide from 2006 to 2014 in (1,000 units)

However with the increasing number of vehicles, the fatalities that caused by traffic accidents also goes up, which reached 1.25 million in 2013 [5]. It means every 25 seconds, a road user will die of an road accident. From the statics of 2012 1-2, road traffic accidents has become the globally leading cause of death especially among those aged 15-29 years. In low-income countries ((e.g., African and Eastern Mediterranean countries) the fatality rates are more than double of those in high-income countries, even though low-income countries have lower level of motorization. The report of World Health Organization[5] shows that 90% of road traffic deaths occur in low- and middle-income countries, yet these countries have just 54% of the worlds' vehicles. The report also reveals that almost half of all deaths on the world's roads are among those with the least protection - motorcyclists (23%), pedestrians (22%) and cyclists (4%). These user types are also often referred as vulnerable road users (VRUs). According to the survey[8], the three key risk factors are speed, drink-driving and distracted driving.

1.2 Intelligent Vehicles

Road traffic injuries can be prevented. On one hand, the interventions that target road user behavior are important, such as setting and enforcing laws relating to key risk factors, raising public awareness to avoid distracted driving and so on. On the other hand, intelligence technologies could be applied for improving the safety features of vehicles. In general, there are

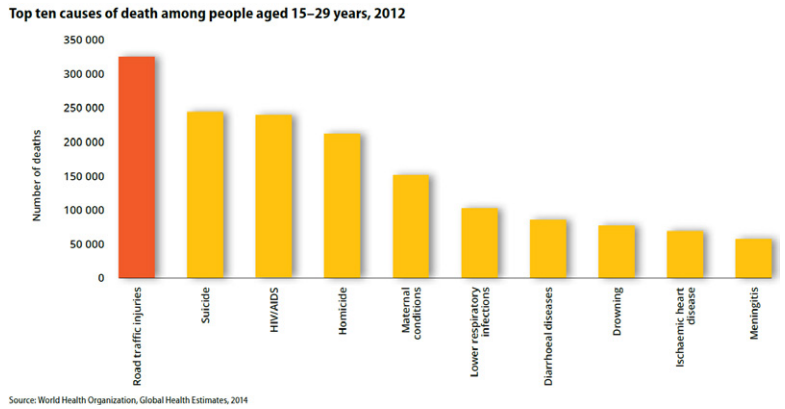


Figure 1-2: Top ten causes of death among people aged 15-29 years, 2012 [5]

two types of fields where intelligence technologies are applied - driver assistance system and fully self-driving system. For driver assistance systems, intelligence technologies include traffic sign recognition, objects detection, adaptive cruise control, pre-crashing braking, automated parking and so on. With some driver assistance systems, the car is capable of self-driving for some moments. However the driver is also expected to take over the control especially when facing situations that the system is unable to handle. For fully self-driving systems, the car is designed to do all the work of driving and to be able to handle any situation that might happen on road. The human 'driver' is never expected to take control of the vehicle at any time. Through history, intelligence technologies used in vehicles have been studied and developed for decades and in recent years more and more companies have started to develop their own intelligent vehicles. However there has not been a fully self-driving car brought to the market yet. Most of the products in-use still remains at the level of driver assistance system. Only a few number of companies ((e.g., Google) are working on fully self-driving. To achieve fully self-driving, there is still a long way to go.

1.3 Pedestrian/Cyclist Detection in Intelligent Vehicles

For both fully self-driving cars and driver assistance systems, the detection of pedestrians/cyclists plays a vital role. In the thesis, we focus our work on pedestrian/cyclist detection for intelligent vehicles. A general detection architecture is illustrated in Figure 1-3. The system is able to detect vulnerable users on road and prevent accidents by warning the driver or triggering autonomous braking or steering. According to the survey of Enzweiler and Gavrila[9], the best pedestrian detection method achieved the precision of 77.2% at 250 ms per frame. During the last decades, the success in the field of pedestrian detection has also leads to market introduction. However different approaches still have their limitations, and none of products

could guarantee 100% precision. The first fatal accident caused by wrong detection of driving assistance system is reported this year [10]. The research in this field is still underway.

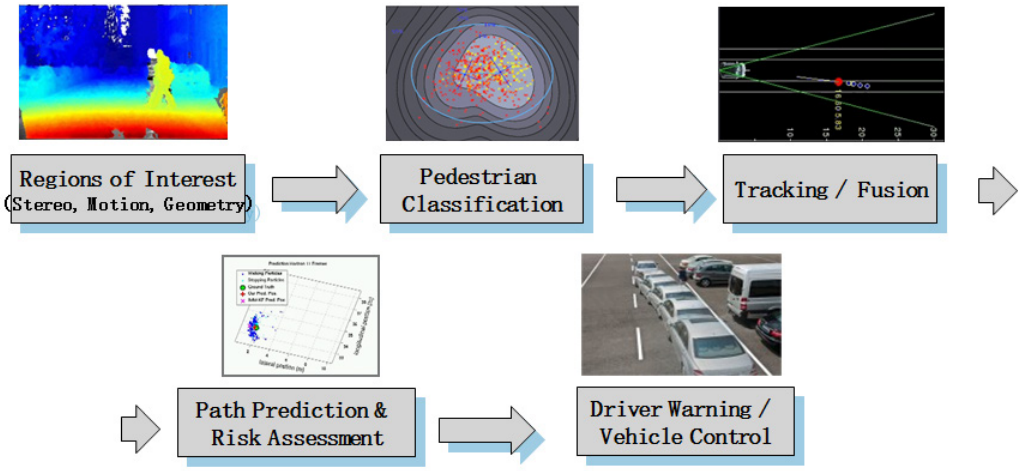


Figure 1-3: Pedestrian detection system architecture [11]

1.4 Tasks of the Thesis

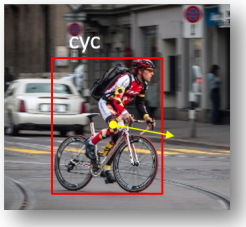
Our goal is to detect pedestrians/cyclists and estimate their pose. Furthermore we apply a part-based approach to detect body parts ((e.g., head, torso, bike) and estimate pose of body parts. The combination of pose information would benefits the analyses of road user behaviors and intentions, speed up the initialization of object trackers, improve detection precision, etc. The long term goal is to provide reliable detection and estimation results for intelligent vehicles system to make more accuracy risk assessment, hence avoid accidents and protect life.

In our approaches, we utilize the convolutional neural networks, which recently has shown very promising results in computer vision in recent years[12, 13, 3, 14]. We evaluate our methods on the public available Tsinghua-Daimler Cyclist Benchmark[4]. Though all the methods are only applied for cyclists, they are implemented to support multi-class tasks. In other words, they can be applied for other road user types such as pedestrians, cyclists and cars[15]. The tasks of this work can be summarized in the following points:

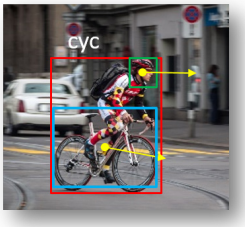
1. Utilize convolutional neural networks for cyclist detection.

2. Add ability to jointly detect and estimate pose information of a cyclist. (see Figure 1-4a).

3. Add ability to detect cyclists' body and head, and estimate their orientations respectively (see Figure 1-4b).



(a) Detection and orientation estimation for cyclist.



(b) Bike and head detection and orientation estimation for cyclist.

Figure 1-4: Tasks of this thesis.

The structure of thesis is as follows: Chapter 2 will give the introduction of fundamental knowledges required for this work. Chapter 3 and 4 will give an more close insight of the state-of-art technologies that are directly related to our approaches, including convolution neural networks and orientation estimation. Chapter 5 will detailedly describe the approaches that are proposed and deployed in our work. The experiments settings and results will be shown in Chapter 6. We will summarize our work and give the possible directions of further research in the last chapter.

Chapter 2

Fundamentals

2.1 Object Detection

Object detection is always a challenging task in computer vision for decades. Not like humans who can easily recognize a variety of objects with different scales, illumination conditions, angles of views, occlusions, computers only see matrices of digits. In this section we are going to give the basic knowledge of in-trend object detection approaches.

In the field of computer vision there are three related concepts: classification, detection and recognition. Briefly speaking, for classification you are given an unknown object input and your task is to classify what object category it belongs to (eg. book, computer, cup, etc). The input of classification task is different objects and the output is the category of each input. For object detection, it usually consists two tasks: finding the position of an potential object and classifying this object. The common approach for object detection is first searching for locations to look for object candidates, and then using classifiers to identify whether it's object/non-object and its category as well. Recognition is similar to classification, whoever it usually refers to finer 'classification', like identifying the identity of an person, the brand/model of a car.

One of the task in our work is detecting cyclists/pedestrians, so we go more in detail about concepts of object detection. A typical detection pipelines generally starts from searching for possible regions (called RoI, region of interest), extracts features region by region and then determines the object categories of each region with classifiers. The classifiers are usually trained off-line with positive and negative samples. Figure 2-1 shows a basic scheme for head detection.

In test case the first import part is searching for RoIs at different resolutions and locations. A

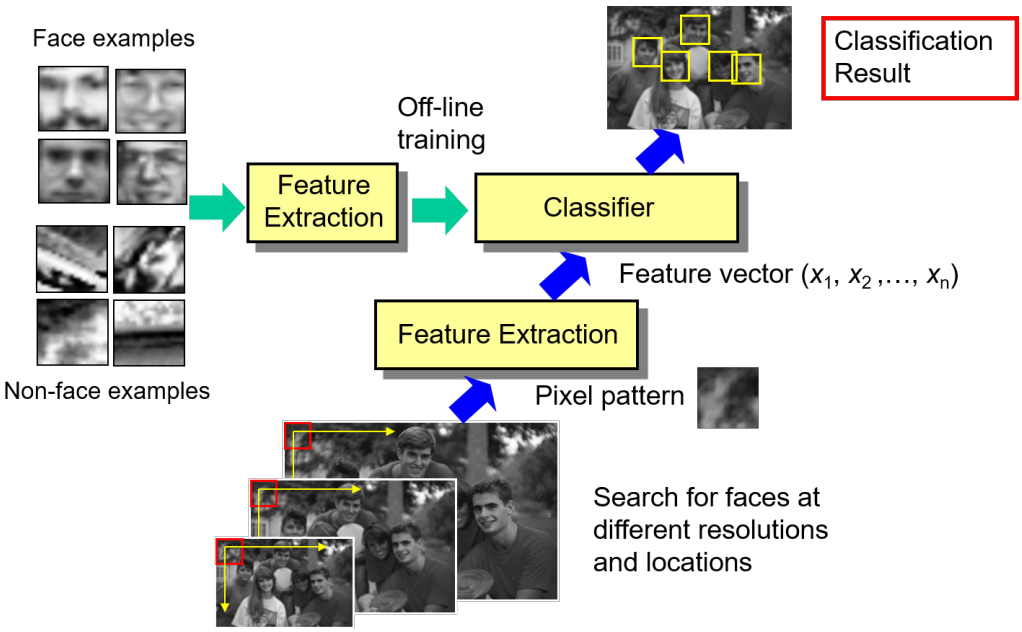


Figure 2-1: A basic scheme for head detection. Adapted from [16]

simple and robust approach is called Sliding Window (see Figure 2-2), where an exhaustive list of boxes with different positions, scales and aspect ratios are evaluated. Usually the detector of sliding window will heavily depend on the image size and scales of searching, which is very time consuming. To address this problem, detection proposal methods such as Selective Search [1], Edge Boxes [17] and Regional Proposal Networks [18] are proposed.

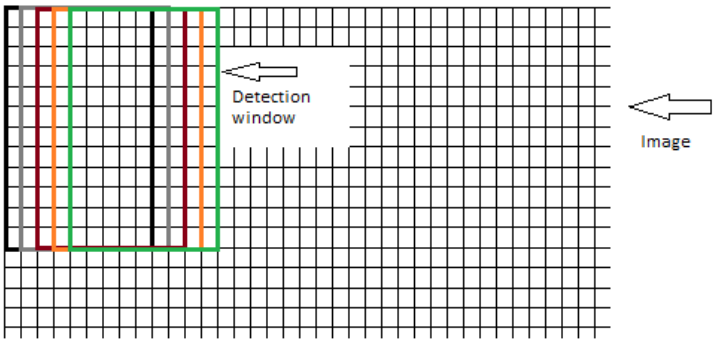


Figure 2-2: Sliding window method.

Selective Search [1] adapt a hierarchical bottom-up grouping method [19, 20] for generating object locations. It combines the best of both the intuitions of segmentation and exhaustive search. They use the method of Felzenszwalb and Huttenlocher[20] to create initial regions. Then a bottom-up hierarchical grouping algorithm is used to iteratively group regions. The similarities between all neighboring regions are calculated. The pair with highest similarity are grouped together and new similarities involving this new region are updated. For similarity calculation, they use different color spaces and similarity measures to make the algorithm robust

against various image conditions and object classes.

Edge Boxes [17] is based on sliding window approach. Instead of directly calculating fine features of each window, they demonstrate a effective method to evaluate the proposed boxes: scoring a box based on the number of contours it wholly encloses. The contours are obtained by first calculating edge responses of pixels [21, 21] and then grouping the edges according to their affinity. The top-ranked object proposals are further refined using a simple coarse-to-fine search.

Region Proposal Network (RPN) [18] is part of Faster R-CNN [18] which is proposed to make R-CNN [1] rid of relying on external proposals and to have a end-to-end network. To generate proposals, a small network is slided over the convolutional feature map output and trained for evaluating the 'objectness' of the current location and regressing the possible object bounding boxes in different scales and ratio aspects.

The above methods are image-based proposal methods which only use monocular information. And in other scenarios, (e.g., intelligence vehicle, sensors like binocular camera and LIDAR are usually used to provide stereo information. Works like [22, 23, 24, 25] employ stereo-based methods for generating proposals efficiently. [23, 24, 25] use stereo information to filter boxes generated from sliding window, while [22, 26] adopts the Stixel World [27] for directly proposing boxes.

The second import part of the detection pipeline is feature extraction and classifier training. A proper selection of features and classifier players an important role for object detection and classification tasks.

Histogram of Oriented Gradients (HOG) [28] is one of the common used feature descriptor. The thought behind this approach is that local object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions. It takes gray-scale regions as input and the regions are then divided as grid cells ((e.g., 16×16). For the pixels within each cell, a histogram of gradient directions is compiled. The HOG descriptor (feature) vector is formed by concatenating and normalizing all the histograms inside this region. The feature extraction pipeline is shown in Figure 2-3. The obtained HOG descriptors are fed into supervised recognitions systems like Support Vector Machine (SVM) or neural network classifiers to make decisions.

Another well know approach is using Haar-like features with cascade classifiers for face detection [30, 31], where they used Sliding Window to search the locations. For each detection window, a set of filters are applied to the tiled locations with different scales in the region and there will be in total up to 160,000 response values to form the feature vector. With such high

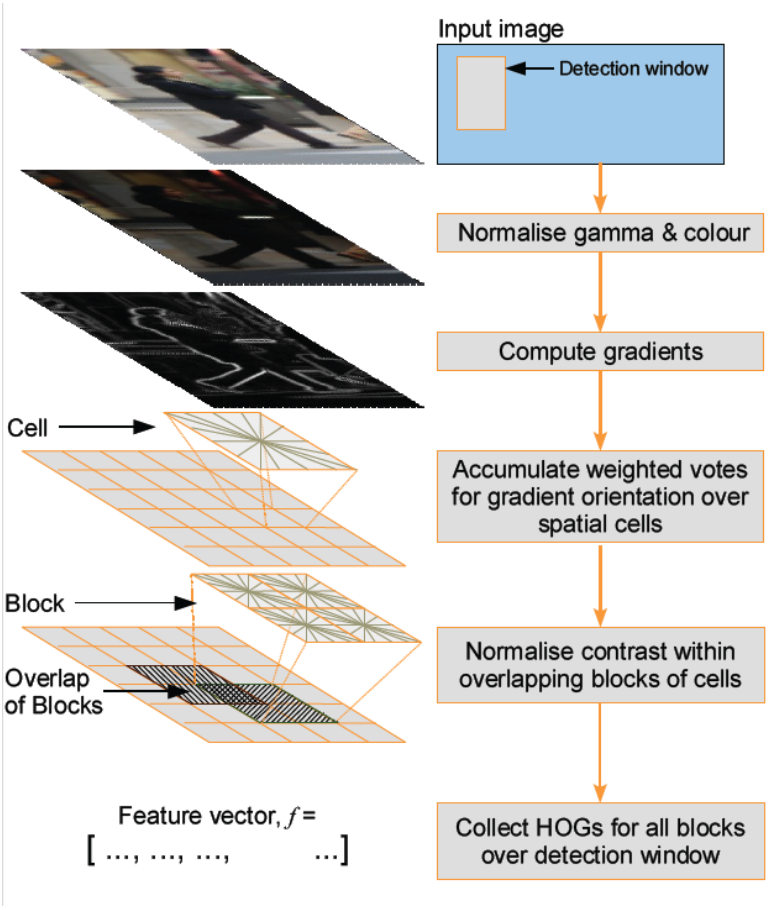


Figure 2-3: HOG descriptor.[29]

dimensional features, Viola and Jones used AdaBoost to train cascade weak classifiers to get a strong classifier. Figure 2-4 shows the Haar-like features and their overlay with face regions.

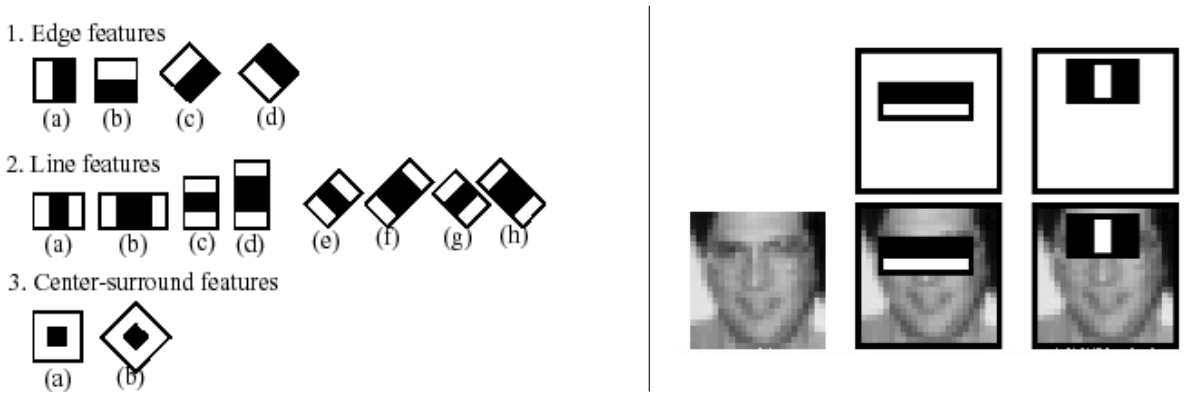


Figure 2-4: Left: Haar-like features (filters)[31]. Right: Two features are shown in the top row and then overlayed on a typical training face in the bottom row[30].

In the above approaches, each object is represented by one model which works well when the objects are nearly rigid and the viewing angle does not vary much. However in most real cases, the previous preconditions are not satisfied. Felzenszwalb *et al.* proposed deformable part model [32] to address these issues, which extends the HOG-SVM framework by defining not only detectors for whole objects, but also for parts under higher image resolutions. The

whole-object detection and parts detections are scored together to get finer object detection results.

In most detection tasks, the systems are required to handle object in different scales. The standard way is to re-sample the images to create scale pyramid and calculate feature channels for each scale. Noticing that the feature channel calculation is time-consuming, Dollár *et al.* proposed Aggregated Channel Features (ACF) [33] to speed up the feature calculation under different scales. In ACF, the images are only re-sampled to a sparse set of scales and the feature channels are to be calculated for thesis scales. Then the feature channels for the intermediate scales are directly approximated from the pre-calculated scales.

The Conlutional Neural Network (CNN) based feature extraction methods share the similar idea with Haar-like feature extraction. Both methods relies on calculating the image/region response of different filters/templates and use the response values to form feature vectors for classification. However unlike HOG descriptors, Haar-like features, etc. CNNs do not rely on manual constructed feature types but learn the features (filters) itself. More details about CNNs will be described in Section 2.3.

2.2 Pedestrian and Cyclist Detection

Among all the detection tasks, pedestrian and cyclist detection is a key problem with several applications that have potential to directly have positive impact of our life. It has seen widely use in video surveillance systems, intelligent vehicle systems, etc. as it provides fundamental information for semantic understanding of the scenes.

Different from challenges like Large Scale Visual Recognition Challenge (ILSVRC) which has 1000 object categories, the detection systems of cars or surveillance usually focus on a limit number of object categories including pedestrians, cyclists, riders, cars and so on, however face several challenges which are application specific:

- Low Resolution. The target object can be very far from the viewing point resulting small scale of the presentation.

- Various of appearance. For pedestrians and riders, the clothing styles, colors, textures various from person to person. Besides, the persons also shows different pose and gestures. For vehicles, they have different orientations and the models also shows a large variety.

- Occlusion. In real road conditions, occlusion always happens not only among objects

but also between target objects and static background objects. The occlusion levels also various a lot.

- Object crowding. Pedestrians can appear in crowds frequently. It is challenging to identify single person from a crowd of people.
- Motion blur. Both the movement of cameras and target objects can cause blurring of acquired images and videos. And the level of blurring also various depending on the speed.
- Motion of background. For The movement of cars also cause the motion of background, which will affect the separation of background and moving objects.
- Various illumination conditions. The illumination conditions largely depend on weather, light source position, shadows resulting different appearance of same objects.
- Efficiency. The detection systems on cars requires real-time performance with low delay.

(a) Backlighting (b) Motion blur

(c) Crowd (d) Occlusion

Figure 2-5: Pedestrian examples with different viewing conditions.

Despite the challenges, numerous approaches have been proposed. For holistic detection, methods for generic objection like HOG-SVM[28] still applies to pedestrian detection. For parts-based detectors[34, 35, 36], pedestrians are modeled as collections of parts. Hypotheses for parts (and full pedestrian) are firstly generated and evaluated jointly to create the estimation for pedestrian detection. Another category of methods relies on motion-based information. Methods like [37, 38] used fixed camera and stationary lighting conditions so that different of

frames in a image sequence could be used to separate the static background and moving objects. Optical flow[39] provides good representation of moving scenes and objects and approaches like [40, 41, 42] used optical flow for object detection and tracking.

2.3 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are variants of Multilayer Perceptrons (MLPs, or Artificial Neural Network - ANN). They were originally biologically inspired by Hubel and Wiesel's early work [43], which are designed to emulate the behavior of a visual cortex. Unlike Regular ANNs which receive single vector as input and transform it through a serious of hidden layers, CNNs take full images as input and employ 3D volumes of neurons. The two main types of layers, Convolutional Layer and Pooling Layer, keep the 2D structure of input images inside CNNs. Thus, the neurons in a layer will only be connected to a small region of the layer before it, which is the similar with how visual cortex works. CNNs saw heavy use in 1990s and it worked well on simple tasks like digits and characters recognitions in documents [44]. But then they fell out of fashion because of the increasing problem complexity and the limit of computing resources. In 2012, [12] brought CNNs back to life by showing significant improvement of the best image classification accuracy on ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [45] and high efficiency on GPU implementation.

2.3.1 Multi-Layer Perceptron

To describe any neural networks, we start from the describing a neuron which is the most basic computational unit in an network. Similar to biological neuron model, the inputs carry the signals to the artificial neuron where they get summed with weights. If the final sum is above a certain threshold, the neuron will fire and send signal to its output channel.

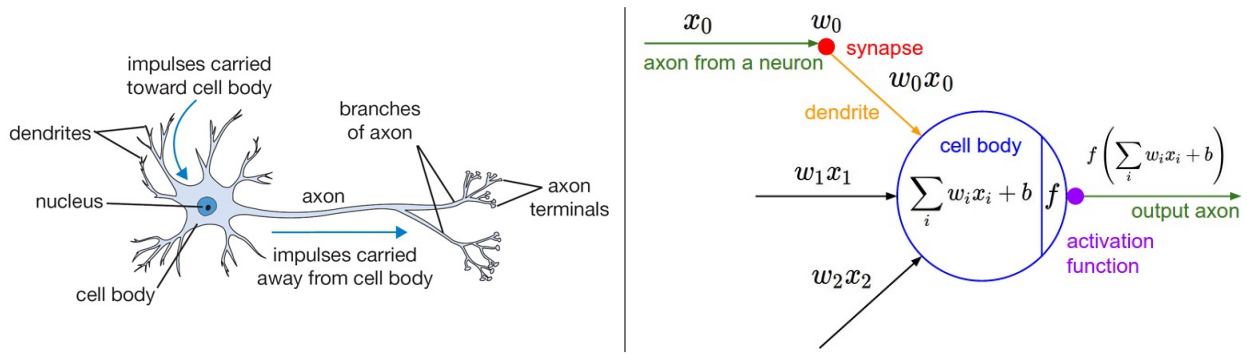


Figure 2-6: A cartoon drawing of a biological neuron (left) and its mathematical model (right).[46]

Figure 2-6 shows the biological structure of a neuron and the mathematical model of the inspired artificial neuron. The neuron takes $\mathbf{x} = (x_0, x_1, \cdots, x_n)^T$ as input and outputs $o(\mathbf{x}) = f(\mathbf{w}^T\mathbf{x} + b) = f(\sum_i w_i x_i + b)$, where $\mathbf{w}$ and b are weights and bias parameters of this neuron which will be learned when training. $f : \mathbb{R} \mapsto \mathbb{R}$ is a non-linear activation function with the following common choices:

Name	Equation
Sigmoid	$f(z) = \frac{1}{1+e^z}$
TanH	$f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
SoftPlus	$f(z) = \ln(1 + e^z)$
ReLU[47]	$f(z) = \max(0, z)$

Table 2.1: Common activation functions.

