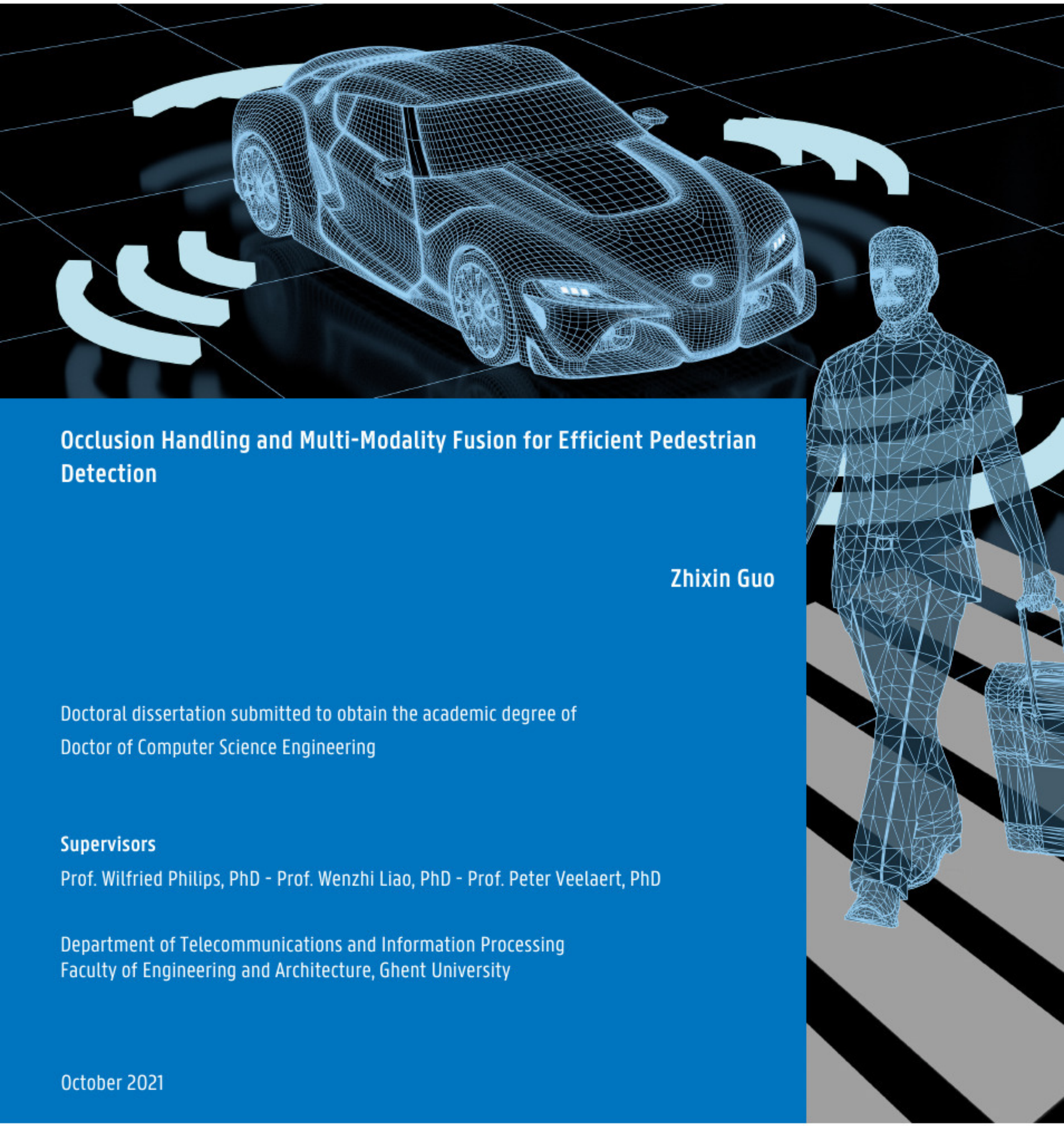
FACULTY OF ENGINEERING
AND ARCHITECTURE



Occlusion Handling and Multi-Modality Fusion for Efficient Pedestrian Detection

Zhixin Guo

Doctoral dissertation submitted to obtain the academic degree of
Doctor of Computer Science Engineering

Supervisors

Prof. Wilfried Philips, PhD - Prof. Wenzhi Liao, PhD - Prof. Peter Veelaert, PhD

Department of Telecommunications and Information Processing
Faculty of Engineering and Architecture, Ghent University

October 2021

GHENT
UNIVERSITY

FACULTY OF ENGINEERING AND ARCHITECTURE

Occlusion Handling and Multi-Modality Fusion for Efficient Pedestrian Detection

Zhixin Guo

Doctoral dissertation submitted to obtain the academic degree of
Doctor of Computer Science Engineering

Supervisors
Prof. Wilfried Philips, PhD - Prof. Wenzhi Liao, PhD - Prof. Peter Veelaert, PhD

Department of Telecommunications and Information Processing
Faculty of Engineering and Architecture, Ghent University

October 2021

GHENT UNIVERSITY

ISBN 978-94-6355-528-9
NUR 982
Wettelijk depot: D/2021/10.500/76

Members of the Examination Board

Chair

Prof. Patrick De Baets, PhD, Ghent University

Other members entitled to vote

Prof. Nikos Deligiannis, PhD, Vrije Universiteit Brussel
Prof. Bart Goossens, PhD, Ghent University
David Van Hamme, PhD, Ghent University
Prof. Peter de With, PhD, Technische Universiteit Eindhoven, the Netherlands

Supervisors

Prof. Wilfried Philips, PhD, Ghent University
Prof. Wenzhi Liao, PhD, Ghent University
Prof. Peter Veelaert, PhD, Ghent University

Acknowledgements

This work would never have been finished without the support from many people. First of all, I would like to express my sincere gratitude to my supervisors, Prof. Wilfried Philips, Prof. Wenzhi Liao and Prof. Peter Veelaert, for providing me this great opportunity to pursue my doctoral degree with the group of Image Processing and Interpretation (IPI), Ghent University. I still remember how "young" I was at the beginning of my PhD study, but they showed me lots of patience and encourage, and continuously inspired me with their expert knowledge and constructive suggestions. They set an excellent example to me about what a qualified scholar should be like, and it will benefit me for the rest of my life.