To summarize, each neuron performs a weighted sum of the input with bias and applies the non-linear activation function. In the case of Sigmoid function, the single neuron structure also corresponds to the input-output mapping defined in logistic regression.

An Artificial Neural Network (ANN) is obtained by putting together multiple neurons and connect them end-to-end in an layered structure. Usually an ANN consists of one input layer, one output layer and several hidden layers. The layers are full-connected, which means that each neuron is symmetric to all the other neurons in the same layer.

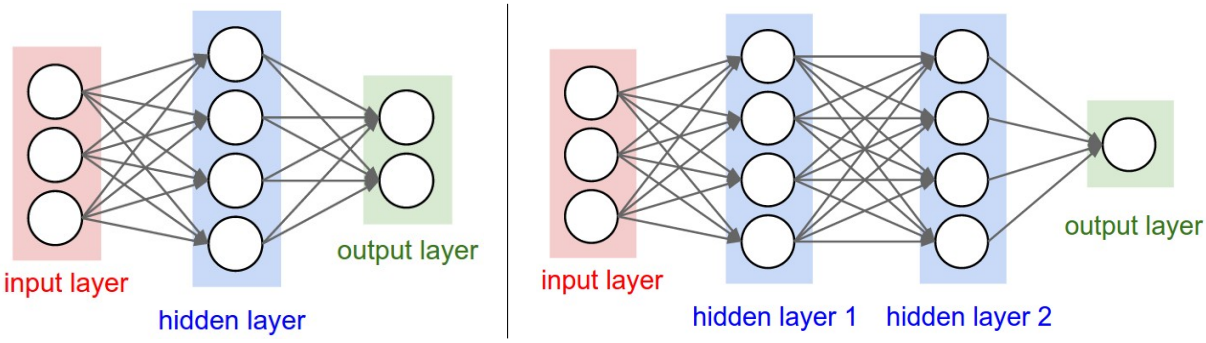


Figure 2-7: Left: A 2-layer Neural Network (one hidden layer of 4 neurons (or units) and one output layer with 2 neurons), and three inputs. Right: A 3-layer neural network with three inputs, two hidden layers of 4 neurons each and one output layer.[46]

The common method of training an network is using back-propagation which is based on gradient descent optimization. To apply gradient descent, the derivatives of the error function with respect to the weights must be evaluated and then the derivatives are used to compute the adjustments to be made to the weights.

Assuming a network that uses activation function h, the neuron unit j in layer l with its weighted sum a_j^l and output z_j^l as defined in (2.1) and (2.2). w_{ij}^l denotes the weight of the

connection between neuron i at layer $l-1$ and neuron j at layer l. E denotes the final error and $\mathcal{L}$ is the error function defined on the output neuron units $\mathbf{a}^L \equiv (a_0^L, a_1^L, \cdots)^T$.

$$a_j^l = \sum_i w_{ij}^l z_i^{l-1} \tag{2.1}$$

$$z_j^l = h(a_j^l) \tag{2.2}$$

$$E = \mathcal{L}(\mathbf{a}^L) \tag{2.3}$$

The derivatives of the error function with respect to the weights can be calculated as follows

$$\begin{aligned} \frac{\partial E}{\partial w_{ij}^l} &= \underbrace{\frac{\partial E}{\partial a_j^l}}_{\delta_j^l} \frac{\partial a_j^l}{\partial w_{ij}^l} \\ &= \delta_j^l z_i^{l-1} \end{aligned} \tag{2.4}$$

The second term in Equation (2.4) is derived from (2.1). The first term δ is often referred as the error that is back-propagated during training. By applying chain rule, δ yields a recursive definition in the back-propagation formula as shown in Equation (2.5).

$$\begin{aligned} \delta_j^l &\equiv \frac{\partial E}{\partial a_j^l} \\ &= \sum_k \underbrace{\frac{\partial E}{\partial a_k^{l+1}}}_{\delta_k^{l+1}} \frac{\partial a_k^{l+1}}{\partial a_j^l} \\ &= \sum_k (\delta_k^{l+1} \frac{\partial}{\partial a_j^l} \sum_i w_{ik}^{l+1} z_i^l) \\ &= \sum_k (\delta_k^{l+1} \frac{\partial}{\partial a_j^l} w_{jk}^{l+1} z_j^l) \\ &= \sum_k \delta_k^{l+1} w_{jk}^{l+1} h'(a_j^l) \end{aligned} \tag{2.5}$$

This shows that the value of δ for a particular neuron in a hidden layer can be obtained by propagating the δ's errors from the higher layer in the network. Because we already know the error E with the definition of function $\mathcal{L}$ of output neurons $\mathbf{a}_j^L$, we can easily derive the initial errors of the output layer

$$\delta_j^L = \frac{\partial E}{\partial a_j^L} = \frac{\partial \mathcal{L}}{\partial a_j^L}. \tag{2.6}$$

The back-propagation procedure can evaluate the error δ for all neuron units by recursively applying Equation (2.5) in the network regardless of its topology.

2.3.2 Convolutional Neural Networks

In image based pattern recognition, an image can also be represented as a vector with the size of $width \times height \times depth$. If we feed this vector into an regular full-connected ANN whose hidden layer has the same magnitude of unites as the input, it will lead to a huge amount of trainable parameters. Furthermore, the full-connected layers discard the structural information inside the image.

CNNs are proposed to address the above issues by introducing a new connectivity pattern between its' neurons, which is inspired by the organization of animal visual cortex. The individual cells in a visual cortex are arranged in such a way that they only sensitive to small sub-regions of the visual field, called receptive field. The sub-regions are tiled to cover the entire visual field and the cells act as local filters over the input space. The neurons' connectivity of CNN is designed like visual cortex, which basically has two main properties, local connectivity and weights sharing.

Local Connectivity

Instead of expanding the input image as a one dimensional vector, CNNs keep the original image structure as input. The hidden layers in the network are also arranged in 3 dimensions: width, height and depth. To avoid dense connectivity in full-connect layers, each neuron inside a layer only connects to a local region of the previous layer referred as receptive field as in visual cortex. Therefore each single neuron is only responsible for a sub-region in previous layer instead of the whole volume. See Figure 2-8.

Weights Sharing

In addition, the connections between an neuron and its corresponding receptive field refers to one filter. Each filter is replicated across the entire image field, which means that the neurons in the same depth slice shares the same parameters/filter. In other words, each depth slice reveals the responses of one filter applied on the previous layer. Usually a depth slice is referred as a feature map, and thus a layer consists of several feature maps.

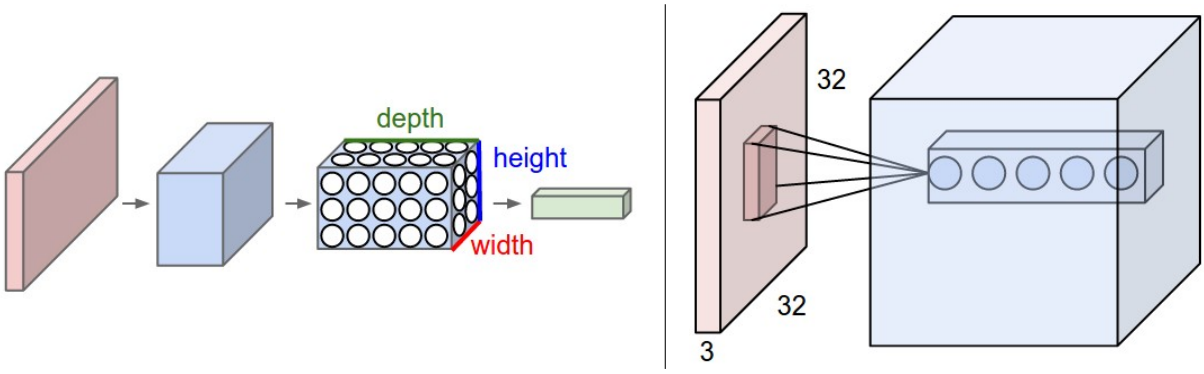


Figure 2-8: Left: A typical CNN arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. The intermediate layers of a CNN transform one 3D input volume to another 3D output volume of neuron activations. The output layer is Right: An example input volume in red ((e.g., a 32x32x3 CIFAR-10 image), and an example volume of neurons in the first convolutional layer. Each neuron in the convolutional layer is connected only to a local region in the input volume spatially, but to the full depth (i.e. all color channels).[48]

Noticing that this scheme suits the convolution well, convolution operation is adopted as the key part of this kind of networks. Thus CNNs are basically several layers of convolutions with nonlinear activation functions applied to the results.

Beside the above two key properties, a CNN also use other techniques which will be explained in detail in the next chapter.

2.3.3 Caffe

Currently there has been quite some ready-made CNN frameworks or tools with CNN support ((e.g., Caffe [49], Torch [50], Theano [51], TensorFlow [52], Darknet [53]) which could be easily adapted for different tasks image and video recognition, recommender systems and natural language processing.

Caffe is a framework specialized for CNN using C++ implementation and Google protocol buffers for defining network architectures and controlling training/testing. It provides APIs for Python and Matlab. Caffe provides clean architecture, great performance, detailed documentations, modularity and flexibility which encourage developers to create implementations for state-of-art methods. Furthermore, Caffe introduced 'Caffe Model Zoo' where you could get and share models with their definitions and trained weights, thus allowing researchers to build on top of each other's work. Besides the above advantages, the system on our testing vehicle has already integrated Caffe. Thus we choose Caffe for the implementation of our work.

Chapter 3

Convolutional Neural Networks for Object Detection

3.1 Overview of Related Work

CNNs have been widely used for full-image based classification applications. In recent years, a series of Region-based Convolutional Neural Netwroks (R-CNN) [54, 1, 18] methods are proposed to apply CNNs on object detection tasks. The original version of R-CNN [54] takes full image and object proposals as input. The regional object proposals could come from a variety of methods and in their work they use Selective Search. Each proposed region is then cropped from the original image and wrapped to a unified 227×227 pixel size. A 4096-dimensional feature vector is extracted by forward propagating the subtracted region through fine-tuned CNN with five convolutional layers and two fully connected layers. With the feature vectors, a set of class-specific linear SVMs are trained for classifications.

R-CNN achieves excellent object detection accuracy, however, it has notable drawbacks. First, training and testing has multiple stages including fine-tuning CNN with Softmax loss, training SVMs and learning bounding-box regressors. Secondly, the CNN part is slow because it performs forward pass for each object proposal without sharing computation. To address the speed problem, Spatial Pyramid Pooling network (SPPnet) [55] and Fast R-CNN [1] are proposed. Both methods compute one single convolutional feature map for the entire input image and do the cropping on the feature map instead of on the original image and then extract feature vectors for each region. For feature extraction, SPPnet pools the feature maps into multiple sizes and concatenate them as a spatial pyramid [56], while Fast R-CNN only use single scale of the feature maps. The feature sharing of SPPnet accelerates R-CNN by 10 to

100× in testing and 3× in training. However it still has the same multiple-stage pipeline as R-CNN. In Fast R-CNN Girshick propose a new type of layer, region of interest (RoI) pooling layer, to connect the gap between feature maps and classifiers. With this layer, they build an 'semi' end-to-end training framework which only rely on full image input and object proposals.

The above methods all rely on external object proposal input. Ren *et al.* proposed proposal-free framework called Faster R-CNN [18]. In Faster R-CNN, they use a Region Proposal Network (RPN), which slides over the last convolutional feature maps to generate bounding-box proposals in different scales and ratio aspects. These proposals are then fed back to Fast R-CNN as input. Another proposal-free work You Only Look Once [57] is proposed by Redmon *et al.* Their network uses features from the entire image to predict object bounding box. Instead of sliding windows on the last convolutional feature maps, their network connects the feature map output to an 4096-dimensional followed by another full-connected $7 \times 7 \times 24$ tensor. The tensor is a 7×7 mapping of the input image. Each grid of the tensor is a 24-dimensional vector which encodes bounding boxes and class probabilities of the object whose center falls into this grid on the origin image. The YOLO network is 100 to 500× faster than Fast R-CNN based methods, though with less than 8% mean average precision drop on VOC 2012 test set [58].

Some other specific R-CNN variants are also proposed to solve different problems. Gkioxari *et al.* present a R-CNN based networks with triple loss functions combined for the task of keypoints (as representation pose) prediction and action classification of people [59]. They also adapt R-CNN to use more than one region, but also contextual subregions for human detection and action classification called R*CNN[60]. Ouyang *et al.* proposed DeepID-Net with deformation constrained pooling layer[61], which models the deformation of object parts with geometric constraint and penalty.

3.2 Important Concepts

In this section, we are going to start with important concepts related general CNNs including the structure of networks, essential layers and loss calculation.

3.2.1 Network Structures

A typical convolutional neural network structure is shown in Figure 3-1. The input image is processed by several convolution layers and subsampling layers to get feature maps of the current input. The feature maps are usually then fully connected (by fully connected layers)

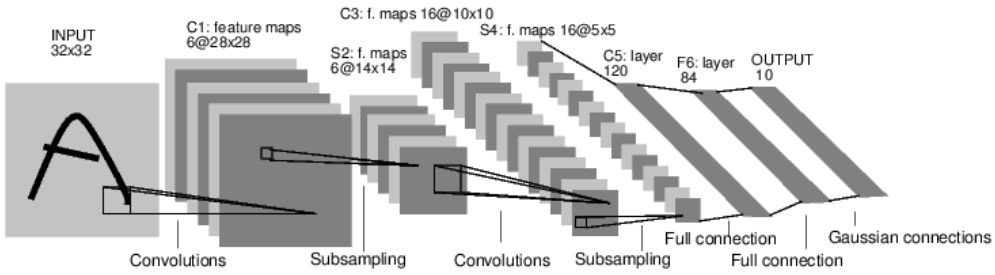


Figure 3-1: Architecture of LeNet-5[44].

to several vector-shape layers to get the feature vector of the input image. These feature vectors can be used for any tasks that relies on feature extraction of images, like training SVM classifiers. Another option is direct connect these feature outputs to regression or classification output layers, which will be explained in later sections.

3.2.2 Layers in CNNs

As shown above, convolutional neural networks are commonly made up of mainly three layer types: convolutional layer, pooling layer (usually subsampling) and fully connected layer. We are going to give explanations of these layers and additionally we are also going to give introduction of other auxiliary layers that are not shown in Figure 3-1.

Convolution Layer

As mentioned in Section2.3.2, in very high level, the convolution operation replicates a filter across the entire image field to get the response of each location and form a response feature map. Given multiple filters, the network will get a stack of features maps to form a new 3D volume.

Formally a convolution layer accepts a volume of size $W_1 \times H_1 \times D_1$ from previous layer as input data. The layer defines K filters with the shape $F \times F \times D_1$ each. The convolution of input volume and filters produces the output volume of size $W_2 \times H_2 \times K$, where the new volume's W_2 and H_2 are dependent on the filter size, stride and pad settings of the convolution operation. Figure 3-2 illustrate a 2D version convolution where the $5 \times 5 \times 1$ input volume is convolved with one 3×3 filter. With 0 pad and 1 stride settings, it produce a $3 \times 3 \times 1$ output volume.

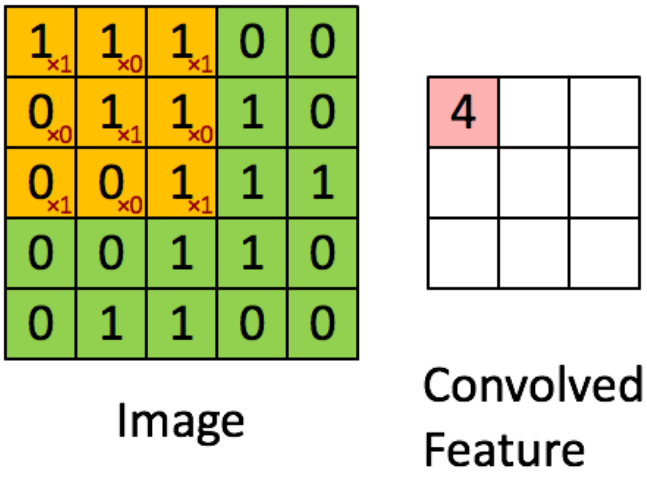


Figure 3-2: Convolution in 2D.

Pooling Layer

Another important operation is pooling (also referred as downsampling). Its function is to reduce the spatial size of representation and hence reduce the amount of parameters and computations in the network and also control overfitting. Common pooling methods includes max-pooling, average-pooling and stochastic-pooling. It is an depth-slice-wise operation, thus the pooling of each depth-slice is independent. Pooling is an translation-invariance operation. The pooled image keeps the structural layout of the input image.

Formally a pooling layer accepts a volume of size $W_1 \times H_1 \times D_1$ as input and output a volume of size $W_2 \times H_2 \times D_1$. The output width W_2 and height H_2 are dependent on the kernel size, stride and pad settings.

Fully Connected Layer

Neurons in a fully connected layer have full connections to all neurons in the previous layer. It provides a form of dense connectivity and loses the structural layout of the input image. Fully connected layers are usually inserted after the last convolution layer to reduce the amount of features and creating vector-like representation.

Activation Layer

In activation layers, the activation functions (see Table 2.1) are element-wise applied to the neurons. The input and output volumes shapes are identical. For CNNs, ReLU is in more common use than other activation functions because it is much more efficient[12] and largely

avoids vanishing gradient problem[62].

Local Response Normalization Layer

Basically the local response normalization (LRN) layer performs a kind of 'lateral inhibition', which is a neurobiology concept. It refers to the capacity of an excited neuron to subdue its neighbors. This tends to create a contrast in the excited area, hence increasing the sensory perception. Mathematically, it performs normalization by dividing each input value x_i with normalizer $(1 + \frac{\alpha}{dn} \sum_i x_i^2)^\beta$, where n is the size of each local region, and the sum is taken over the region centered at that value. α and β are layer hyper parameters.

3.2.3 Loss Calculation

Loss calculation drives the learning process by comparing the network output with the target and minimizing the cost. The loss itself is calculated by forward pass and the gradient of network parameters with respect to loss is calculated by the backpropagation.

For multi-class classification tasks, Softmax classifier with loss is commonly used. It first takes multi-class scores as input, uses Softmax function $f_j(z) = \frac{e^{z_j}}{\sum_k e_{z_k}}$ to normalize the input and get a distribution-like output. Then the loss is computed by calculating the cross-entropy of the target class probability distribution and the estimated distribution. The cross-entropy between the target distribution p and the estimation distribution q is given by $H(p, q) = -\sum_j p_j \log q_j$.

In Caffe framework, multiple tasks with loss layers can be defined in the same network. Each loss layer is assigned with a weight. The final loss is the weighted sum of all the loss layers. This scheme is called 'Multi-task Loss' in Caffe.

3.2.4 Inception Module

Additionally we introduce the inception module[3] developed by Szegedy *et al.* which is also used in our work. An inception module is composed by several fundamental layers (see Figure 3-3). The idea behind is that normally we only take one scale of convolution, however the patterns on a feature map may appear at different scales. Thus multiple scales of convolutions (1×1, 3×3 and 5×5) are deployed from the same layer and the output feature maps are the concatenated together. This small architecture is proven to be beneficial for training CNNs in their work.

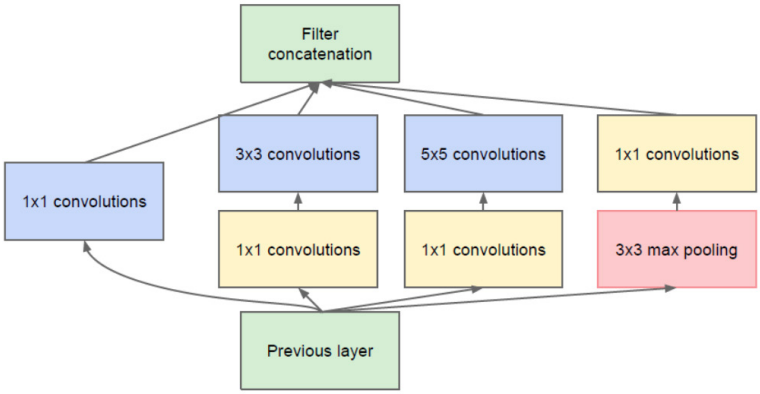
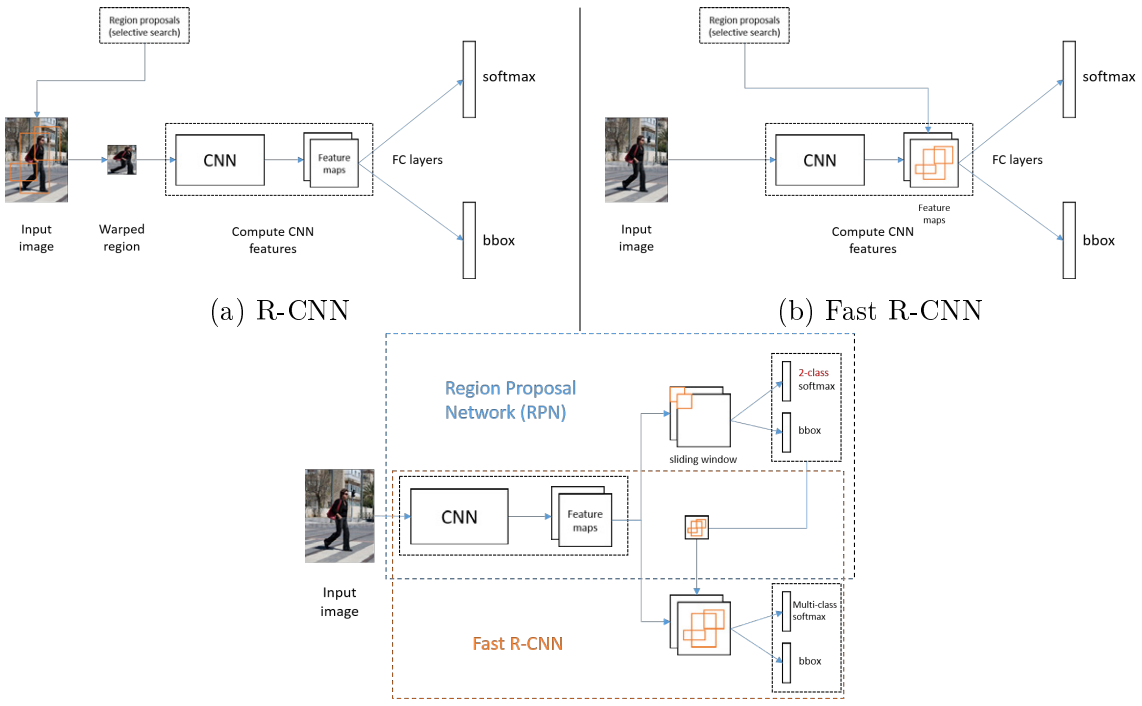


Figure 3-3: Inception module[3].

3.3 R-CNN Variants

3.3.1 The R-CNN Family



(a) R-CNN (b) Fast R-CNN

(c) Faster R-CNN

Figure 3-4: The R-CNN family

The original R-CNN Family contains three works: R-CNN[54], Fast R-CNN[1] and Faster R-CNN[18]. The relationship of the three architectures are illustrated in Figure 3-4. Briefly speaking, R-CNN and Fast R-CNN rely on external region proposals (generated by Selective Search, etc). R-CNN directly crop regions from the image and feeds them into the network for training and classification, while in Fast R-CNN the cropping takes place on the last convolutional feature maps. The advantage of using Fast R-CNN is that it does not require external

disk space for saving cropped regions, which achieves end-to-end training. And since this is only one forward pass calculation for each image, the running time reduces a lot compared with original R-CNN. Faster R-CNN introduced Region Proposal Network (RPN), which slides over the last convolutional feature maps to generate bounding-box proposals in different scales and ratio aspects. The RPN is a two-class classification which evaluate the 'objectness' of the candidate boxes. The boxes with high 'objectness' are selected as region proposals and then fed into the Fast R-CNN for finer object classification. Faster R-CNN achieves fully end-to-end training without relying on external proposals.

3.3.2 R-CNN Specific Concepts

In R-CNN based networks, there are some specially developed structures. This section will give the explanation of these building blocks.

RoI Pooling Layer

RoI pooling layer is a variant of max pooling layer. It takes previous layer output and a list of region proposal boxes (RoIs) as input and pool each box into a small feature map with a fixed spatial extent of $W_2 \times H_2$ ((e.g., 6×6), where W_2 and H_2 are layer hyper-parameters that are independent of any particular RoI. Each box defines the coordinates of a rectangle from the original image.

Assume that the input layer has size $W_1 \times H_1 \times D_1$, the box list contains K RoIs and the original image size is $W_0 \times H_0 \times D_0$. The RoI pooling works by first scaling the box list with ratio $\frac{W_1}{W_0}$ and $\frac{H_1}{H_0}$, then locating the scaled boxes on the input layer and pooling them into $W_2 \times H_2 \times D_1$ feature maps. K feature maps are stacked together to create a batch with size $K \times W_2 \times H_2 \times D_1$ for training.

Bounding-box Regression

In high level, bounding-box regression solves the problem which is finding the bounding-box offset which better surround the target object given the proposed RoI. (x, y, w, h), (x_p, y_p, w_p, h_p) and (x^*, y^*, w^*, h^*) denote the predicted box, proposal box and ground-truth box respectively.

The target offsets are encoded as

$$\begin{cases} t_x^* &= (x^* - x_p)/w_p \\ t_y^* &= (y^* - y_p)/h_p \\ t_w^* &= \log(w^*/w_p) \\ t_h^* &= \log(h^*/h_p) \end{cases} \tag{3.1}$$

and the predicted offsets are encoded as

$$\begin{cases} t_x &= (x - x_p)/w_p \\ t_y &= (y - y_p)/h_p \\ t_w &= \log(w/w_p) \\ t_h &= \log(h/h_p) \end{cases} . \tag{3.2}$$

For training, the loss function is defined as

$$L_{loc}(t, t^*) = \sum_{i \in \{x,y,w,h\}} smooth_{L_1}(t_i - t_i^*), \tag{3.3}$$

in which

$$smooth_{L_1}(x) = \begin{cases} 0.5x^2 & if|x| < 1 \\ |x| - 0.5 & otherwise \end{cases} \tag{3.4}$$

is a robust L_1 loss that is less sensitive to outliers than the L_2 loss used in R-CNN[1].

Chapter 4

Orientation Estimation

4.1 Overview of Related Work

For orientation estimation, there are a various of methods depending on applications. One early category is appearance template methods[63, 64, 65, 66] which compare new samples with a set of training examples (each labeled with a discrete pose) to find the most similar pose. The second category is detector-array-based [67, 68, 69, 70, 71, 72, 73, 74, 26], in which a series of head detectors are trained, each attuned to a specific pose. And a discrete pose is assigned to the detector with the greatest support. Based on classification result, probabilistic models could be introduced to continuous estimation result [71, 72, 73, 74, 26]. The third category is nonlinear regression, which estimates pose by learning a non-linear functional mapping from the image/feature space to orientations. This category includes earlier Support Vector Regressions (SVRs) [75, 76, 77], Neural Networks [78, 79, 80, 81] and latest Convolutional Neural Networks [82, 2]. Since appearance template methods are quite early approaches which is not showing state-of-art performances, we are not to explain them in detail. The following sections will mainly focus on classification based and non-linear regression methods.

4.2 Classification-based Orientation Estimation

Given the success of various object detection approaches, it is a natural extension to estimate orientation by training multiple detectors, each can be a binary classifier specifying a specific discrete orientation. The orientation can be estimated by the detector with greatest support. Some early approaches of a detector array directly use images as input (without feature extraction)[67, 68]. [67] trained three SVMs for discrete head orientations [67] and [68] proposed a

neural network based approach to output 36 discrete orientations. Due to limited computing power, the input images size were restricted to small scales ((e.g., 20×20). Later it is common to extract features of images/regions for orientation classification. There are various of feature-classifier combinations proposed, like SIFT [83, 84], HOG [71, 72, 84, 73, 74], Haar-like features [69, 70, 85, 72, 86] in combination with decision trees / random forests [69, 84, 74], AdaBoost [69, 85], SVM [70, 71, 73, 86], neural networks [72, 26]. Based on orientation classification results, Bayesian frameworks of Hidden Markov Models (HMMs) can be integrated to stabilize the estimations over time [71, 73, 26]. And continuous orientation estimation could also be obtained by introducing probabilistic models [72, 26].

4.3 Non-linear Orientation Regression

Non-linear regression methods estimate orientations by learning a non-linear functional mapping from the image/feature space direct to orientation outputs. Earlier Support Vector Regression (SVR) which is a variant of SVM has been successfully used for orientation regression[75, 76, 77]. [75, 76] used principle component analysis (PCA) to reduce the dimensionality of the input image so that it could be fed into SVR, while [77] used localized gradient orientation histograms for feature extraction. In recent years, of the non-linear regression tools used for orientation estimation, neural networks have been more widely used[78, 79, 80, 81]. These methods used image regions containing head as inputs and train neural networks with normalized 0 to 1 orientation outputs. The use of neural networks takes the advantage of the non-linear activation functions inside the network. The approaches are straight forward, however ignoring that periodicity of the angels.

4.4 CNN-based Orientation Regression

CNN-based orientation regression is also an non-linear regression method which adopted the recently widely used CNN architecture. The idea behind CNN-based method is the same as regular neural network[82], however taking advantages of the CNN networks. To address the periodicity issue, [2] adopted von Mises loss function with proposed Biternion representation.

Von Mises Loss Function

Von Mises distribution[87], also known as circular normal distribution, is a continuous probability distribution on the circle which address the first problem above. The von Mises distribution

$M(\mu, \kappa)$ has probability density function

$$g(\theta; \mu, \kappa) = \frac{1}{2\pi I_0(\kappa)} e^{\kappa \cos(\theta - \mu)}, \tag{4.1}$$

where θ is an angle, μ is the mean angle of the distribution, κ is the concentration parameter which is inversely related to the variance of the approximated Gaussian and I_0 is the modified Bessel function of the first kind and order 0:

$$I_0(\kappa) = \frac{1}{2\pi} \int_0^{2\pi} e^{\kappa \cos\theta} d\theta, \tag{4.2}$$

which is an constant when hyper parameter κ is given. Since the cosine function is periodic on circle, it naturally avoids the problem of discontinuity and well suits gradient-based optimization. By inverting and scaling constants, the von Mises loss function is defined as

$$L_{VM}(\theta | t; \kappa) = 1 - e^{\kappa(\cos(\theta - t) - 1)}, \tag{4.3}$$

where θ is the predicted angle, t is the target and κ is a hyper parameter.

Biternion Representation

The biternion representation[2] is proposed to make full use of the linear representation of network outputs, which is inspired by quaternion representation often seen in computer graphics. It wraps the linear-production output onto a unit circle in an elegant way. For an angle θ, the biternion representation is a two-dimensional vector consisting of its cosine and sine

$$\mathbf{y} = (\cos\theta, \sin\theta). \tag{4.4}$$

With the trigonometric identities we could rewrite the term $\cos(\theta - t)$ in (4.3) as following

$$\begin{aligned} \cos(\theta - t) &= \cos\theta \cos t + \sin\theta \sin t \\ &= (\cos\theta, \sin\theta) \cdot (\cos t, \sin t) \\ &= \mathbf{y} \cdot \mathbf{t} \end{aligned} \tag{4.5}$$

where $\mathbf{t} = (\cos t, \sin t)$ is the biternion representation of the target angle. The Biternion representation of the von Mises loss function is derived by substituting (4.5) back into (4.3)

$$L_{VM}(\mathbf{y}|\mathbf{t};\kappa) = 1 - e^{\kappa(\mathbf{y}\cdot\mathbf{t}-1)}. \tag{4.6}$$

Chapter 5

Methodology

At the time of writing this thesis, there has been no publications on combined cyclist detection and orientation estimation using R-CNNs. In this work, we first deploy the R-CNN frameworks on our cyclist dataset and introduce continuous orientation regression to the networks. The new architecture is named Pose-RCNN. We adapted the GoogLeNet for R-CNN architecture for our work as well, which is named R-GoogLeNet. In the last stage the networks are extended with parts detection and orientation estimation.

In this chapter we will go through the methodological details regarding the methods and setups in this work. First the a fast version of Pose-RCNN network architecture will be described. Secondly, this chapter will explain how orientation regression is derived and fitted into our framework in detail. Then the object proposal method that is used in this work will be introduced. Moreover, in the next section we will illustrate R-GoogLeNet which is also originally proposed in our work. Finally, the network architectures for object parts detection and orientation regression will be described.

5.1 Network Architecture for Joint Cyclist Detection and Orientation Estimation

Our Pose-RCNN is based on Fast R-CNN framework (see Figure 3-4b) which provides state-of-art object detection performance. To have orientation estimation of detected objects, we extend the framework with orientation output and corresponding loss function introduced in Section 4.4. The overview of new network architecture is shown in Figure 5-1.

To lay the foundation of our joint detection and orientation regression work, we firstly choose a relative small and computationally efficient model, the 'fast' version of ZF-Net [13]

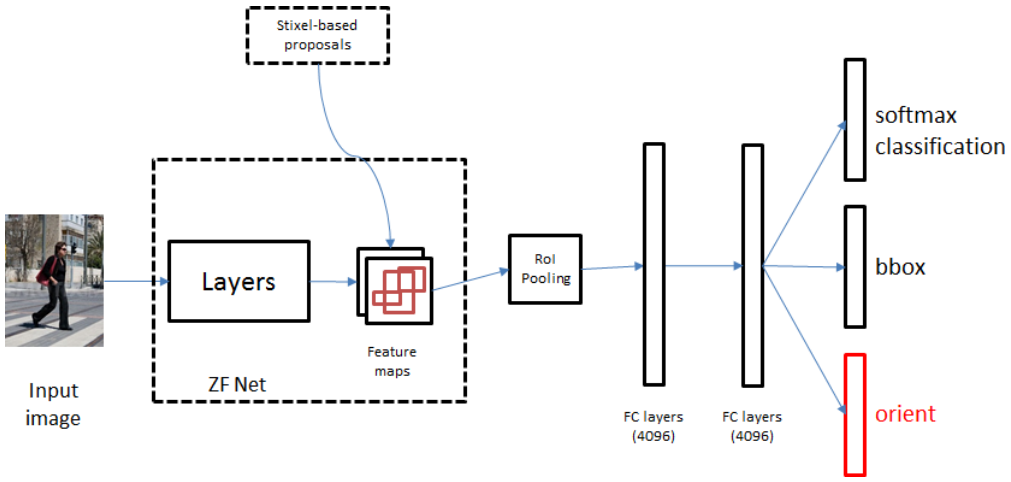


Figure 5-1: Network architecture of ZF-Net based Pose-RCNN.

which has five convolutional layers and three fully connected layers. Figure 5-2 illustrates the detailed layer structure of ZF-Net based Pose-RCNN.

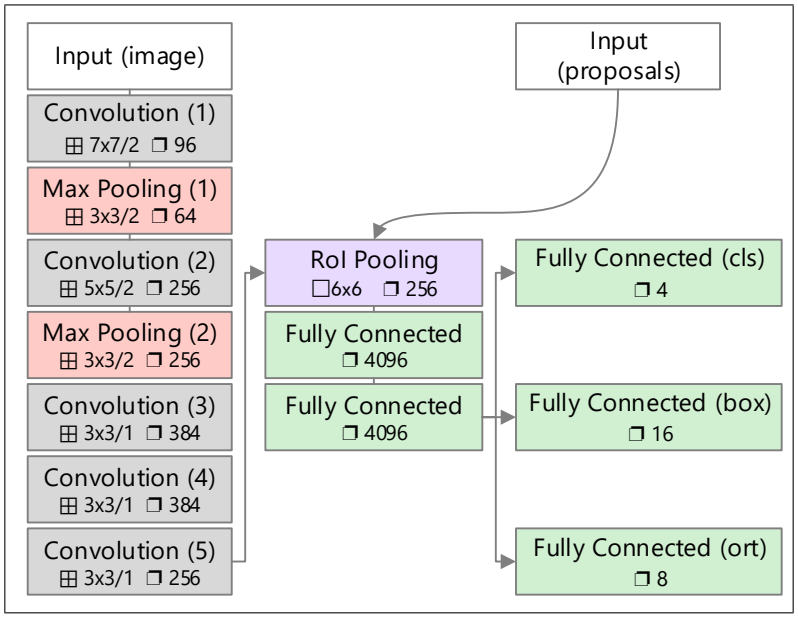


Figure 5-2: ZF-Net based Pose-RCNN architecture. An input image and multiple regions of interest (RoIs) proposals are input into the network. The RoI pooling takes place on the last convolutional feature map. The pooled feature map is then mapped by a sequence of fully connect (FC) layers to generate the three outputs of the network per RoI: softmax probabilities, per-class bounding-box regression offsets and per-class orientation regression (⊞ denotes the size of a layer's kernel and stride, ☐ is the RoI pooling layer width and height, and ❒ gives the depth of layers).

Similar to R-CNN, the network takes an entire image and a set of object proposals as input. The image is first processed by the network with several convolutional layers (see Section 3.2.2) and max pooling layers (see Section 3.2.2) to get a convolutional feature map of the whole image. For each object proposal, the RoI pooling layer (see Section 3.3.2) crops the corresponding region on the feature map and pools it into a fixed-size regional feature map followed by several

fully connected layers to generate prediction outputs. Different from R-CNN, the Pose-RCNN will have three outputs:

- **Softmax probabilities**, probability distribution over $K+1$ classes (K object classes plus one 'background' class).
- **Bounding-box regression**, four real-valued numbers for each of the $K+1$ classes. Each set of 4 values encodes bounding-box offset results for the corresponding class.
- **Orientation regression**, two real-valued numbers of for each of the $K+1$ classes. Each tuple of 2 values is the Biternion representation for corresponding class.

We use a multi-task loss L on each labeled RoI to jointly train for Softmax classification, bounding-box regression and orientation regression:

$$\begin{aligned} L(p, p^*, t, t^*, o, o^*) = & L_{cls}(p, p^*) \\ & + \lambda[p^* > 0] L_{box}(t, t^*) \\ & + \mu[p^* > 0] L_{ort}(o, o^*) \end{aligned} \tag{5.1}$$

Here, p, t and o are the predicted classification probabilities, bounding-box offsets and orientations of a proposed RoI respectively. The ground-truth label $p^* = 0$ if the proposal is background, and $p^* > 0$ for object classes. $[p^* > 0]$ is the indicator function which evaluates to 1 if the expression inside is true and 0 otherwise. t^* and o^* are the ground-truth bounding-box offsets and orientation. $L_{cls}(p, p^*)$ is the softmax loss, $L_{box}(t, t^*)$ is smooth L_1 loss defined in [1] and $L_{ort}(o, o^*)$ is the von Mises loss defined in Equation (4.6). λ and μ by default set to 1, thus the three tasks are equally weighted.

5.2 Orientation Regression for CNNs

As shown in the work of Beyer *et al.* [2], the von Mises loss with Biternion representation achieved quite promising performance. To adapt this approach into our work, there are several issues to be resolved. Firstly, the 2-dimensional output for orientation regression does not preserve the property of Biternion representation. It needs to be normalized before use. Secondly, both normalization and von Mises loss require Caffe layer implementation, thus the forward and backward propagation should be derived.

In general, the orientation works as follows: The two dimensional CNN output is treated as a vector starting from the origin of the coordinate plane. The vector is then normalized to an unit vector. And the loss is calculated given the unit-vector representations of predicted and target angles. This pipeline is shown in Figure 5-3.

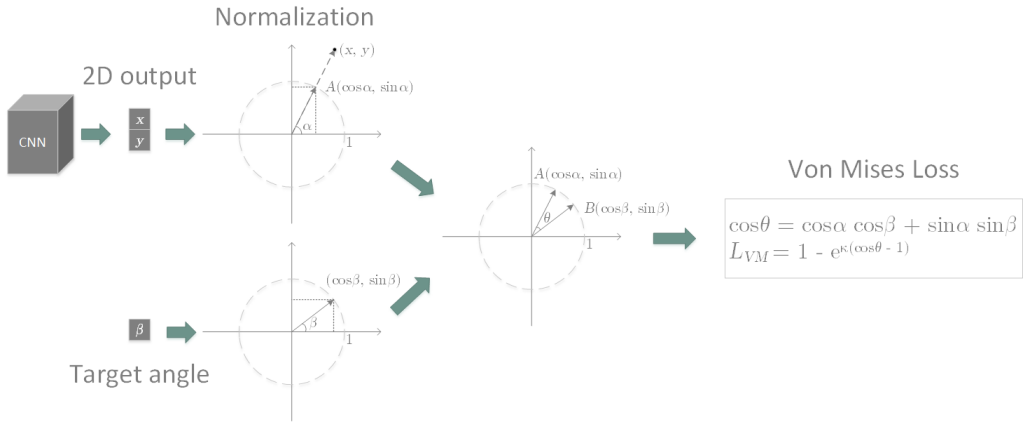


Figure 5-3: Orientation regression pipeline.

5.2.1 Normalization Layer

We derive the general form of normalization operation for an vector with any length in this section. Given an d-dimension vector $\mathbf{x} = (x_1, x_2, \ldots, x_d)$, the forward propagation is simply the normalization operation which is

$$\mathbf{y} = \frac{\mathbf{x}}{\|\mathbf{x}\|} = \frac{\mathbf{x}}{\sqrt{\mathbf{x} \cdot \mathbf{x}}}. \tag{5.2}$$

For backward propagation, we need to derive the partial derivative of loss with respect to each dimension of $\mathbf{x}$

$$\frac{\partial L}{\partial x_j} = \sum_{i=1}^{d} \frac{\partial L}{\partial y_i} \frac{\partial y_i}{\partial x_j}, \ j \in \{1, 2, \ldots, d\} \tag{5.3}$$

where $\frac{\partial L}{\partial y_i}$ comes from the successor layer. Now we derive the second term in above equation which is the partial derivative between each output and input variable:

$$\begin{aligned} \frac{\partial y_i}{\partial x_j} &= \frac{\partial}{\partial x_j}\left(\frac{x_i}{\sqrt{\mathbf{x} \cdot \mathbf{x}}}\right) \\ &= \frac{1}{\sqrt{\mathbf{x} \cdot \mathbf{x}}} \frac{\partial x_i}{\partial x_j} + x_i \frac{\partial}{\partial x_j} \frac{1}{\sqrt{\mathbf{x} \cdot \mathbf{x}}} \\ &= \frac{\delta_{ij}}{\sqrt{\mathbf{x} \cdot \mathbf{x}}} - \frac{x_i x_j}{\mathbf{x} \cdot \mathbf{x} \sqrt{\mathbf{x} \cdot \mathbf{x}}} \\ &= \frac{\delta_{ij} - y_i y_j}{\sqrt{\mathbf{x} \cdot \mathbf{x}}} \end{aligned} \tag{5.4}$$

where $\delta_{ij} = \begin{cases} 0, & \text{if } i \neq j \\ 1, & \text{if } i = j \end{cases}$ is the Kronecker delta. Substituting (5.4) into (5.3) we have

$$\begin{aligned} \frac{\partial L}{\partial x_j} &= \sum_{i=1}^{d} \frac{\partial L}{\partial y_i} \frac{\delta_{ij} - y_i y_j}{\sqrt{\mathbf{x} \cdot \mathbf{x}}} \\ &= \frac{1}{\sqrt{\mathbf{x} \cdot \mathbf{x}}} \sum_{i=1}^{d} \frac{\partial L}{\partial y_i} (\delta_{ij} - y_i y_j) \\ &= \frac{1}{\sqrt{\mathbf{x} \cdot \mathbf{x}}} \big(\frac{\partial L}{\partial y_j} - \sum_{i=1}^{d} \frac{\partial L}{\partial y_i} y_i y_j \big). \end{aligned} \tag{5.5}$$

Writing as vector form, we get the computation equation for backward propagation

$$\frac{\partial L}{\partial \mathbf{x}} = \frac{1}{\sqrt{\mathbf{x} \cdot \mathbf{x}}} \big(\frac{\partial L}{\partial \mathbf{y}} - (\frac{\partial L}{\partial \mathbf{y}} \cdot \mathbf{y}) \mathbf{y} \big). \tag{5.6}$$

5.2.2 Von Mises Loss Layer with Biternion Representation

In Section 4.4, the von Mises loss with Biternion representation is already given by

$$L_{VM}(\mathbf{y}|\mathbf{t}; \kappa) = 1 - e^{\kappa(\mathbf{y} \cdot \mathbf{t} - 1)} \tag{5.7}$$

which is the equation for forward propagation. Now we need to derive the computations for the backward propagation, which is obtained by computing the gradient of the loss L with respect to the layer parameters:

$$\frac{\partial L}{\partial y_k} = \frac{\partial L}{\partial L_{VM}} \frac{\partial L_{VM}}{\partial y_k}, \ k \in \{1, 2\}. \tag{5.8}$$

In back-propagation computation, $\frac{\partial L}{\partial L_{VM}}$ is known from the successor layer and the second term is derived as follows

$$\frac{\partial L_{VM}}{\partial y_k} = -e^{\kappa(\mathbf{y} \cdot \mathbf{t} - 1)} \kappa y_k. \tag{5.9}$$

So we get

$$\frac{\partial L}{\partial y_k} = -\frac{\partial L}{\partial L_{VM}} e^{\kappa(\mathbf{y} \cdot \mathbf{t} - 1)} \kappa y_k \tag{5.10}$$

For conciseness, we also write the above equation with subscriptions into a single one with vector form

$$\frac{\partial L}{\partial \mathbf{y}} = -\frac{\partial L}{\partial L_{VM}} e^{\kappa(\mathbf{y} \cdot \mathbf{t} - 1)} \kappa \mathbf{y} \tag{5.11}$$

5.2.3 Caffe Implementation of Layers

All the above layers are implemented with CPU version (C++) and GPU version (CUDA). We also create test scripts to check that the forward and backward implementation are in numerical agreement. For checking the forward pass, dummy input data is firstly created, and then forward function of the layer is called to calculate the outputs. The outputs are then checked with results externally calculated based on input data for agreement. And the backward pass is checked automatically by Caffe with finite difference method.

5.3 Orientation Classification

Additionally we introduce a orientation classification based network as the baseline for comparison. This network outputs probability distribution over 8 orientation classes followed with a mixture of von Mises distributions model to create a continuous distribution of orientation on a circle, inspired by [26]. The likelihood in terms of orientation class Ω of the current observation z,

$$p(z|\omega) = \sum_{\Omega} p(z|\Omega)p(\Omega|\omega). \tag{5.12}$$

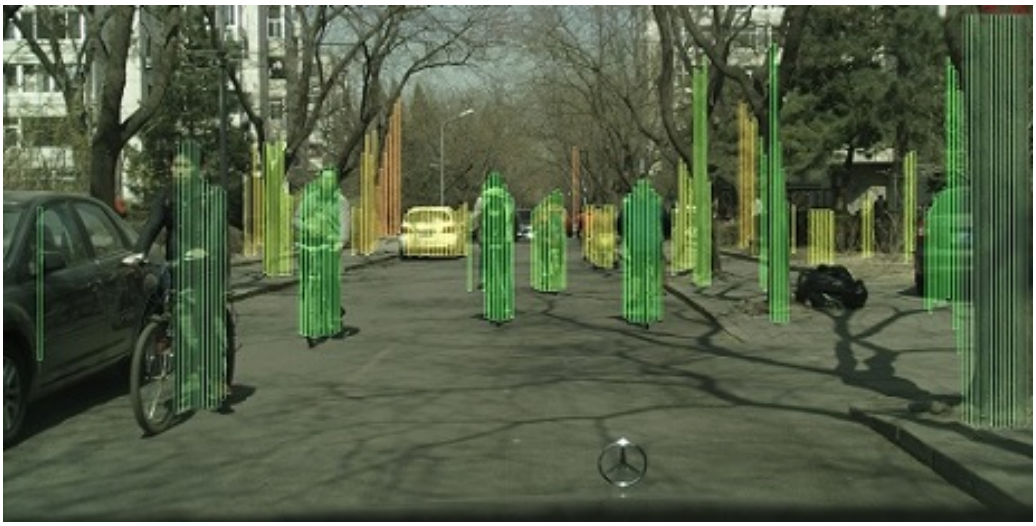
The term $p(z|\Omega)$ can be defined as the classifier's probability output over discrete orientation classes. And the second term $p(\Omega|\omega)$ expresses the probabilistic relationship between the continuous orientation and angle ω and discrete class Ω, which is obtained by Bayes' rule

$$p(\Omega = o|\omega) = \frac{p(\omega|\Omega = o)p(\Omega = o)}{\sum_{k\in\Omega} p(\omega|\Omega = k)p(\Omega = k)}. \tag{5.13}$$

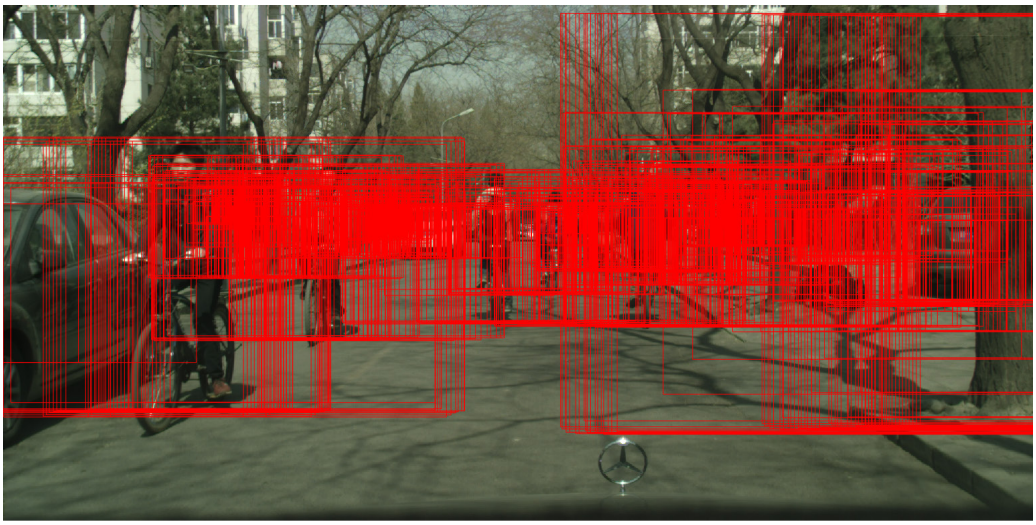
Here $p(\Omega)$ is a prior on discrete class. Since we don't have particular prior, we set it as uniform distribution. And $p(\omega|\Omega = o)$ is the von Mises distribution $\mathcal{V}(\omega; c_o, \kappa_o)$ with c_o and κ_o the mean and concentration of the distribution for orientation class o.

5.4 Stixel-based Proposal Generation

In our test car environment, there are multiple sensors available which can provide information other than using monocular images. Unlike the original R-CNN which uses Selective Search or Region Proposal Networks for extracting object proposals, our work utilize stereo data for proposal generation based on stixel representation [22].



(a) Stixels



(b) Proposals

Figure 5-4: The stixels (a) after applying the defined constraints. For each stixel, multiple proposals with different aspects are then generated (b) and will be used as the object proposal input of our Pose-RCNN.

The Stixel World

The stixel world is an image based compact medium-level representation of the 3D environment. Stixels are vertically oriented rectangles with a fixed width (e.g., 5px) and variable heights, adjacently aligned over objects in an image. The stixels are generated from stereo image pairs as proposed in [88]. This representation allows for an enormous reduction of raw input data, e.g. approximately 2 billion disparity measurement from a 2048×1024 px stereo image pair are reduced to a few hundred stixels only.

Proposal Generation

Given stixels, RoI proposals are generated with the method proposed in [22]. The working principles of proposal generation is closely related to prior knowledge of 3D scene and target objects. The first assumption is that objects are ground-based. The vertical position of proposals are based on the planar ground model of stixel world and hence the proposals are aligned to the stixel bottom location in the image. Secondly, by combining the knowledge of 2D target geometry, the distance of stixels to the camera can be estimated. In our work, stixels in distance range $[4m, 100m]$ are considered, which covers the vulnerable situations that we are interested in. Thirdly, the prior knowledge of cyclist/pedestrian shape are considered, e.g. aspect ratios. Thus for each stixel, we sample three proposal aspects (1:2, 2:3, 1:1) with the same height as the stixel. Each proposal is also jittered in size and shifted up/down/left/right relatively to the proposal size and location to get more proposals. Furthermore, we estimate the objects' height with their scale and distance information. Since pedestrian/cyclist's height is normally in range $[1.2m, 2.4m]$, we first choose the stixels with estimated hight in $[1.2m, 2.4m]$ for sampling proposals. And for stixels higher than $2.4m$, we also sample the proposal height in $[1.2m, 2.4m]$ with a step size of 0.3m (because of quantization of stixels, e.g. pedestrians standing near a wall will be over-smoothed and overseen otherwise).

Figure 5-4a shows the stixels after applying the above constraints and Figure 5-4b shows the generated proposals. The stixel-based proposals generates 18/4002/815 (min/max/average) proposals per image in the dataset we use.

5.5 R-GoogLeNet

To gain better performance, we choose a second network model developed by Szegedy *et al.* called GoogLeNet [3] which was the winning architecture of ILSVRC14. It has 12× less param-

eters than the winning algorithm of ILSVRC12 AlexNet [12] but significantly better accuracy. The original GoogLeNet is made of three regular convolutional layers close to the input followed by nine inception modules [3] each consisting of six convolutional layers and several max pooling layers. As stated in their original work, given the relatively large depth of network, the efficiency of back-propagating through all the layers is an issue. They resolve this problem by adding auxiliary classifiers connected to intermediate layers.

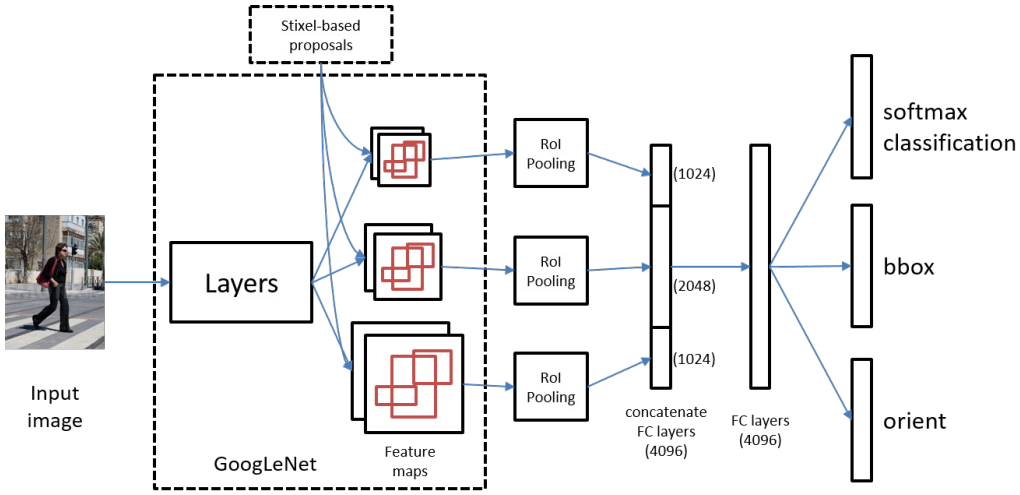


Figure 5-5: Overview of R-GoogLeNet architecture.

Inspired by there approach, we proposed the variant of GoogLeNet called R-GoogLeNet which use intermediate layers for generating feature maps separately. The overview of architecture is illustrated in Figure 5-5 For each region of interests (RoI), the RoI pooling [1] and first stage of fully connected mapping are done independently on each feature map. The vector output from each feature map are then concatenated to a single vector followed by the second stage of fully connected mapping to get the single RoI feature vector. Each feature vector is then branched into three sibling output layers as in Pose-RCNN: softmax probabilities, per-class bounding-box regression and per-class orientation regression. The multi-task loss function remains the same defined in Equation (5.1) for the joint detection and orientation regression task. And detailed layer structure is shown in Figure 5-6.

Moreover, we want to know which of the three intermediate feature maps provides the most informative features for prediction, we developed three more network architectures which are substructures of the R-Google named R-GoogLeNet$_1$, R-GoogLeNet$_2$ and R-GoogLeNet$_3$. Instead of using all three feature maps, each of the above networks only use one of those feature maps. The three networks structures are shown in Figure 5-7.

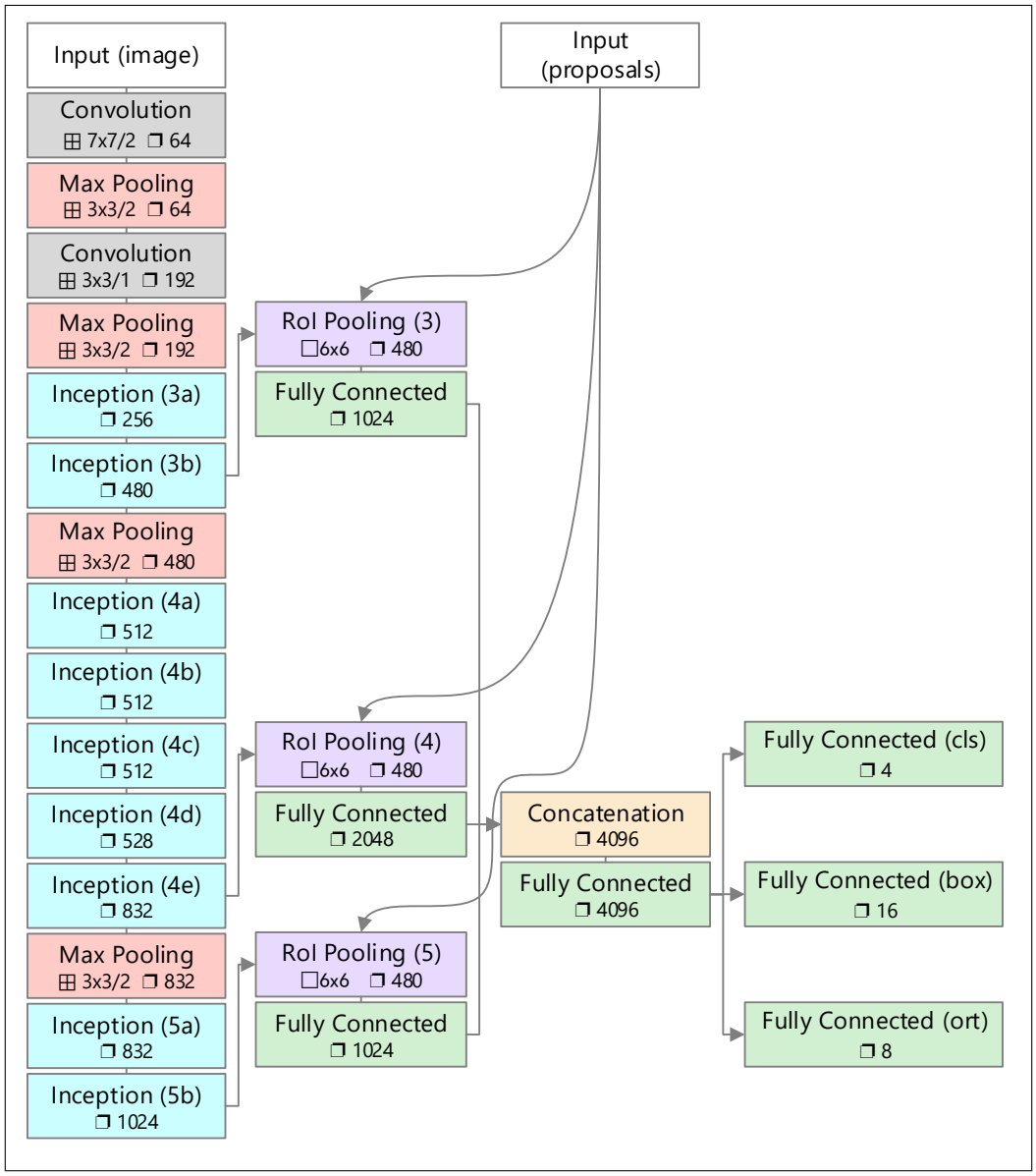
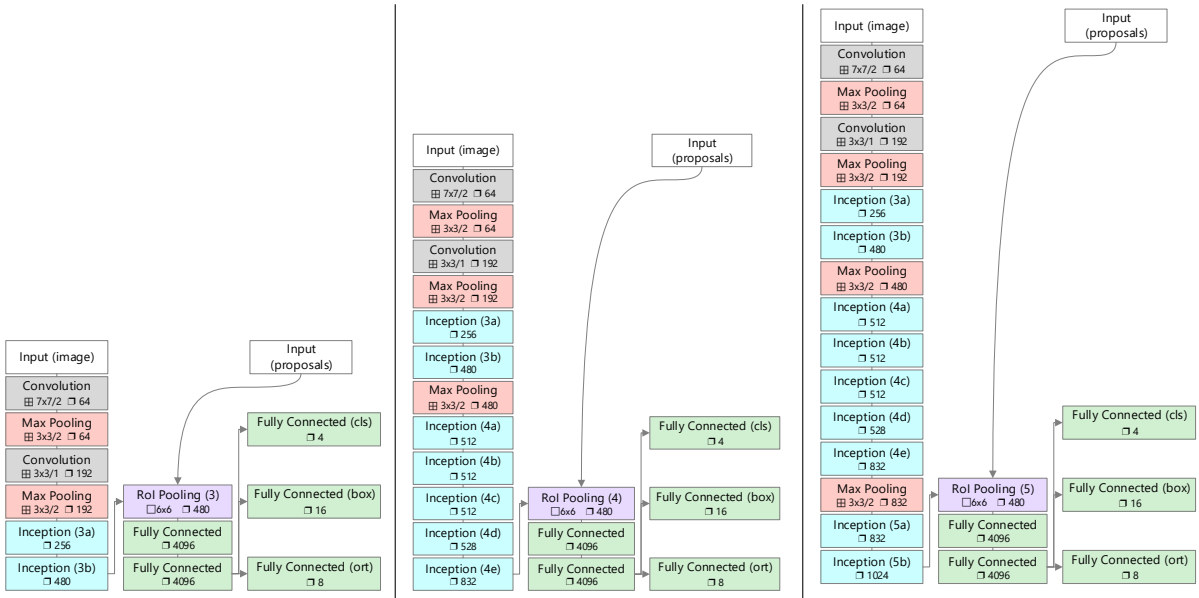


Figure 5-6: R-GoogLeNet architecture. An input image and multiple regions of interest (RoIs) proposals are input into the network. The RoI pooling takes place on three intermediate feature maps simultaneously. The pooled feature maps are fully connected to the first level of feature vectors separately and then concatenated as a single feature vector. The vector is then mapped by another FC layers followed by three sibling FC layers to generate the output of the network per RoI:softmax probabilities, per-class bounding-box regression offsets and per-class orientation regression (⊞ denotes the size of a layer's kernel and stride, ☐ is the RoI Pooling layer width and height, and ❒ gives the depth of layers).



(a) GoogLeNet$_1$ (b) GoogLeNet$_2$ (c) GoogLeNet$_3$

Figure 5-7: R-GoogLeNet$_1$, R-GoogLeNet$_2$ and R-GoogLeNet$_3$ architectures. Instead of using all three feature maps, each of them only uses one feature map for RoI pooling. Without concatenating vectors from different feature maps, RoI pooling layer is directly connected to two 4096-dimensional FC layers followed by sibling FC output layers.

5.6 Object Parts Regression

Head orientation of a cyclist plays an important role in risk assessment for the environment perception system of cars. By estimating the head orientation, the system could know if the person has noticed the coming car or not, and thus making more accurate risk assessment. So we are going to extend R-GoogLeNet and apply the bounding-box regression and orientation regression for parts, namely head and bike separately. We name these category of networks Parts-RCNN and two approaches are proposed in this section.

5.6.1 Parts-RCNN

In the first approach, we assume that the pooled feature map of the proposal box for full object contains essential information of both head and body. It means that we could directly do the regression of head and body bounding-box and orientation from the proposed full object features. The network architecture is based on R-GoogLeNet shown in Figure 5-6. The feature vector generation of proposals is the same as R-GoogLeNet. The feature vector is then fully connected to six sibling outputs: softmax probabilities, per-class full object bounding-box offsets, per-class head/body bounding-box offsets and orientation predictions. The multi-task

loss function therefore combines all the above six tasks and is defined as:

$$\begin{aligned} L = & L_{obj\_cls} + \lambda_{obj} L_{obj\_box} \\ & + \lambda_{head\_box} L_{head\_box} + \mu_{head\_orient} L_{head\_orient} \\ & + \lambda_{body\_box} L_{body\_box} + \mu_{body\_orient} L_{body\_orient} \end{aligned} \tag{5.14}$$

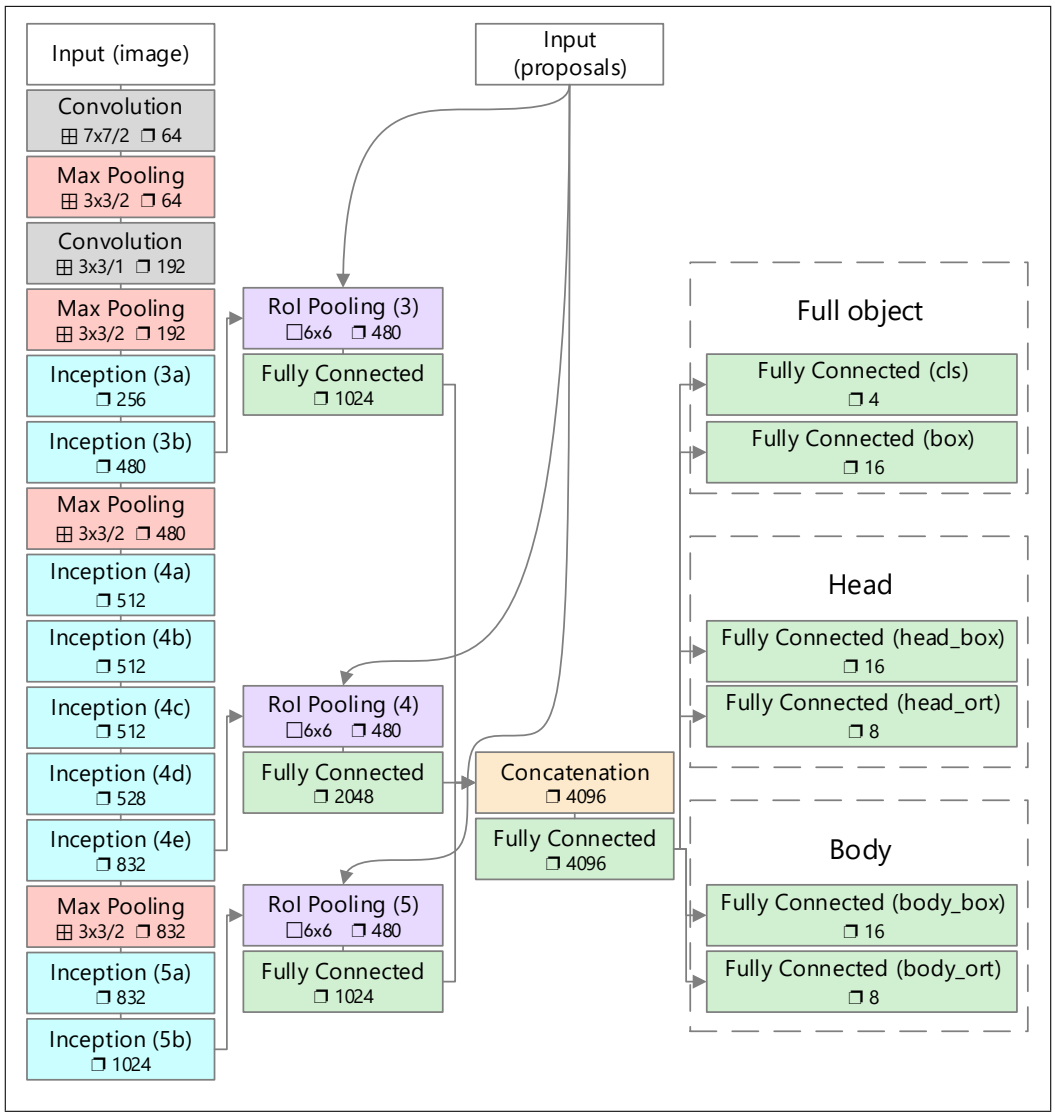
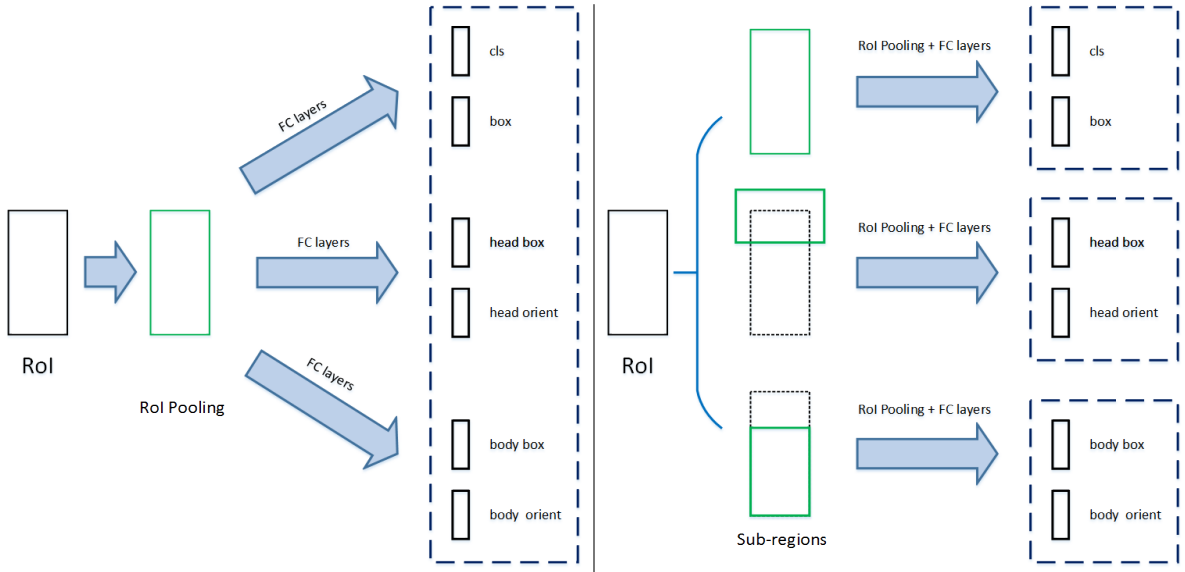


Figure 5-8: Parts-RCNN based on R-GoogLeNet. The feature vector generation is the same as R-GoogLeNet. The multi-task network contains six outputs: softmax probabilities, per-class full object bounding-box offsets, per-class head/body bounding-box offsets and orientation predictions.

In the first approach, we assign six tasks to only one feature vector which comes from the full object proposal, see Figure 5-9a. However it is not necessary to make all the predictions from a single feature vector. We consider using regions that possibly contain parts solely for corresponding parts estimation. According to the geometric structure of cyclist, we assume

that the head appears in the upper part of an bounding-box and the body appears in the lower part. Thus, we limit the search area for parts on specific sub-regions of the original proposed box. This bounding-box splitting technique is illustrated in Figure 5-9b. We implement this by extending the RoI pooling layer in Caffe so that it supports multiple sub-region definitions and that it RoI pool these sub-regions into separate pooled feature maps. Then the feature vectors for the same sub-regions are concatenated and connect to specific task's outputs. In this case, the full box is responsible for classification and full-object bounding-box regression, upper part for head bounding-box and orientation, lower part for body.



(a) Parts regression with full-proposal bounding-box. The proposal is assigned with all the tasks including full object and parts.

(b) Bounding-box splitting architecture. Each proposal box is split to three sub-regions (including original one) for specific tasks.

Figure 5-9: Parts regression with full-proposal bounding-box (a) and with split-proposal bounding-box (b).

Chapter 6

Experiments and Results

This chapter first gives the introduction of the experiment settings including our test vehicle environment and dataset. The dataset we use comes from Tsinghua-Daimler Cyclist Benchmark[4]. Next we are going to give the evaluation metrics used in our experiments. In the last part, we are going to describe the training settings and results for each experiment we have conducted.

6.1 Environment and Dataset

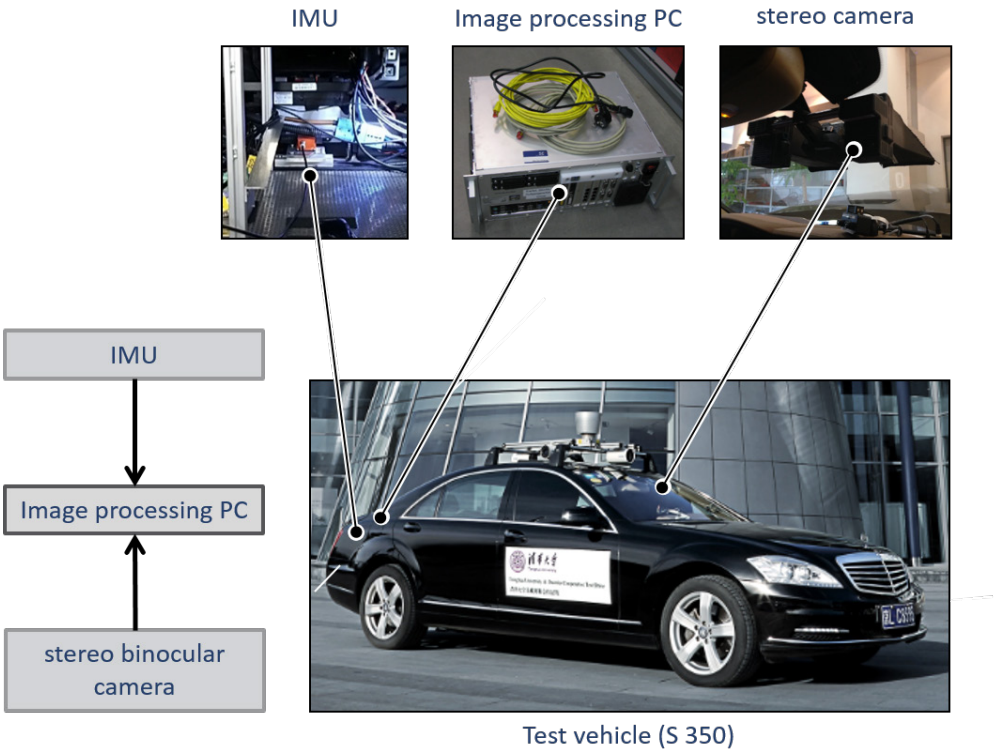


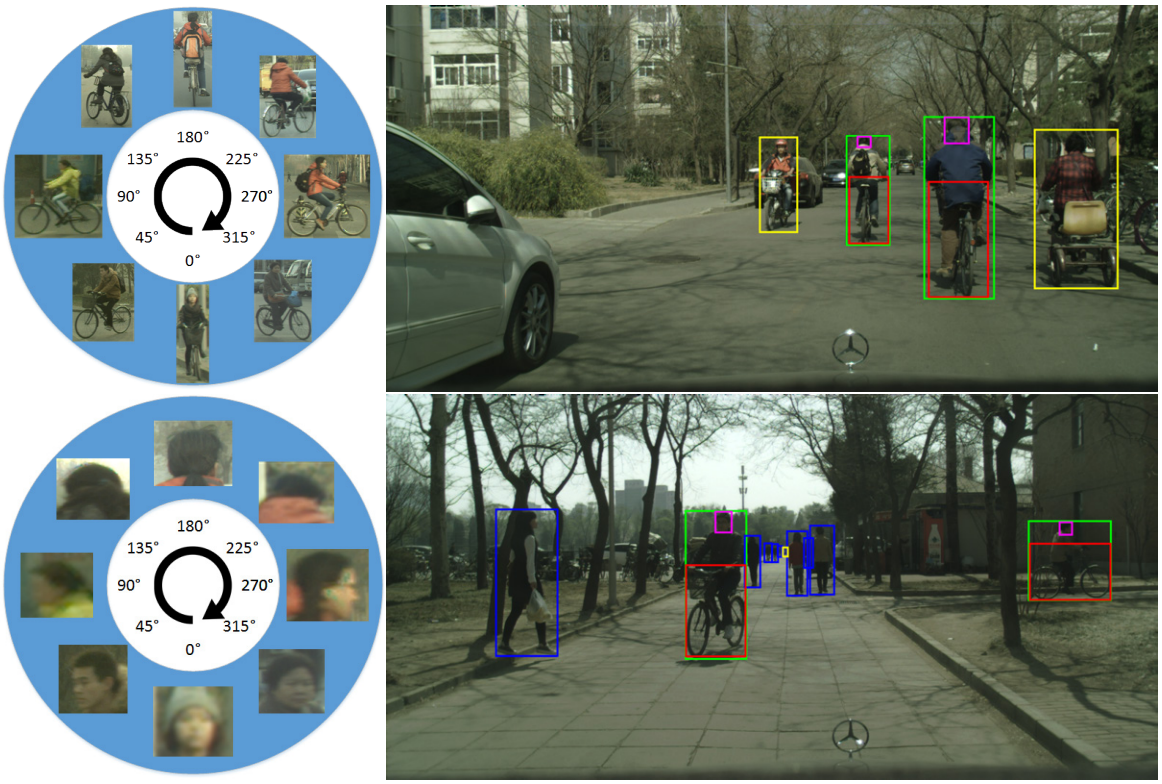
Figure 6-1: Data recording platform.

6.1.1 Recording Platform

Data recording vehicle is based on a Mercedes-Benz S 350 car which is equipped with stereo cameras and inertial measurement unit (IMU) for data collection, PC for data processing (Illustrated in Figure 6-1). The stereo camera is mounted on the inside of the front windshield with baseline 20cm and horizontal field of view (FOV)±23°, recording image sequences with reolution of 2048 × 1024 pixel at 25Hz. The IMU collects information, such as velocity, pitch, roll and yaw.

6.1.2 Dataset

The video sequences are recorded by driving through regular urban and campus traffic in Beijing. In total there are more than 5 million images, in which about 14674 frames are labeled, containing a total of 32361 vulnerable road users (VRUs) including cyclists, pedestrians, tricyclists and motor-cyclists etc. The dataset is divided into four clusters: training, validation, none-VRU and testing.



(a) (b)

Figure 6-2: (a) Cyclists and head orientation domain and samples in dataset; (b) Images with bounding-box annotations: green (cyclist), blue (pedestrian), yellow (other rider), red (bike of cyclist), purple (head of cyclist).

The training set is partly labeled, in which only ideal cyclists were annotated with three tight

bounding boxes, indicating the full extent of the cyclist, bike and head of the rider respectively (See Figure 6-2b). Among the three boxes, bike and head boxes are also labeled with 2D orientation in domain [0, 360) shown in Figure 6-2a. Cyclists lower than 60 pixels, occluded or truncated more than 10% or heavily motion blurred were ignored. Other objects are ignored in this part.

The training and validation set are fully labeled. All objects which are higher than 20 pixels, not occluded more than 80% and not truncated more than 50% are all annotated with tight bounding boxes indicating the full extent of the object. Tri-cyclists, motor-cyclists and mopped are additionally labeled with bike bounding box. Cyclists are additionally labeled with head and bike bounding boxes and orientations (See Figure 6-2b). Three discrete occlusion level are tagged to all objects, which are no occlusion (occlusion$\leq$10%), partially occluded (10%$<$occlusion$\leq$40%) and heavily occluded (40%$<$occlusion$\leq$80%). Table 6.1 shows the detailed statistics.

	Training	Validation	Non-VRU	Test	Total
Total Frames	9741	5095	1000	14570	30406
Labeled Frames	9741	1019	1000	2914	14674
Total BBs	16202	3016	0	13143	32361
Cyclist BBs	16202	1301	0	4658	22161
Pedestrian BBs	0	1539	0	7380	8919
Other rider BBs	0	176	0	1105	1281

Table 6.1: Statistics of Tisinghua-Daimler cyclist benchmark[4].

We use the training set in Table 6.1 for training. We evaluate the full-cyclist detection and orientation performance on test set. Due to the limit of time, we only evaluate the parts regression tasks on validation set. After checking the dataset files, we found little discrepancy with the dataset description from the original paper. Table 6.2 show the statistics of part we use. Figure 6-3 illustrate the cyclist bounding-box distributions of the dataset that we use.

Noticing that the object has different scales depending on their distance from the camera, we want to evaluate the our models' performance with different objects scales. Thus we also group the test dataset by object scales as following: $\leq$7.5m($\geq$281px) / 7.5m-12.5m(168px-281px) / 12.5m-17.5m(120px-168px) / 17.5-22.5m(94px-120px) / 22.5m-27.5m(77px-94px) / $\geq$27.5m(0px-77px). In each group, the numbers outside brackets indicate the distance range in meter, and the numbers in brackets indicate the corresponding cyclist height in pixel. Figure 6-5 shows the distributions of labeled cyclists in different scales.

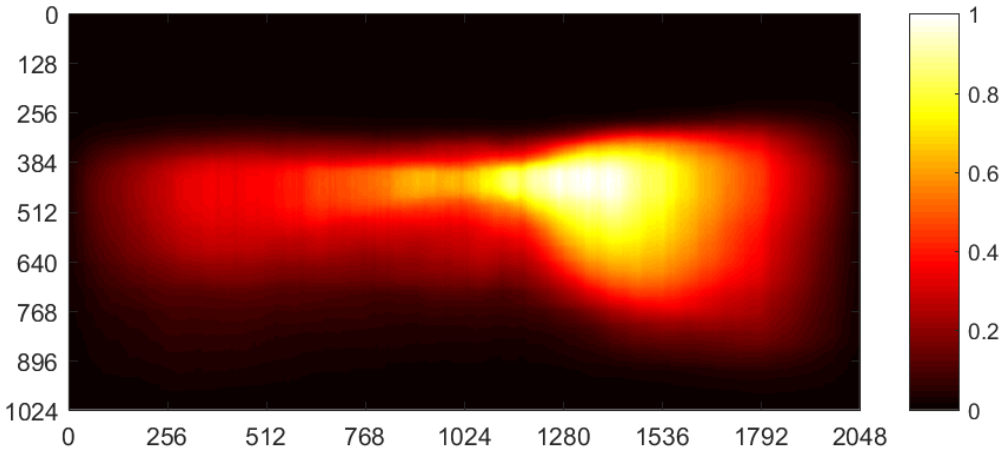


Figure 6-3: Cyclist bounding-box distributions in dataset.

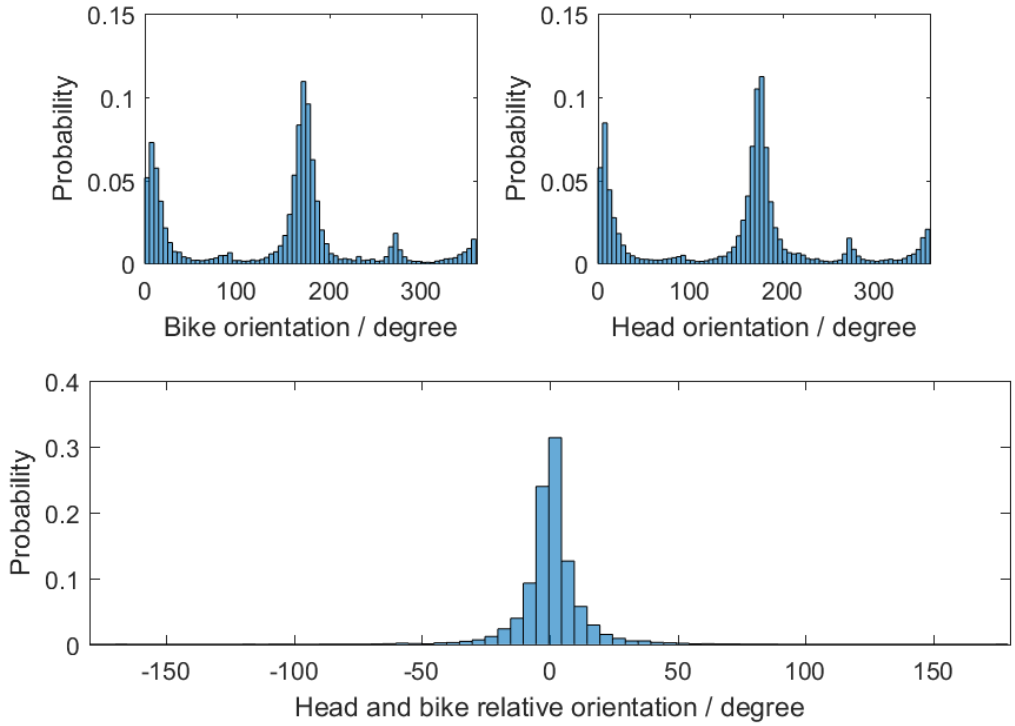
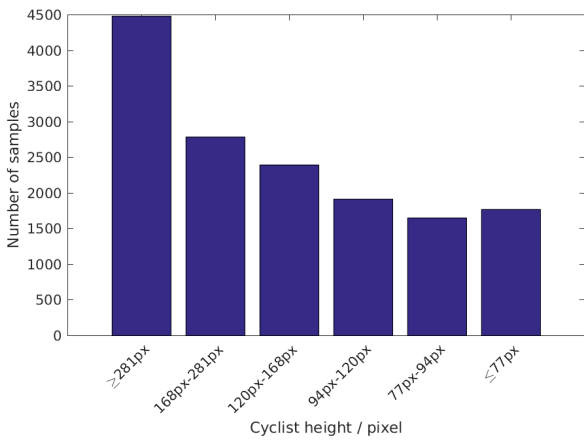
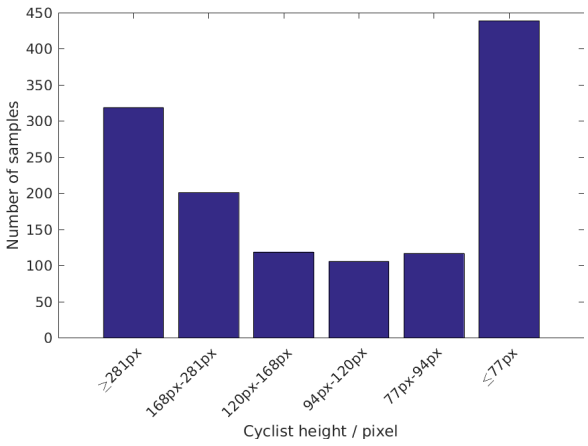


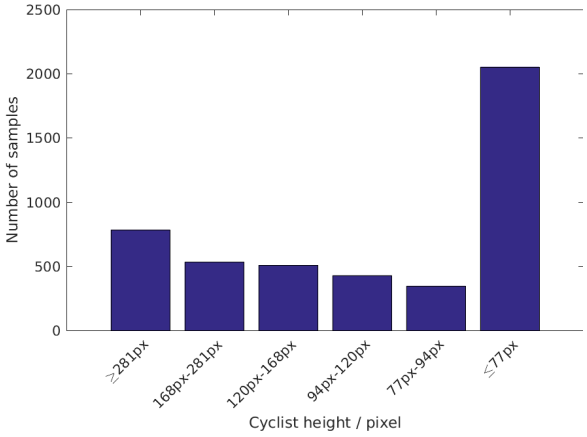
Figure 6-4: Histogram of orientation of the cyclist bounding-boxes in dataset. The width of each bin is 5°.



(a) Histogram of labeled cyclist scales in training set.



(b) Histogram of labeled cyclist scales in validation set.



(c) Histogram of labeled cyclist scales in test set.

Figure 6-5: Histogram of labeled cyclist scales in training, validation and test sets.

Dataset	Fully Labeled	Frames	Cyclist	Pedestrian	Other Rider
Training	✗	9037	15001	0	0
Validation	✓	1019	1301	1539	176
Test	✓	2916	4658	7380	1105

(a) Dataset statistics.

Category	Full BB	Head BB	Body BB	Orientation
Cyclist	15001	15001	15001	✓
Pedestrian	0	0	0	✗
Other Rider	0	0	0	✗

(b) Labeling status of training set.

Category	Full BB	Head BB	Body BB	Orientation
Cyclist	1301	1007	1301	✓
Pedestrian	1539	0	0	✗
Other Rider	176	0	176	✗

(c) Labeling status of validation set.

Category	Full BB	Head BB	Body BB	Orientation
Cyclist	4658	3058	4658	✓
Pedestrian	7380	0	0	✗
Other Rider	1105	0	1105	✗

(d) Labeling status of test set.

Table 6.2: Dataset statistics and labeling status. For riders, 'body' refers to the vehicle that the person is riding, including bike, tricycle, motorcycle and mopped. Some cyclist bounding-boxe does not contain head bounding-box labeling, because of occlusion, unclear object and manually miss labeling.

6.2 Evaluation Metrics

This section is arranged as two parts, detection evaluation and orientation evaluation. For each of them, we are going to give the evaluation metrics that we use for our experiments.

6.2.1 Detection Evaluation

In this part we first explain how rectangle boxes are compared, based on which we give the metrics for full object detection and parts detection.

Intersection over Union

In literatures related to object detection, the criterion of comparing two object-areas (not necessarily rectangle) is usually defined as Intersection over Union (IoU) given by $IoU = \frac{A \cap B}{A \cup B}$, where A and B are two areas. The definition of an 'hit' detection is determined by the IoU with a threshold (for example 0.5).

Precision and Recall

When evaluating models for binary classification on a given dataset of positive and negative samples, usually four types of data are defined: true positive (TP), true negative (TN), false positive (FP) and false negative (FN), whereas true and false refer to whether the positives or negatives are correctly classified by the model. For multi-class classification, the negative samples of one category refers to all the other classes. With the numbers of TP, TN, FN and FN , we adopt the classic precision-recall metrics defined as:

$$Precision = \frac{TP}{TP + FP} \tag{6.1}$$

$$Recall = \frac{TP}{TP + FN}. \tag{6.2}$$

Basically, Precision (P) represents the fraction of correctly predicted instances among all retrieved instances. It is an measurement of the correctness of the model output. Recall (R) represents the fraction of correctly predicted instances among all positive instances which should be retrieved. It measures the instance retrieving ability of the model. In our experiments, we require true positives to IoU-overlap by more than 0.5 and count multiple detections of the same object as false positive. For the task of full-object detection, we calculate the average precision (AP) of different models as the single value character for comparison. The AP is defined as precision averaged across all values of racall between 0 and 1 :

$$AP = \int_0^1 p(r)dr \tag{6.3}$$

where r is recall and $p(r)$ is the precision at recall r. We took the traditional 11-point approach for drawing precision-recall curve, where the precision is measured at the 11 recall levels of $0.0, 0.1, \cdots, 1.0$. And the average precision is approximated by averaging over the 11 points:

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \cdots, 1\}} p(r). \tag{6.4}$$

Mean IoU

For bounding-box regression tasks, we only take the true positive detections for comparison. When several detections are assigned to the same ground-truth, only the one with highest class probability is considered. We choose to calculate the mean IoU for bounding-boxes of true positive detections (IoU with ground-truth more than 0.5). And we show the mean IoU curve by grouping objects by their scales.

6.2.2 Orientation Evaluation

For orientation evaluation, we also consider two parts: the evaluation for full-object orientation and parts orientation.

Orientation Similarity

For full-object orientation evaluation, we choose orientation similarity (OS) and average orientation similarity (AOS) [89] which combines recall with orientate error measurement. The definition of orientation similarity takes the similar approach of 11-point Interpolated Average Precision[90]. For a given recall rate r orientation similarity is defined as the highest similarity found for any recall level $\tilde{r} \geq r$:

$$OS(r) = \max_{\tilde{r}:\tilde{r} \geq r} s(\tilde{r}) \tag{6.5}$$

where $s \in [0, 1]$ is a normalized variant of cosine similarity at recall r defined as

$$s(r) = \frac{1}{|\mathcal{D}(r)|} \sum_{i \in \mathcal{D}(r)} \frac{1 + \cos \Delta_\theta^{(i)}}{2} \delta_i \tag{6.6}$$

and $\mathcal{D}(r)$ denotes the set of all object detections at recall rate r and $\Delta_\theta^{(i)}$ is the difference in angle between estimated and ground-truth orientation of detection i. δ_i is a penalize multiplier when multiple detections are assigned to a single ground-truth. $\delta_i = 1$ if and only if detection i is assigned to a ground-truth bounding box and it's the one with highest IoU among all the detections that are assigned to the same bround truth bounding box. And $\delta_i = 0$ otherwise. And average orientation similarity is defined as orientation similarity averaged across all values of racall between 0 and 1:

$$AOS = \int_0^1 OS(r) dr. \tag{6.7}$$

We take the same 11-point recall level approach for drawing AOS-precision curve to illustrate the performances of different models. And in this case, the average orientation similarity is approximated by averaging over 11 recall levels:

$$AOS = \frac{1}{11} \sum_{r \in \{0, 0.1, \cdots, 1\}} OS(r). \tag{6.8}$$

Angle Error

We also calculate the mean absolute angle error of true positive detections for evaluation. When several detections are assigned to the same ground-truth, only the one with highest class probability is considered. Firstly, the mean absolute angle error for each part is calculated. Similar to mean IoU, we also group the testing objects by scale/distance and plot the curve for showing the performance on different object scales. Furthermore, we adopt boxplot and confusion matrix to illustrate the angle errors.

A boxplot is a convenient way of graphically depicting groups of numerical data through their quartiles. In our boxplots, central marks show the median (Q_2) of angle errors, and the bottom and top edges of the boxes indicate the 25th and 75th percentiles (Q_1 and Q_3). The boxes contain 50% of samples. And the wiskers defines 99.3% coverage of data, which extend to the most extreme data points not considered outliers. Samples are dawn as outliers if they are greater than $Q_3 + 1.5 \times IQR$ or less than $Q_1 - 1.5 \times IQR$, where $IQR = Q_3 - Q_1$. Figure 6-6 illustrate the structure of a boxplot.

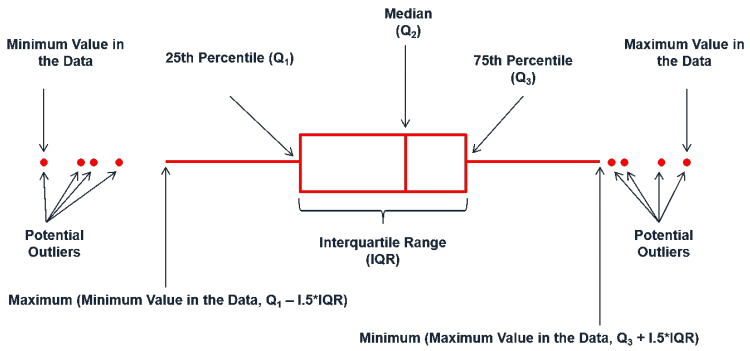


Figure 6-6: Boxplot structure.

A confusion matrix is a specific table visualizing the classification performance of an algorithm. Each column of the matrix represents the predicted class of samples and each row represents the ground-truth class. We adopt confusion matrices to show the predicted orientation distribution with respect to their ground-truth orientation. Since confusion matrix is based on classification resutls, we make discretization of the orientation domain into k ori-

entation classes, each class covers $\frac{1}{k}$ degrees. The samples are then distributed into different ground-truth and predicted orientation classes according to their ground-truth and predicted orientations. Figure 6-7 illustrate a confusion matrix with 9 classes.

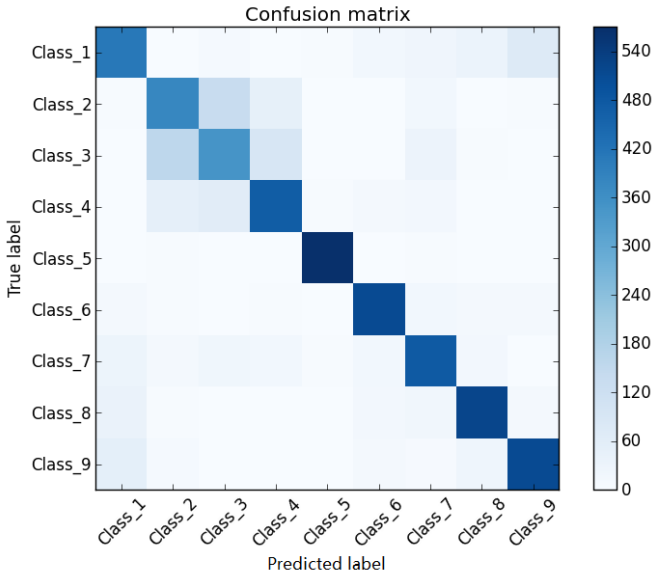


Figure 6-7: Confusion matrix.

6.3 Experiments

6.3.1 Cyclist Detection

The first step of our experiments is applying ZF-Net and R-GoogLeNet on our cyclist dataset and evaluating the feasibility of these two migrations. In this case, we only evaluate the detection performance of our networks without orientation regression. Both models take full size images (2048×1024pixels) as input. For object proposal, we use stixel-based proposal generation method (see Section 5.4). We also flip the training images and object proposals to double the size of training set.

For training of the two models, the public pre-trained ZF-Net and GoogLeNet are applied for initialize the network weights. The ZF-Net is pre-trained by Ren *et al.* at Microsoft Research Asia (MSRA)[91] and GoogLeNet comes from the Berkeley-trained models in Caffe Model Zoo[92]. During fine-tuning, the final sibling layers are adapted to this detection task. Each training batch is constructed from two images and contains 128 mini-batches which are random sampled from RoI inputs with positive and negative ratio 1:3. We use ground-truth boxes plus proposed boxes whose $IoU \geq 0.7$ with a ground-truth object as positive samples. Proposed boxes whose $IoU < 0.3$ are used as negative samples. For both models we set the

base learning rate as 1e-3, and it reduce by half every 10000 iterations. In total we train 100000 iterations for all the models.

During test phase, we discard all the other object categories and treat them as background. The detections whose IoU with ground-truth is at least 50% are considered as positive. In order to evaluate performance on various subsets of the dataset, similar to [89], we define three dificulty levels as follows for testing:

Difficulty	Cyclist height	Occlusion
Easy	≥60pixels	≤10%
Moderate	≥45pixels	≤40%
Hard	≥30pixels	≤80%

Table 6.3: Difficulty definitions of dataset.

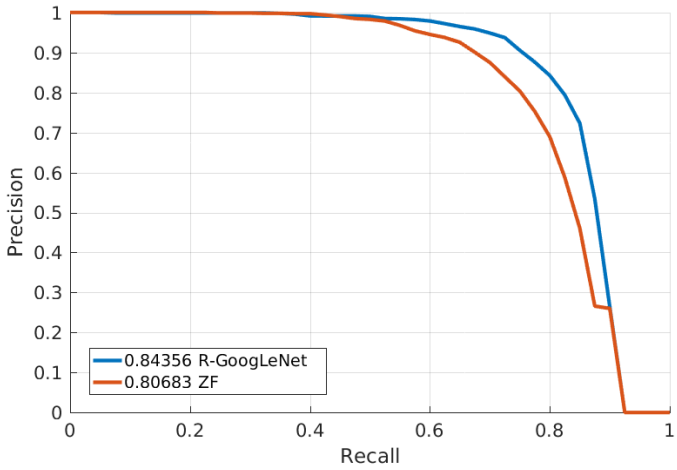
The precision-recall curves with average precisions are shown in Figure 6-8 and mean IoU of true positive detections is shown in Figure 6-9.

From the above figure, we see that the ZF-Net achieves AP of 0.806/0.647/0.537 for easy/moderate/hard cases, while the R-GoogLeNet over-perform it by around 0.04 with 0.844/0.685/0.571. And the precision-recall curves shows that R-GoogLeNet has in general higher precision than ZF-Net with different recall. On the other hand, though ZF-Net has lower precision, its mean IoU for true positive samples are slightly better than R-GoogLeNet.

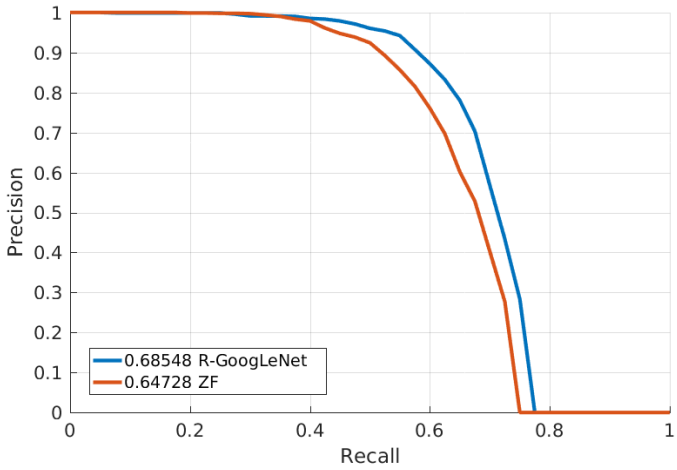
6.3.2 Cyclist Orientation Regression

In the second series of experiments we combine orientation regression to R-CNN based networks. We adapt the sibling network output with one more orientation regression output. For training and testing these networks, we use the same settings as in previous subsection.

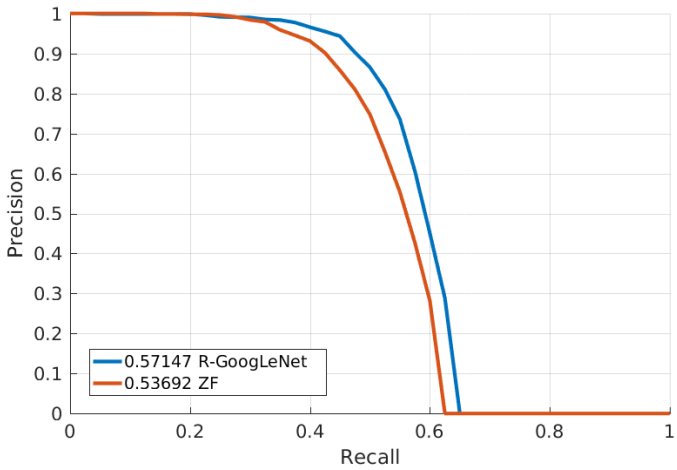
Firstly, we want to see how the networks performances are affected when introducing orientation regression task. In this case, we compare the cyclist detection performance of four networks: ZF-Net without orientation regression, ZF-Net with orientation regression, R-GoogLeNet without orientation regression and R-GoogLeNet with orientation. The precision-recall curves are shown in Figure 6-10. We notice that the introducing of orientation regression leads to a performance drop for both network architectures. The reason for this is that when adding new task to an existing network, the new task shares the feature maps with existing tasks. And during training, the back-propagated errors from different tasks are added up and propagated through the shared feature maps. And thus the errors are affecting each other, resulting lower performance for any single tasks under the same training parameter settings. And the curves



(a) easy



(b) moderate



(c) hard

Figure 6-8: PR-curves and AP of ZF-Net and R-GoogLeNet for cyclist detection task.

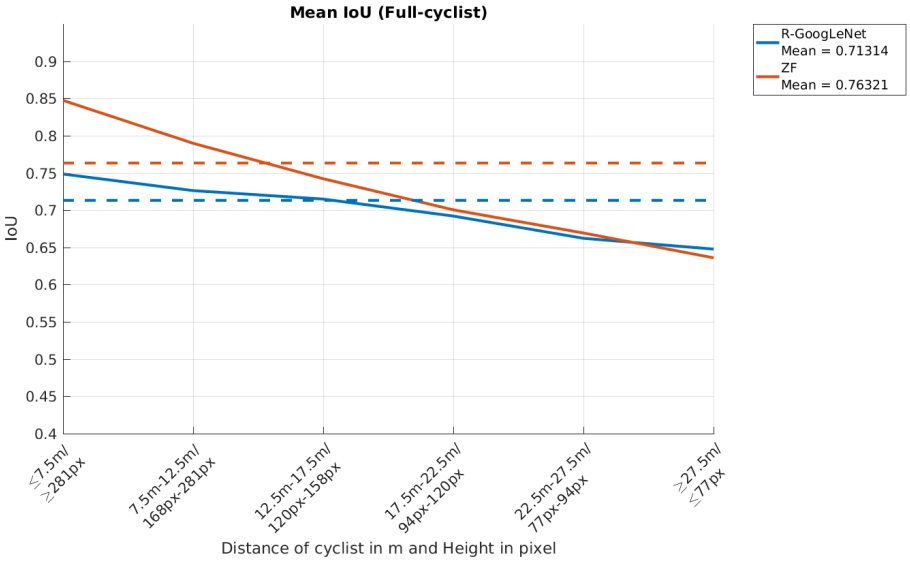


Figure 6-9: Mean IoU of true positive cyclist detections. The dashed lines shows the mean IoU of the whole test dataset.

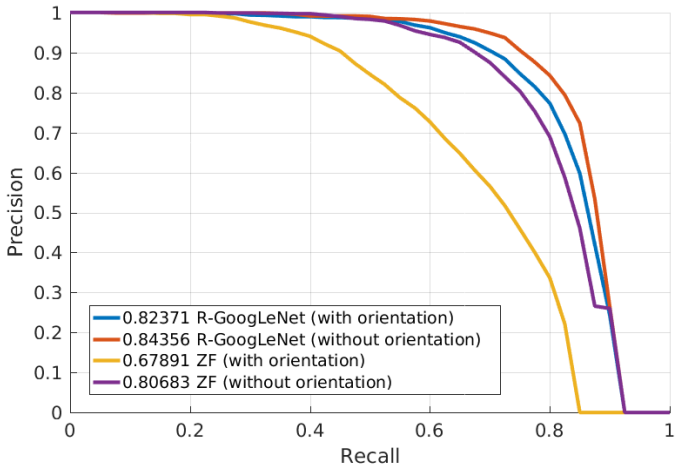
also show that the introducing of orientation regression has stronger impact on ZF-Net than R-GoogLeNet. It means that R-GoogLeNet is more robust than ZF-Net when assigning new tasks to it.

Secondly, we are going to compare the orientation regression performance of different network architectures. For training R-GoogLeNet$_1$, R-GoogLeNet$_2$ and R-GoogLeNet$_3$ (as illustrated in Figure 5-7), since they are all parts of the original GoogLeNet, we can still use the same pre-trained model to initialize their weights.

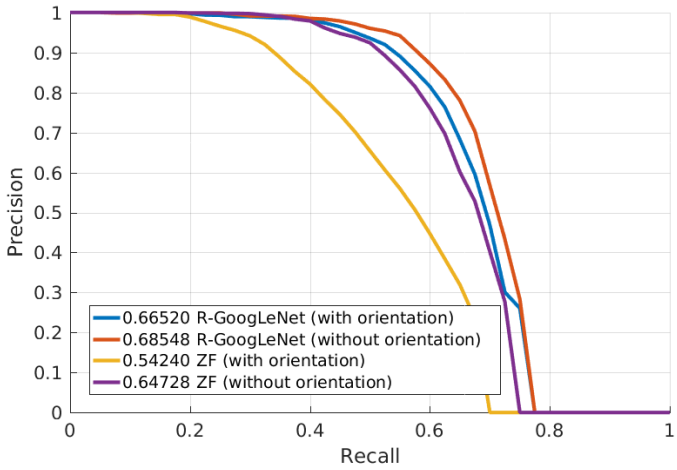
For training the classification network introduced in Section5.3, we discretize the continuous orientation labels into 8 orientation classes and use these classes plus one background class for training the new network. The network is based on ZF-Net version of Fast R-CNN. The other training parameters are the same as other networks in our experiments.

The Precision-Recall curve are shown in Figure 6-11 and AOS-Recall curve are plotted in Figure 6-12. The figures show that our complete version of R-GoogLeNet in general higher AP than the other models. When comparing the curves of the R-GoogLeNet based networks, we notice that R-GoogLeNet, R-GoogLeNet$_1$ shows quite competitive results while R-GoogLeNet$_3$ which only use the deepest convolutional layer as feature map performances significantly worse than the other models. Figure 6-13 shows the results of true positive detections under different object scales. ZF-Net and classification based ZF-Net shows quite good IoU for true positive detections, however has quite large mean absolute angle error. R-GoogLeNet and R-GoogLeNet$_1$ are still quite competitive in orientation estimation.

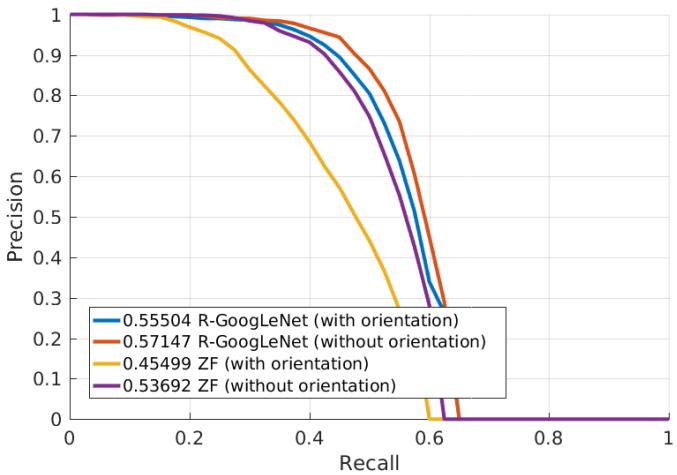
Figure 6-14 shows the confusion matrices of cyclist orientation estimation. The 360°domain



(a) easy

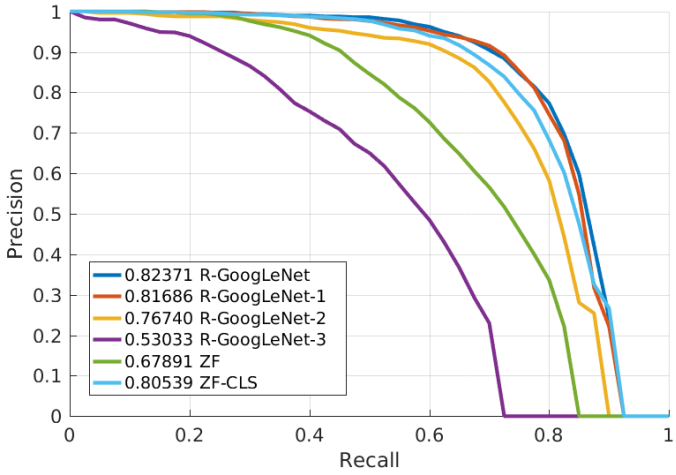


(b) moderate

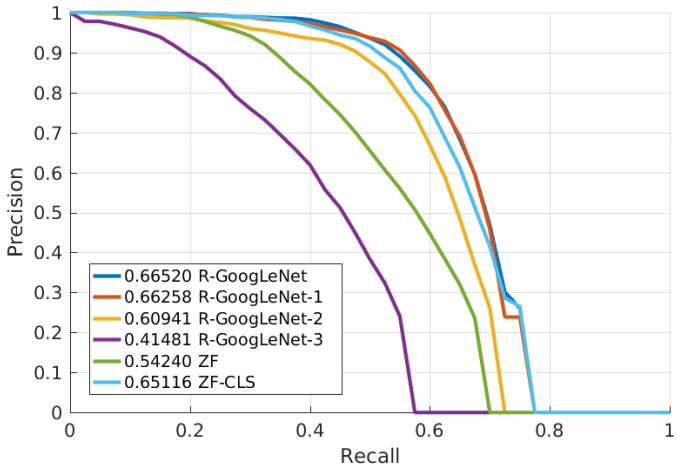


(c) hard

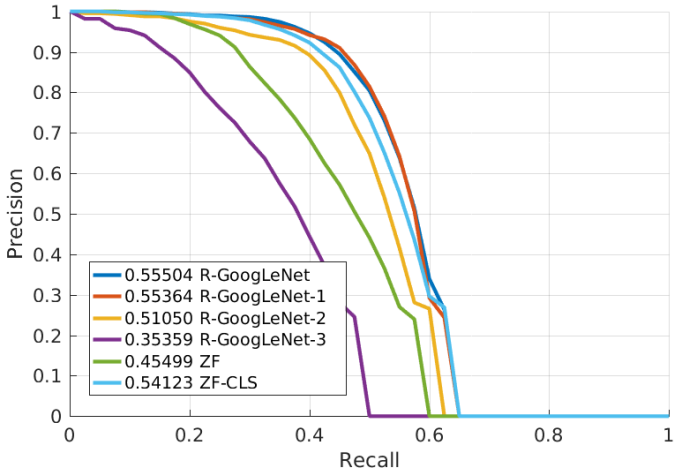
Figure 6-10: precision-recall curves of cyclist detection performance for ZF-Net and R-GoogLeNet with/without orientation regression.



(a) easy

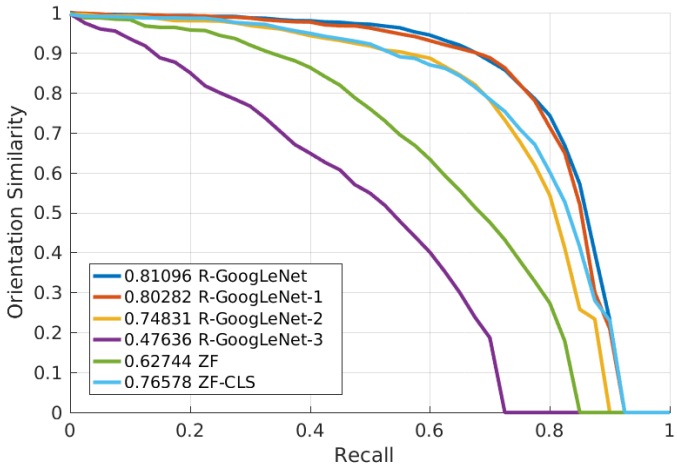


(b) moderate

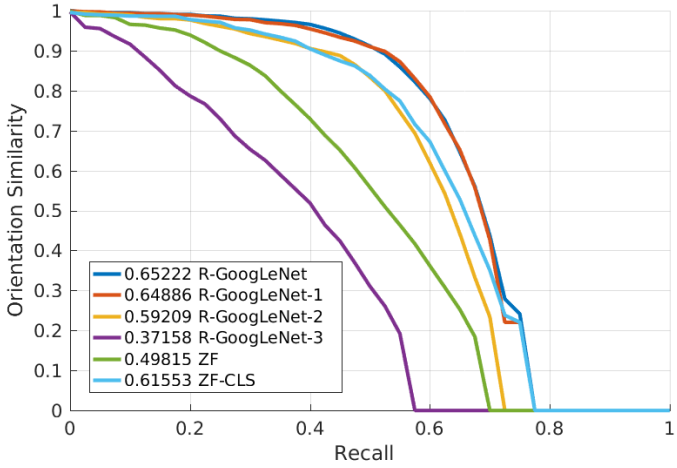


(c) hard

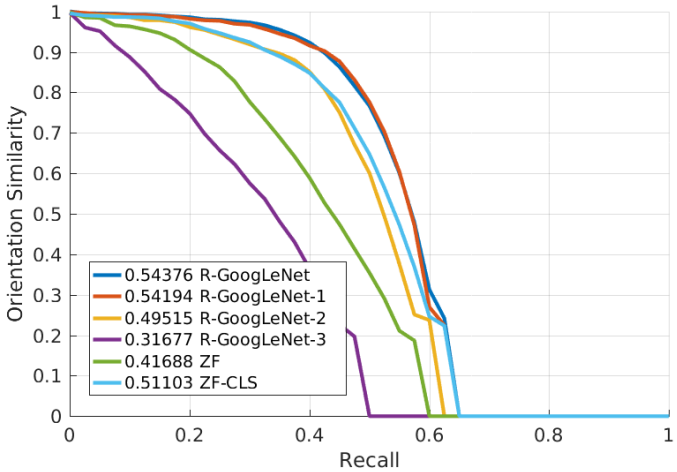
Figure 6-11: Precision-Recall curves of various of models shown for different difficulty settings.



(a) easy

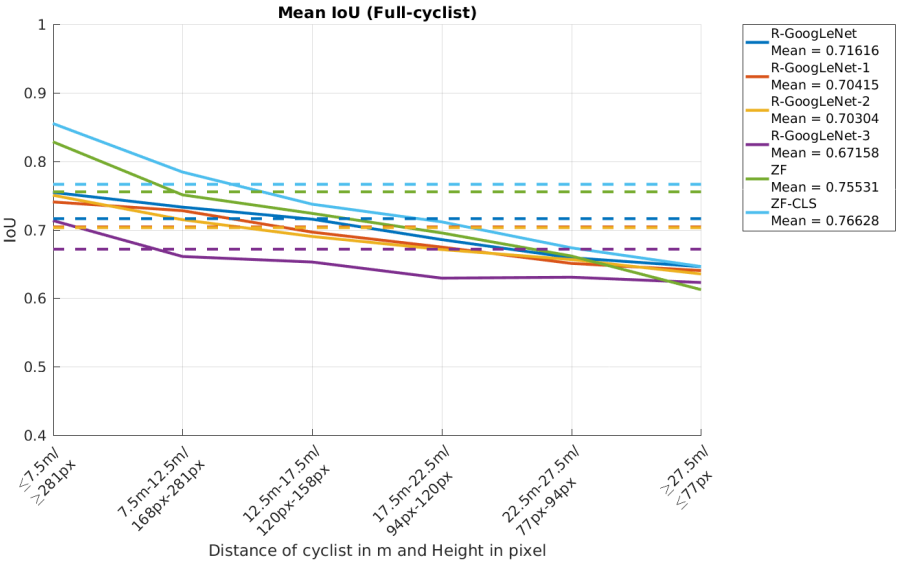


(b) moderate

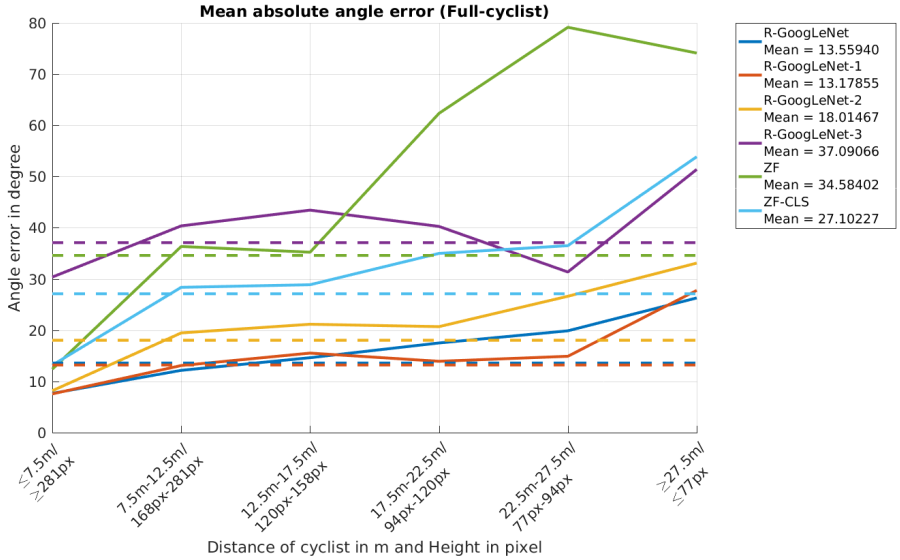


(c) hard

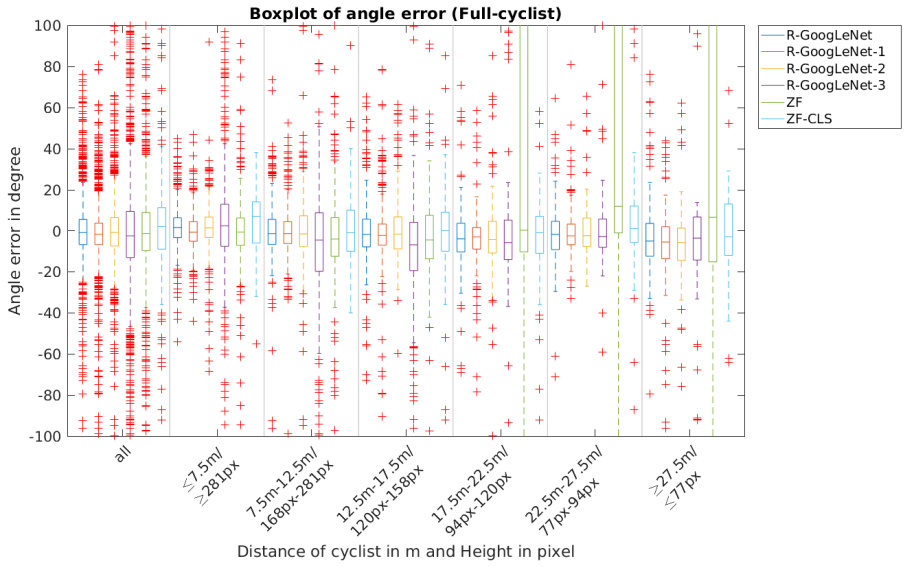
Figure 6-12: AOS-Recall curves of various of models shown for different difficulty settings.



(a) Mean IoU.



(b) Mean absolute angle error.



(c) Boxplot of angle errors.

Figure 6-13: Mean IoU (a), mean absolute angle error (b) and boxplot of angle errors (c) of true positive detections.

is discretized into 36 orientation classes, each of the classes covering 10°. From the confusion matrix result, we see the disadvantage of using classification based method (ZF-CLS): the estimation is biased to the 8 orientation classes. ZF-Net, R-GoogLeNet, R-GoogLeNet$_1$ and R-GoogLeNet$_2$ are making good continuous estimations, while ZF-Net performances slightly worth than the other three models. R-GoogLeNet$_3$ is not making reasonable estimations. The possible reason for the bad performance of R-GoogLeNet$_3$ is that it uses the deepest convolutional layer (with too many down-sampling processes) as featur, which is too small for extracting good features for making correct estimations. And also given the relative large depth of the network, the ability to propagate gradients back through all the layers of the network is a concern[3] during training.

Some sample results are shown in Figure 6-20 and Figure 6-21.

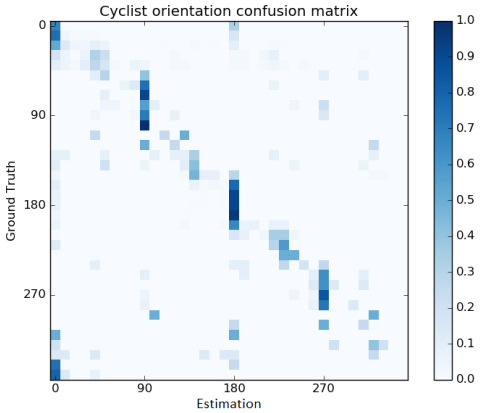
6.3.3 Parts Detection and Orientation Regression

In the third part of our experiment, we want to see how our R-GoogLeNets performs for parts (head and bike) detection and orientation regression. As stated in Section 5.6, we consider two models which are regression parts from full object proposal box and from split proposal box. We split each proposal box into three parts: full box $(0, 0, width, height)$, upper part $(-0.3width, -0.2height, 1.3width, 0.2height)$ and lower part $(0, 0.2height, width, height)$. We also choose to include some area outside the original proposed box because we notice that sometimes the proposal boxes for an object don't contain the parts (like head) properly (For example, the head position might be outside the proposal box). The splitting settings are shown in Figure 5-9b. For training and testing, we take the same parameter settings. And for the multi-task loss defined in Equation 5.14, we set the loss weights λ_{head_box} and μ_{head_orient} to be 5 and the others λ_{obj}, λ_{bike_box} and μ_{bike_orient} to be 1:

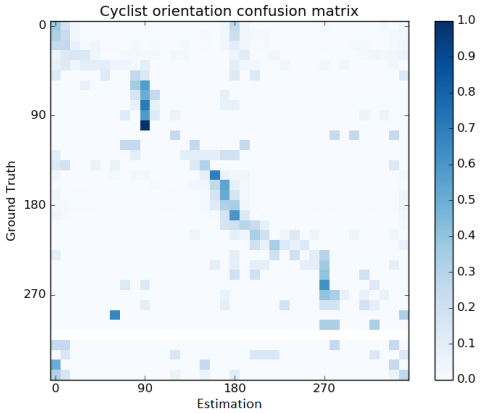
$$\begin{aligned} L = & L_{obj\_cls} + L_{obj\_box} \\ & + 5L_{head\_box} + 5L_{head\_orient} \\ & + L_{bike\_box} + L_{bike\_orient}. \end{aligned} \tag{6.9}$$

This head-biased parameter setting ensure that the network could make reasonable estimation of head part given the relative small head region. And the other training and testing parameters remains the same as in previous subsections.

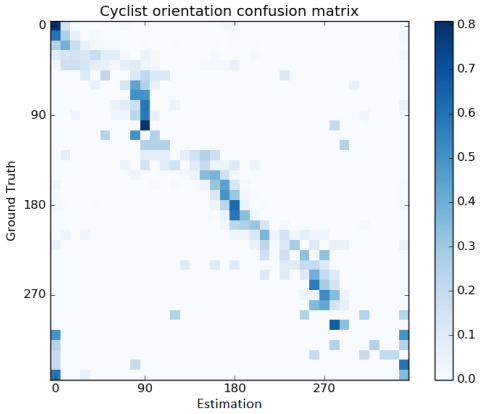
We take the true-positive detection, whose IoU is at least 0.5, for evaluation IoU and angle



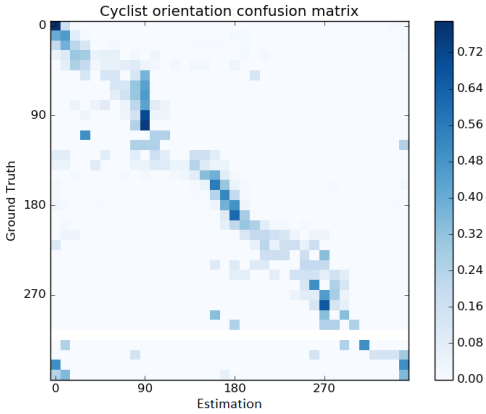
(a) ZF-CLS



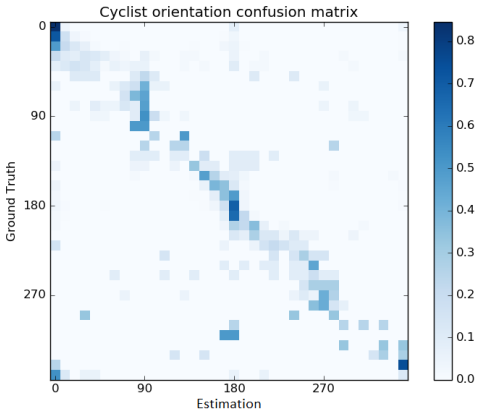
(b) ZF



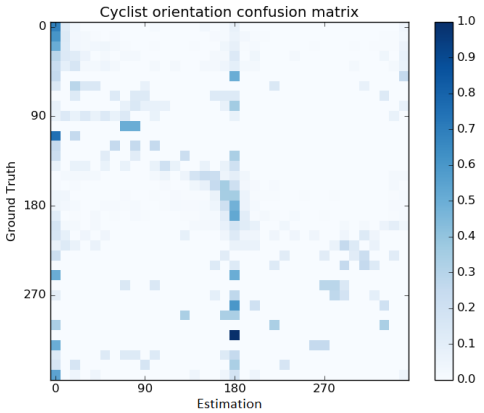
(c) R-GoogLeNet



(d) R-GoogLeNet-1



(e) R-GoogLeNet-2



(f) R-GoogLeNet-3

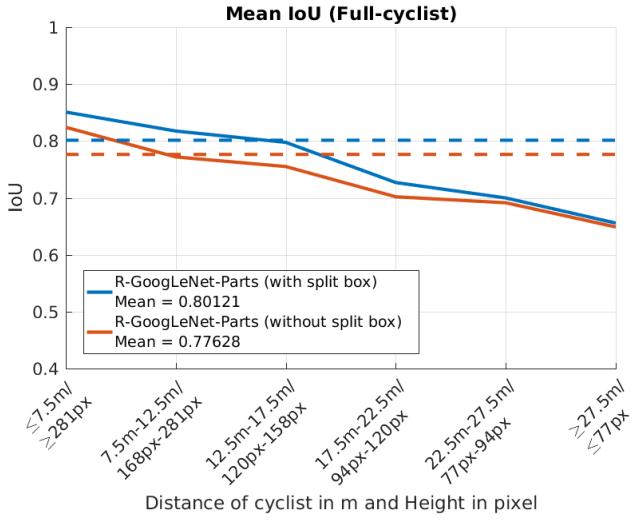
Figure 6-14: Cyclist orientation estimation of different models.

error. From Figure 6-15 and Figure 6-17, we see that in general, the two models shows quite similar performance. The mean IoU of full dataset achieves nearly 0.8 for both methods while the model with split-proposal box is slightly better. The reason for the performance difference is that, in full-proposal approach, all five estimation tasks are assigned to one single feature map. In split-proposal approach, the tasks for parts are assigned to their own feature maps, and the feature map of the full-object proposal is only assigned with the classification task and full-object bounding-box regression. Therefore, the full-object bounding-box regression in split-proposal approach is less affected by other tasks than that in full-proposal approach.

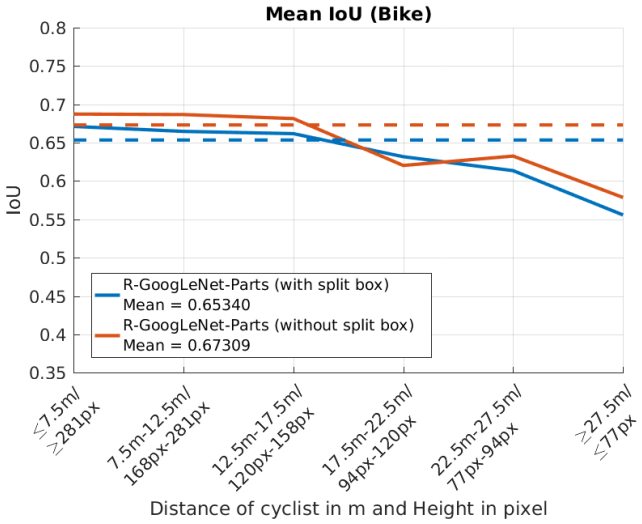
For parts bounding-box regression, the mean IoU is lower than full object bounding-box regression. And comparing the performance of parts, the regression of bike bounding-box is better than head. The reason behind is that we adopted relative representation of the bounding-boxes during training. All bounding-boxes coordinates are transformed to the offset representation (see Section 3.3.2) with respect to the full-proposal boxes. When investigating into the dataset, we found that the distribution of head bounding-boxes has larger variance variance than the distribution of bike-bounding boxes (see Figure 6-16). This makes it harder to estimate the head location than the bike location. And we also noticed that for parts bounding-box regression tasks, the regression of full-proposal approach is slightly better than of split-proposal. This is because the parts locations actually obey to geometric structure law of cyclists, and thus they are more dependent on global features rather than local features. So for parts bounding-box regression, the full-proposal approach over-performance the split-proposal approach.

And for parts orientation, the absolute angle error of bike is lower than head with the mean value of around 25 degree for both methods. The results in Figure 6-17 also show that the mean angle error from full-proposal box is better than from split-proposal box. The cause of the difference is that the angle estimation of parts still benefits from features from full-object, especially when the parts orientation biased to be the same as the full-objects.

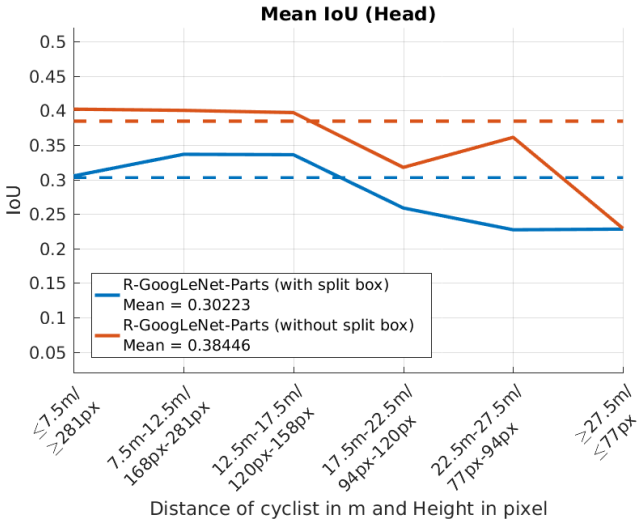
Figure 6-18 shows boxplots for the estimated head and bike orientation across different full-object sizes. The line inside each box marks the median and red crosses indicate the outliers. The box contains 50% of samples and the whiskers define 99.3% coverage of samples in each group. In general, the variance of angle error gets larger with the decrease of cyclist scales. We notice that at the scale group 22.5m-27.5m/77px-94px, the blue boxes (split-proposal) and their whiskers are significantly larger than the group 17.5m-22.5m/94px-120px, while the red boxes (full-proposal) remains a stable size. This indicates that when the object's scale is smaller than 94px, the orientation regression results from split-proposal approach become



(a) Mean IoU of full cyclist.

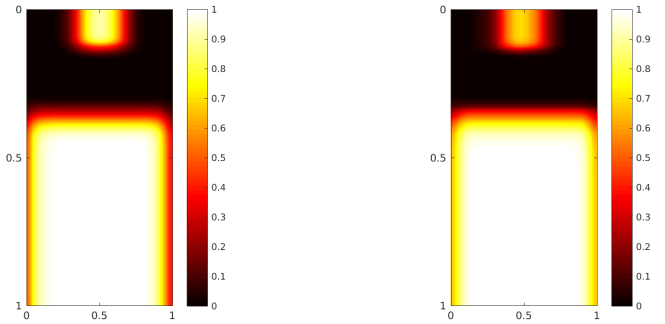


(b) Mean IoU of bike part.



(c) Mean IoU of head part.

Figure 6-15: Mean IoU curve of full-cyclist (a), bike (b) and head (c) for parts regression from full-proposal (red lines) and split-proposal (blue linex), grouped by object distance. The dashed lines shows the mean IoU of the whole validation set.



(a) Training set (b) Validation set

Figure 6-16: Head and bike bounding-box distribution inside full-cyclist bounding-boxes of training set (a) and validation set (b). The highlight in upper part of boxes is the distribution of head bounding-boxes, and the highlight in lower part of boxes is the distribution of bike bounding-boxes.

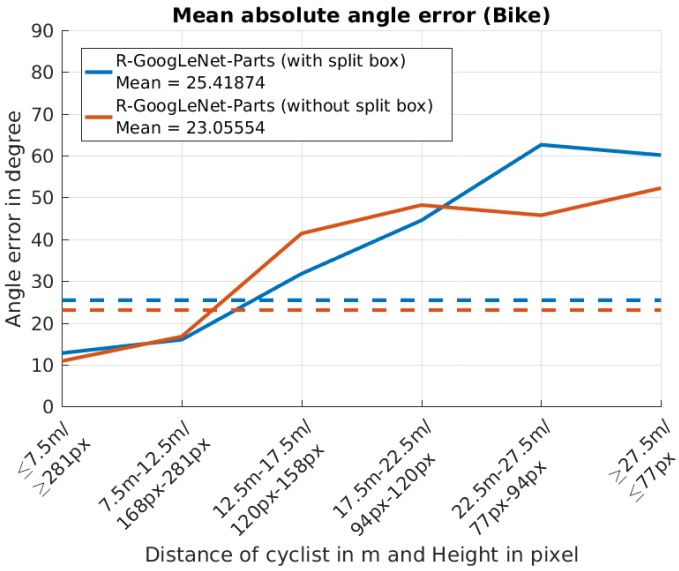
unstable, because the sub-regions are too small for orientation estimation. At this scale, parts orientation estimation of the full-proposal approach, however, could benefit from global features and make relative more stable estimation. And at scale ≥27m/0px-77px both methods are making unstable estimations. It means that in this scale, the feature maps of either full-object or parts are too small for parts orientation estimation.

Figure 6-19 shows the confusion matrices of parts orientation regression with two approaches. For both methods, the bike orientation estimation shows better result than head orientation estimation. The results also shows that the estimated head orientation are biased to 180°and 0°. This might be caused by three reasons. Firstly, the head orientation are mostly the same as the bike orientation. Secondly, the bike orientations in dataset are biased to 180°and 0°(see Figure 6-4). Thirdly, the feature maps for head region are not proper trained for orientation regression.

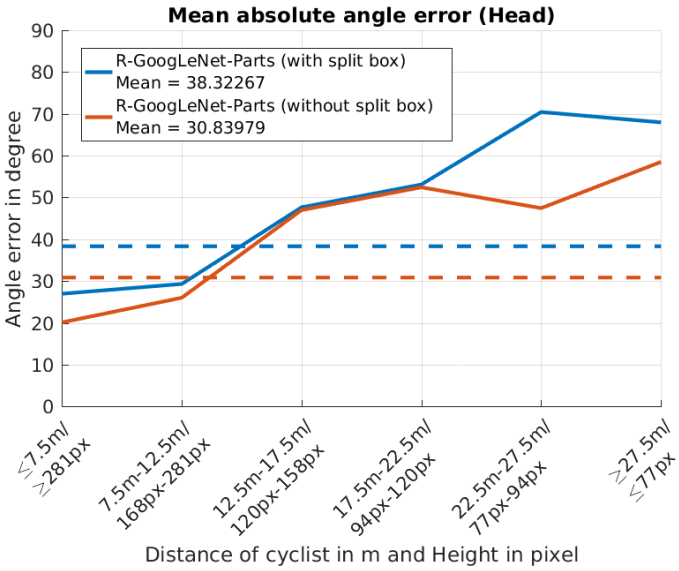
Some sample results are shown in Figure 6-22.

6.3.4 Runtime Speed

At test time we run all our models on an Nvidia Titan X GPU and we evaluate the runtime of CNN calculation. Table 6.4 shows the average estimation speed of each model running on all images. The results shows that the running speed is model dependent. The small ZF-Net based models , though have lower precision, achieve over 10fps. The original R-GoogLeNet based models (R-GoogLeNet, R-GoogLeNet$_3$ and R-GoogLeNet-Parts with full-proposal) are about 5.5fps to 6fps. R-GoogLeNet$_1$ and R-GoogLeNet$_2$ are faster than original R-GoogLeNet based models. Especially, R-GoogLeNet$_1$'s speed is close to ZF-Net, while it remains competitive on estimation accuracy compared to original R-GoogLeNet. R-GoogLeNet-Parts with split-

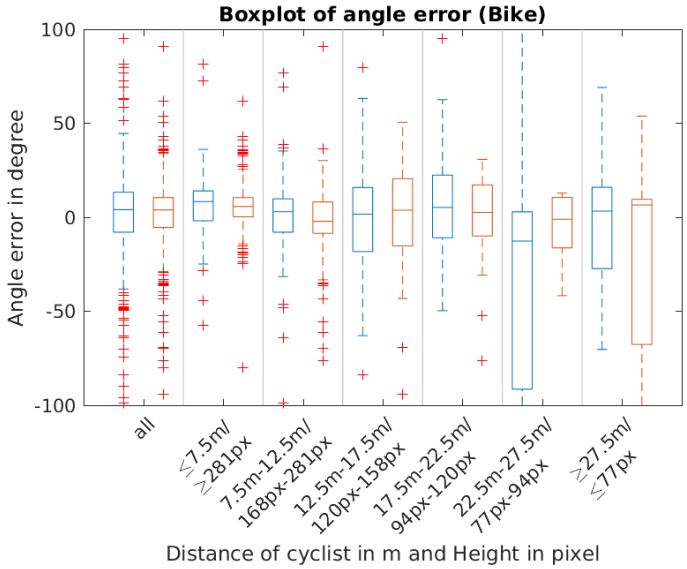


(a) Mean absolute angle error of bike part.

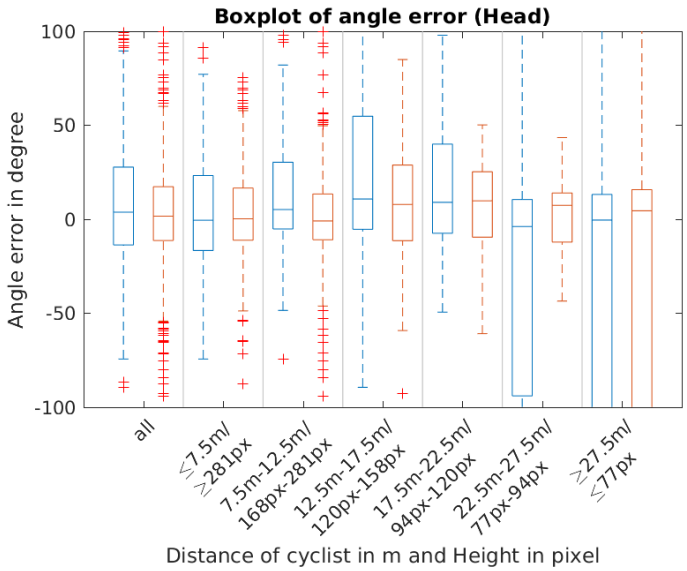


(b) Mean absolute angle error of head part.

Figure 6-17: Mean absolute angle error curve of bike (a) and head (b) part regression from full-proposal (red boxes) and split-proposal (blue boxes), grouped by object distance. The dashed lines shows the mean absolute angle error of the whole validation set.

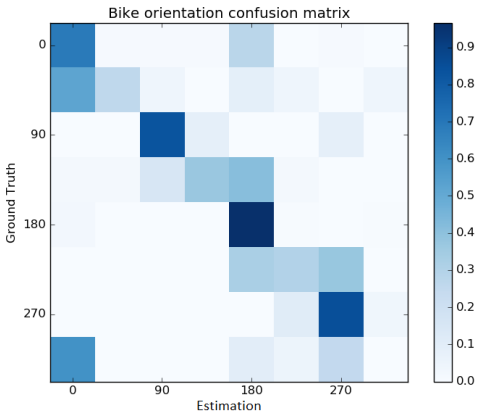


(a) Angle error distribution of bike part.

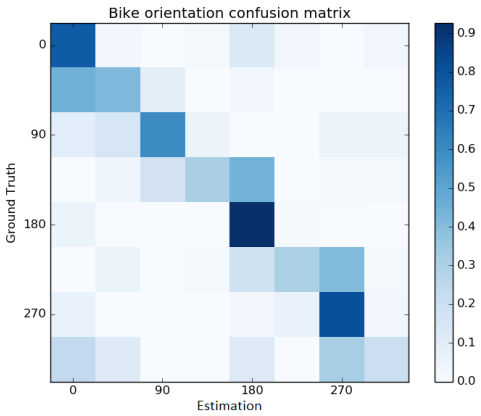


(b) Angle error distribution of head part.

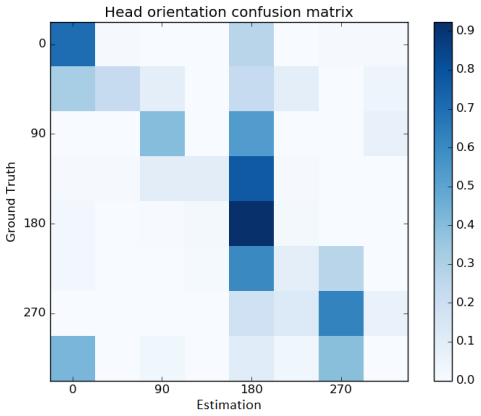
Figure 6-18: The distribution of angle error of bike (a) and head (b) part regression from full-proposal (red boxes) and split-proposal (blue boxes), grouped by object distance.



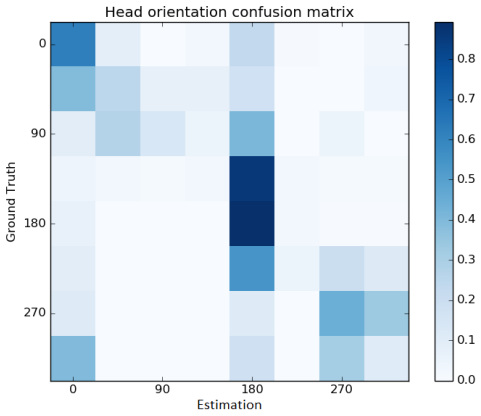
(a) full-proposal, bike



(b) split-proposal, bike



(c) full-proposal, head



(d) split-proposal, head

Figure 6-19: Confusing matrices of bike (a, b) and head (c, d) orientation estimation with full-proposal (a, c) and split-proposal (b, d) methods. 360°is discretized into 8 orientation classes, each covers 45°.

proposal box is significantly slower than original R-GoogLeNet, because it has three feature maps for each object proposal and thus has three times number of parameters in fully-connected layers to be trained.

Task	Model	Time (per frame)	FPS
Detection	ZF	0.095s	10.5
	R-GoogLeNet	0.178s	5.6
Detection and orientation	ZF	0.094s	10.6
	ZF-CLS	0.093s	10.7
	R-GoogLeNet	0.175s	5.7
	R-GoogLeNet-1	0.112s	8.9
	R-GoogLeNet-2	0.155s	6.5
	R-GoogLeNet-3	0.180s	5.6
Parts	R-GoogLeNet-Parts (Full)	0.176s	5.7
	R-GoogLeNet-Parts (Split)	0.397s	2.5

Table 6.4: Processing Timing.

Figure 6-20: Cyclist detection and orientation estimation samples. Green bounding-boxes are the proposal boxes used for the corresponding detections. Blue bounding-boxes are the bounding-box regression results. The arrows represent the estimated orientations.

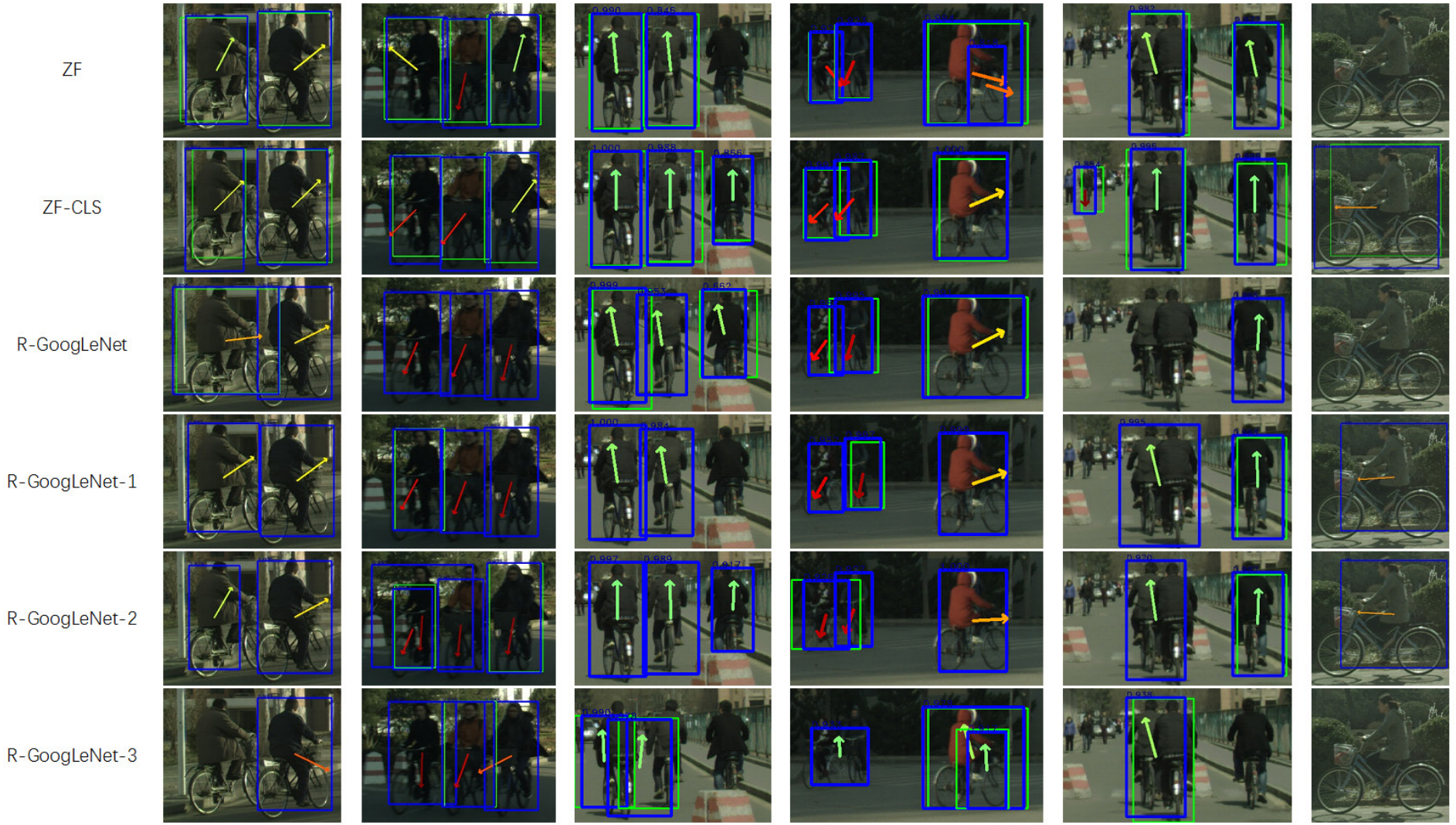


Figure 6-21: Cyclist detection and orientation estimation samples (continued). The last two columns contain the false negative detection samples of R-GoogLeNet. The the failure in second last column might be caused by the heavy overlapping of two target objects. The failure in the last column might be caused by low contrast and similar texture between the cyclist and the background.

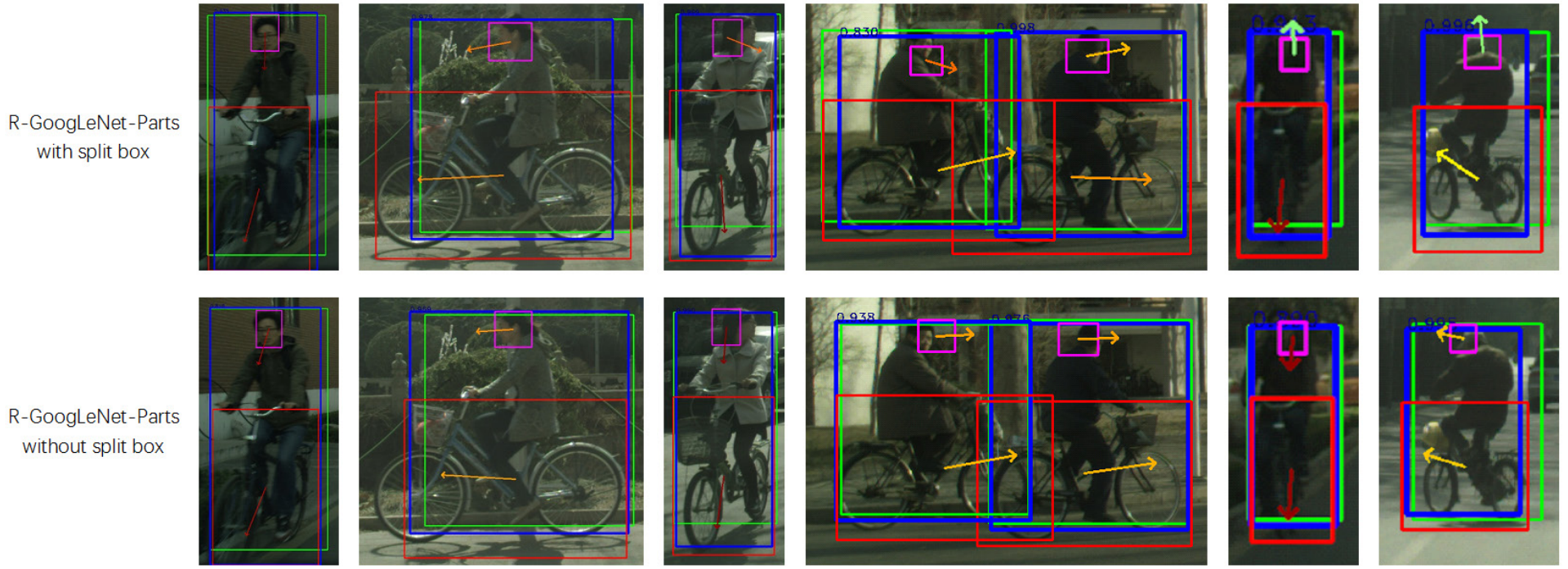


Figure 6-22: Cyclist parts detection and orientation estimation samples. Green and blue bounding-boxes are defined the same as in Figure 6-20. Purple bounding-boxes are the estimated head locations. Red bounding-boxes are the estimated bike locations. The arrows represent the estimated orientations of parts. The first two columns are correct estimations. The third and fourth columns are the cases that split-proposal box version works better when head and body are in different directions. The last two columns are the cases that head orientation estimation of full-proposal box benefits from the orientation constraints, while the results from split-proposal box version ignore the constraints and made wrong estimations.

Chapter 7

Conclusion

This chapter will first summarize the work presented in this thesis. thereafter, the limitations of our proposed approaches will be described. And the last part will give the possible directions for future work.

7.1 Summary

In this thesis we apply the ZF-Net based Fast R-CNN to our cyclist dataset to lay the foundations of our work. The usage of Fast R-CNN shows promising results in cases of various difficulty (see Section 6.3.1). The relative small ZF-Net also shows good running efficiency which means it would be applicable to a real-time system running in the car. Then comes the major contributions of our work.

First, we propose Pose-RCNN, which is one of our main contribution in our work. The framework combines R-CNN based object detection models with orientation estimation into one single framework. The framework is tested on ZF-Net and different variants of GoogLeNet. The best performance is achieved by our proposed R-GoogLeNet with 0.824/0.665/0.555 mean average precision and 0.811/0.652/0.544 mean orientation similarity for easy/moderate/hard cases (see Section 6.3.2). The R-GoogLeNet process images at about 5.6 fps (see Section 6.3.4).

Secondly, another main contribution is that we proposed R-GoogLeNet which is a Fast R-CNN variant of GoogLeNet. It well combines three intermediate features maps of the original GoogLeNet for extracting features of RoIs and achieves better performance than using any of the single feature map of GoogLeNet. For both detection-only and joint detection-orientation estimation tasks, our R-GoogLeNet shows the best performance comparing to ZF-Net and R-GoogLeNets with different single feature maps. We also noticed that R-GoogLeNet$_1$ (a sub-structure of R-GoogLeNet with less layers, see Section 5.5 for detail) achieves 0.817/0.663/0.554

mean average precision and 0.803/0.649/0.542 mean orientation similarity (see Section 6.3.2), which are very competitive compared to R-GoogLeNet, while it has smaller network structure and 1.5× faster speed than R-GoogLeNet with about 9 fps (see Section 6.3.4).

Thirdly, we proposed two joint parts detection and orientation estimation approaches and applied them to R-GoogLeNet. The first approach is parts regression from full-proposal box, which is an extension of Pose-RCNN. It directly assigns parts bounding-box regression and orientation regression tasks to the full-proposal box. The other approach is parts regression from split-proposal box. In this approach, sub-regions are generated based on geometric-layout constraints of parts (head and bike). Feature maps are generate per sub-region and the regression tasks related to different parts are assigned to corresponding feature maps (see Section 6.3.3 for results). These two models are compared and each of them has advantages and limitations. We are going to explain them in the following section.

7.2 Limitations

Parameter Optimization

Due the limit of time, the parameters of our experiments are all based on the original model settings and are modified slightly in a heuristic fashion to fit our dataset. They are chosen to ensure that we can get workable models. There's still room for optimization.

Object Parts Regression

When comparing the performances of parts regression methods, we noticed that the bounding-box regression and orientation regression of parts from full-proposal box works better than from split-proposal box. The reason is that when using full-proposal box, the parts share a single feature vector and thus the geometric relationship of parts are already implicitly encoded by the network. Especially for orientation regression, the using of split-proposal box is more likely to ignore the orientation relations among parts (for example, the head orientation is more likely to be the same orientation as body, see Figure 6-4), and the missing of this information might lower the orientation prediction accuracy. See last two column in Figure 6-22 for the examples in this case.

We also noticed from the orientation confusion matrices (see Figure 6-19) that the prediction of head orientation is biased to 0°and 180°. When investigating into the training set, we found the cyclist orientations are biased which might cause this problem. On one hand, the cyclist

orientations are biased either towards (0°) or away from (180°) the camera. On the other hand, in most cases the head orientation is the same as the body orientation. Meanwhile, the head area is too small to provide good features for estimation. The above reasons together caused the biased head orientation estimation. However the biased prediction of orientation did not happen to the body. The reason might be that the area of body is larger than head, and thus the appearance of body features could provide enough information for orientation estimation. Then the predictors are trained to predict based on real features instead of geometric constraints among parts.

7.3 Future Work

Known the limitations of current approaches, we are going to list some possible direction for future work.

Dataset

The first limitation of our approaches is that even though our models are all multi-class based implementations, limited by the labeled categories in our dataset, we only conduct the experiments for cyclist class. So it would worthy to find other dataset with more object category labels, evaluate the models' performances and compare with other existing methods.

As stated before, there are two issues of our dataset. Firstly, it only contains orientation labeling of cyclists. Secondly, the orientations of head and bike are biased distributed. To resolve these issues, the first option will be training and evaluating on other dataset like KITTI [89] which contains orientation labeling for pedestrians, cyclists and cars. The KITTI benchmark also provides results of other existing methods which worth to be compared with. To address the biased orientation issue, bootstrapping could be introduced[4]. During training, the hard samples (with high training loss) will have higher possibility to be reused in next iterations. In our case, the hard samples could contain objects towards left or right and objects whose head and body are in different orientations. Bootstrapping is in-general a way to improve accuracy. And our models are likely to be over-fitting on small amount of samples with biased orientation. So another option could be introducing larger dataset with various orientation distributions and preprocessing the dataset and ensure that the orientation are balanced distributed, so that there will always be roughly the same amount of samples in different orientation for training.

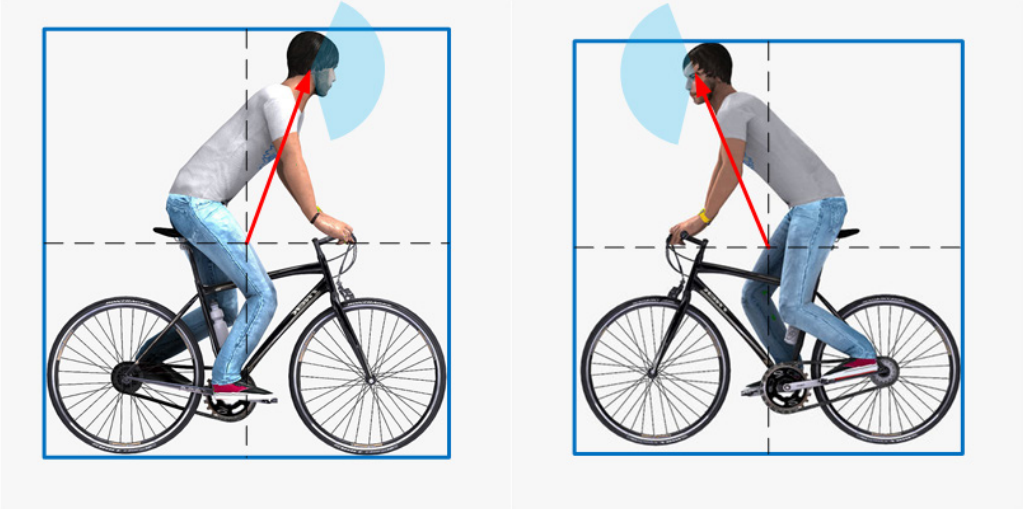


Figure 7-1: Head location is related to the orientation of the cyclist. When cyclist is towards right, the head tend to be distributed at middle-right and vice versa. And the head orientation can be restricted to an sector region centered at the cyclist's orientation.

Loss Function

Currently we are using the weighted sum of multiple task losses. However it doesn't reveal the structural prior of parts in an object. For example, the head location is not always located at the top-middle of a cyclist but related to the cyclist's orientation (see Figure 7-1). And the head orientation can also be modeled as a normal distribution centered at the bike orientation. Thus the next research direction would be modeling the loss function so that the loss could evaluate the parts based on there layout relation, relative orientation and etc. This means introducing structural priors into the CNN. The introducing of parts relationships in loss function would lead to a real joint-parts approach and it would possibly to improve the performance of parts regression, in the hope that CNNs can make use of well defined prior to learn more appropriate features for prediction.

Training Parameters

In our approaches, the hyper-parameters (including learning rate, learning rate dropping factor, step size, number of iterations, loss weights, size of RoI pooling layer, etc.) are chosen to ensure that we have a workable framework but not guaranteed to be optimal ones. So an important step of future work would be tuning the parameters and finding the best combinations for each of the models. Methods like grid search, randomized search could be adopted to optimize the parameter settings.

Bibliography

[1] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.

[2] L. Beyer, A. Hermans, and B. Leibe, "Biternion Nets: Continuous Head Pose Regression from Discrete Training Labels," in *German Conference on Pattern Recognition (GCPR)*, Springer, 2015, pp. 157–168.

[3] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[4] X. Li *et al.*, "A new benchmark for vision-based cyclist detection," *IEEE Intelligent Vehicles Symposium (IV)*, forthcoming.

[5] World Health Organization. Violence and Injury Prevention and World Health Organization, *Global status report on road safety 2015*. World Health Organization, 2015.

[6] Organisation Internationale des Constructeurs d'Automobiles. (2016). Worldwide automobile production from 2000 to 2015, [Online]. Available: `http://www.oica.net/category/production-statistics/` (visited on 07/01/2016).

[7] ——, (2015). World vehicles in use from 2006 to 2014, [Online]. Available: `http://www.oica.net/category/vehicles-in-use/` (visited on 07/01/2016).

[8] Law Offices of Michael Pines. (2013). Top 25 causes of car accidents, [Online]. Available: `https://seriousaccidents.com/legal-advice/top-causes-of-car-accidents/` (visited on 08/08/2016).

[9] M. Enzweiler and D. M. Gavrila, "Monocular pedestrian detection: Survey and experiments," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 31, no. 12, pp. 2179–2195, 2009.

[10] The Tesla Team. (2016). A tragic loss, [Online]. Available: `https://www.teslamotors.com/blog/tragic-loss` (visited on 07/02/2016).

[11] D. M. Gavrila, "Active pedestrian safety: From research to reality," Damiler AG Group Research and Development, Ulm, Germany, Tech. Rep., 2013. [Online]. Available: `https://www.robots.ox.ac.uk/seminars/Extra/2013_04_10_DariuGavrila.pdf`.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1097–1105.

[13] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *International Journal of Computer Vision (ECCV)*, Springer, 2014, pp. 818–833.

[14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *International Conference on Learning Representations (ICLR)*, 2015.

[15] *Pose-RCNN: Joint object detection and pose estimation using 3d object proposals*, 2016.

[16] B. Heisele, "Object recognition in images," Honda R&D Americas, Tech. Rep., 2002. [Online]. Available: `http://www.mit.edu/~9.520/spring04/Classes/objectrecognition.ppt`.

[17] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *International Journal of Computer Vision (ECCV)*, Springer, 2014, pp. 391–405.

[18] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in the Neural Information Processing Systems (NIPS)*, 2015, pp. 91–99.

[19] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 24, no. 5, pp. 603–619, 2002.

[20] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision (IJCV)*, vol. 59, no. 2, pp. 167–181, 2004.

[21] P. Dollár and C. L. Zitnick, "Structured forests for fast edge detection," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2013, pp. 1841–1848.

[22] M. Enzweiler, M. Hummel, D. Pfeiffer, and U. Franke, "Efficient stixel-based object recognition," in *IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2012, pp. 1066–1071.

[23] D. M. Gavrila and S. Munder, "Multi-cue pedestrian detection and tracking from a moving vehicle," *International Journal of Computer Vision (IJCV)*, vol. 73, no. 1, pp. 41–59, 2007.

[24] D. Gerónimo, A. D. Sappa, D. Ponsa, and A. M. López, "2D–3D-based on-board pedestrian detection system," *Computer Vision and Image Understanding*, vol. 114, no. 5, pp. 583–595, 2010.

[25] C. G. Keller, M. Enzweiler, M. Rohrbach, D. F. Llorca, C. Schnörr, and D. M. Gavrila, "The benefits of dense stereo for pedestrian detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1096–1106, 2011.

[26] F. Flohr, M. Dumitru-Guzu, J. F. Kooij, and D. M. Gavrila, "A probabilistic framework for joint pedestrian head and body orientation estimation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 1872–1882, 2015.

[27] H. Badino, U. Franke, and D. Pfeiffer, "The stixel world-a compact medium level representation of the 3D-world," in *Joint Pattern Recognition Symposium*, Springer, 2009, pp. 51–60.

[28] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, vol. 1, 2005, pp. 886–893.

[29] N. Dalal, "Finding people in images and videos," PhD thesis, Institut National Polytechnique de Grenoble-INPG, 2006.

[30] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, vol. 1, 2001, pp. I–511.

[31] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, IEEE, vol. 1, 2002, pp. I–900.

[32] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 32, no. 9, pp. 1627–1645, 2010.

[33] P. Dollár, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 36, no. 8, pp. 1532–1545, 2014.

[34] B. Wu and R. Nevatia, "Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors," in *IEEE International Conference on Computer Vision (ICCV)*, IEEE, vol. 1, 2005, pp. 90–97.

[35] K. Mikolajczyk, C. Schmid, and A. Zisserman, "Human detection based on a probabilistic assembly of robust part detectors," in *International Journal of Computer Vision (ECCV)*, Springer, 2004, pp. 69–82.

[36] H. Cho, P. E. Rybski, A. Bar-Hillel, and W. Zhang, "Real-time pedestrian detection with deformable part models," in *IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2012, pp. 1035–1042.

[37] O. Barnich and M. Van Droogenbroeck, "ViBe: A universal background subtraction algorithm for video sequences," *IEEE Transactions on Image Processing*, vol. 20, no. 6, pp. 1709–1724, 2011.

[38] S. Piérard, A. Lejeune, and M. Van Droogenbroeck, "A probabilistic pixel-based approach to detect humans in video streams," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2011, pp. 921–924.

[39] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial intelligence*, vol. 17, no. 1-3, pp. 185–203, 1981.

[40] P. Viola, M. J. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," *International Journal of Computer Vision (IJCV)*, vol. 63, no. 2, pp. 153–161, 2005.

[41] S. Walk, N. Majer, K. Schindler, and B. Schiele, "New features and insights for pedestrian detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2010, pp. 1030–1037.

[42] D. Park, C. L. Zitnick, D. Ramanan, and P. Dollár, "Exploring weak stabilization for motion feature extraction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 2882–2889.

[43] D. H. Hubel and T. N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *The Journal of Physiology*, vol. 195, no. 1, pp. 215–243, 1968.

[44] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[45] O. Russakovsky *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[46] A. Karpathy. (Jun. 2, 2016). CS231n convolutional neural networks for visual recognition, [Online]. Available: `http://cs231n.github.io/neural-networks-1/`.

[47] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2010, pp. 807–814.

[48] A. Karpathy. (Jun. 7, 2016). CS231n convolutional neural networks for visual recognition, [Online]. Available: `http://cs231n.github.io/convolutional-networks/`.

[49] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the ACM International Conference on Multimedia*, ACM, 2014, pp. 675–678.

[50] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A matlab-like environment for machine learning," in *Neural Information Processing Systems (NIPS) Workshops*, 2011.

[51] R. Al-Rfou *et al.*, "Theano: A Python framework for fast computation of mathematical expressions," *ArXiv e-prints*, vol. abs/1605.02688, May 2016. [Online]. Available: `http://arxiv.org/abs/1605.02688`.

[52] M. Abadi *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *CoRR*, vol. abs/1603.04467, 2016. [Online]. Available: `http://arxiv.org/abs/1603.04467`.

[53] J. Redmon, *Darknet: Open source neural networks in c*, 2013–2016. [Online]. Available: `http://pjreddie.com/darknet/`.

[54] 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 (CVPR)*, 2014, pp. 580–587.

[55] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *International Journal of Computer Vision (ECCV)*, Springer, 2014, pp. 346–361.

[56] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, vol. 2, 2006, pp. 2169–2178.

[57] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You Only Look Once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015. [Online]. Available: `http://arxiv.org/abs/1506.02640`.

[58] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*, http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[59] G. Gkioxari, B. Hariharan, R. B. Girshick, and J. Malik, "R-CNNs for pose estimation and action detection," *CoRR*, vol. abs/1406.5212, 2014. [Online]. Available: `http://arxiv.org/abs/1406.5212`.

[60] G. Gkioxari, R. Girshick, and J. Malik, "Contextual action recognition with r* cnn," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1080–1088.

[61] W. Ouyang *et al.*, "DeepID-Net: Deformable deep convolutional neural networks for object detection," pp. 2403–2412, Jun. 2015, ISSN: 1063-6919. DOI: `10.1109/CVPR.2015.7298854`.

[62] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proceedings of the International Conference on Machine Learning (ICML)*, vol. 30, 2013.

[63] D. J. Beymer, "Face recognition under varying pose," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 1994, pp. 756–761.

[64] S. Niyogi and W. T. Freeman, "Example-based head tracking," in *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition (FG)*, IEEE, 1996, pp. 374–378.

[65] J. Ng and S. Gong, "Multi-view face detection and pose estimation using a composite support vector machine across the view sphere," in *In Proceedings of the IEEE International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, Citeseer, 1999.

[66] ——, "Composite support vector machines for detection of faces across views and pose estimation," *Image and Vision Computing*, vol. 20, no. 5, pp. 359–368, 2002.

[67] J. Huang, X. Shao, and H. Wechsler, "Face pose discrimination using support vector machines (svm)," in *Proceedings of the IEEE International Conference on Pattern Recognition (ICPR)*, IEEE, vol. 1, 1998, pp. 154–156.

[68] H. A. Rowley, S. Baluja, and T. Kanade, "Rotation invariant neural network-based face detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 1998, pp. 38–44.

[69] M. Jones and P. Viola, "Fast multi-view face detection," *Mitsubishi Electric Research Lab TR-20003-96*, vol. 3, p. 14, 2003.

[70] H. Shimizu and T. Poggio, "Direction estimation of pedestrian from multiple still images," in *IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2004, pp. 596–600.

[71] T. Gandhi and M. M. Trivedi, "Image based estimation of pedestrian orientation for improving path prediction," in *IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2008, pp. 506–511.

[72] M. Enzweiler and D. M. Gavrila, "Integrated pedestrian classification and orientation estimation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2010, pp. 982–989.

[73] E. Rehder, H. Kloeden, and C. Stiller, "Head detection and orientation estimation for pedestrian safety," in *IEEE Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2014, pp. 2292–2297.

[74] J. Lallemand, A. Ronge, M. Szczot, and S. Ilic, "Pedestrian orientation estimation," in *German Conference on Pattern Recognition (GCPR)*, Springer, 2014, pp. 476–487.

[75] Y. Li, S. Gong, and H. Liddell, "Support vector regression and classification based multi-view face detection and recognition," in *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition (FG)*, IEEE, 2000, pp. 300–305.

[76] Y. Li, S. Gong, J. Sherrah, and H. Liddell, "Support vector machine based multi-view face detection and recognition," *Image and Vision Computing*, vol. 22, no. 5, pp. 413–427, 2004.

[77] E. Murphy-Chutorian, A. Doshi, and M. M. Trivedi, "Head pose estimation for driver assistance systems: A robust algorithm and experimental evaluation," in *IEEE Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2007, pp. 709–714.

[78] E. Seemann, K. Nickel, and R. Stiefelhagen, "Head pose estimation using stereo vision for human-robot interaction," in *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition (FG)*, IEEE, 2004, pp. 626–631.

[79] R. Stiefelhagen, J. Yang, and A. Waibel, "Modeling focus of attention for meeting indexing based on multiple cues," *IEEE Transactions on Neural Networks*, vol. 13, no. 4, pp. 928–938, 2002.

[80] R. Stiefelhagen, "Estimating head pose with neural networks - results on the Pointing04 Workshop evaluation data," in *IEEE International Conference on Pattern Recognition (ICPR) Workshop*, 2004.

[81] M. Voit, K. Nickel, and R. Stiefelhagen, "Neural network-based head pose estimation and multi-view fusion," in *International Evaluation Workshop on Classification of Events, Activities and Relationships*, Springer, 2006, pp. 291–298.

[82] Z. Hu, "Real-time head pose estimation with convolutional neural networks," Tech. Rep., 2015. [Online]. Available: `http://cs231n.stanford.edu/reports/zhianghu_report.pdf`.

[83] N. Ghadarghadar, E. Ataer-Cansizoglu, P. Zhang, and D. Erdogmus, "A SIFT-point distribution-based method for head pose estimation," in *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, IEEE, 2012, pp. 1–4.

[84] B. Wang, W. Liang, Y. Wang, and Y. Liang, "Head pose estimation with combined 2D SIFT and 3D HOG features," in *IEEE International Conference on Image and Graphics (ICIG)*, IEEE, 2013, pp. 650–655.

[85] T. Vatahska, M. Bennewitz, and S. Behnke, "Feature-based head pose estimation from images," in *IEEE-RAS International Conference on Humanoid Robots*, IEEE, 2007, pp. 330–335.

[86] A. Saeed, A. Al-Hamadi, A. Ghoneim, *et al.*, "Head pose estimation on top of Haar-like face detection: A study using the Kinect sensor," *Sensors*, vol. 15, no. 9, pp. 20 945–20 966, 2015.

[87] K. V. Mardia and P. E. Jupp, *Directional statistics*. John Wiley & Sons, 2009, vol. 494.

[88] D. Pfeiffer and U. Franke, "Towards a global optimal multi-layer stixel representation of dense 3d data.," in *British Machine Vision Conference*, 2011, pp. 1–12.

[89] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2012, pp. 3354–3361.

[90] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. New York, NY, USA: Cambridge University Press, 2008, ISBN: 0521865719, 9780521865715.

[91] R. Girshick. (2015). py-faster-rcnn, [Online]. Available: `https://github.com/rbgirshick/py-faster-rcnn` (visited on 03/08/2016).

[92] Berkeley Vision and Learning Center. (2014). bvlc_googlenet, [Online]. Available: `https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet` (visited on 02/25/2016).