I would like to thank the Chinese Scholarship Council (CSC) for financially supporting my PhD research. And I would like to show my appreciation to all the members from the Examination Board for their careful review and helpful feedbacks.

My sincere gratitude also goes to my colleagues at TELIN in Ghent university. As I have never left China before my PhD career, I had some concerns about life in a foreign city. But they showed me so much kindness and created a pleasant working atmosphere, which swept away my worries soon. Special thanks to my colleagues from office 110.029: Martin Dimitrievski, David Van Hamme, Maarten Slembrouck and Zuhaib Ahmed, who have helped me a lot, both in my research, and in my life in Belgium. I am also deeply grateful to all my Chinese friends for sharing with me so many joyful moments in Ghent.

I owe a big thank you to my parents, Fenglun Guo and Ju Han. They have constructed a warm family for me and provided me good education environment since I was a child. It is their unconditional dedication that brings me to today.

Last, but not least, my deepest gratitude goes to my girlfriend, Yifan Xiao, for her selfless love, patience and trust. I am so excited that she said YES to my proposal, and I am looking forward to a bright future with her.

Ghent, June, 2021.
Zhixin Guo

Contents

List of Figures

List of Tables

List of Abbreviations

ACF	Aggregated Channel Feature
AP	Average Precision
BDT	Boosted Decision Trees
CIE	International Commission on Illumination
CNN	Convolutional Neural Network
CR	Compression Ratio
DPM	Deformable Part-based Model
DT	Decision Trees
FETS	FLOPs-efficient Filter Pruning via Transfer Scale
FLOPs	Floating-point Operations
FN	False Negative
FP	False Positive
HOG	Histogram of Oriented Gradient
ICF	Integral Channel Features
IOU	Intersection over Union
KNN	K-Nearest Neighbor
LBP	Local Binary Pattern
LiDAR	Light Detection and Ranging
LR	Logistic Regression
MAE	Mean Absolute Error
MAP	Maximum A Posterior
MR-FPPI	Miss Rate-False Positive Per Image
MSE	Mean Squared Error
NB	Naive Bayes
NMS	Non-Maximum Suppression
NN	Neural Network

P-R	Precision-Recall
ReLU	Rectified Linear Units
ROI	Region of Interest
RPN	Region Proposal Network
SGD	Stochastic Gradient Descent
SIFT	Scale-Invariant Feature Transform
SS	Selective Search
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
TS	Transfer Scale
WSSN	Weak Segmentation Supervised Network

Samenvatting

Voetgangersdetectie is een belangrijke maar uitdagende taak in veel computer vision-toepassingen, zoals autonoom rijden, videobewaking en analyse van menselijk gedrag. De grootste uitdaging van voetgangersdetectie is de verscheidenheid van het uiterlijk van mensen. Verschillende houdingen, kijkhoeken, kleding en accessoires kunnen het uiterlijk van een voetganger grotendeels veranderen, wat een hoge eis stelt aan de robuustheid van voetgangersdetectoren. In het afgelopen decennium, met het verschijnen van verschillende grootschalige voetgangersdatasets en de ontwikkeling van computer vision-technologieën, zijn de nauwkeurigheid en betrouwbaarheid van moderne voetgangersdetectoren aanzienlijk verbeterd. Er zijn echter nog meer gedetailleerde studies nodig om een aantal uitdagende situaties aan te pakken.

Het occlusieprobleem is een onvermijdelijk probleem voor voetgangersdetectie. In de stedelijke straten wordt een voetganger vaak gedeeltelijk of zelfs volledig afgesloten door de naburige voetgangers of voertuigen, terwijl de missrate van gewone voetgangersdetectoren veel zal toenemen wanneer occlusie optreedt. Een ander probleem bij voetgangersdetectie is het lage contrast in sommige situaties. Wanneer het bijvoorbeeld 's nachts of op bewolkte dagen is, zal de slechte lichtconditie het erg moeilijk maken voor de detector om voetgangers alleen te herkennen aan zichtbare camera's. Daarom is het nodig om gegevens uit verschillende bronnen (bijv.LiDAR of thermisch) samen te voegen om de robuustheid van voetgangersdetectoren te vergroten. Ook het efficiëntieprobleem is een grote zorg voor voetgangersdetectie, omdat de moderne detectoren gewoonlijk gebaseerd zijn op diepe neurale netwerken, die hoge reken- en opslagkosten vereisen. Het zal moeilijk zijn om dergelijke detectoren in praktijktoepassingen in te zetten zonder de structuur van netwerken te optimaliseren.

Om de hierboven genoemde uitdagende problemen aan te pakken, richt dit proefschrift zich op drie specifieke onderwerpen: occlusieafhandeling, RGB-D-detectie en netwerkversnelling. Een meer specifieke samenvatting van onze bijdragen kan in het volgende worden benadrukt:

1. De eerste bijdrage van dit proefschrift is dat we een occlusie-robuuste detectiemethode voor voetgangers voorstellen. Ten eerste hebben we vastgesteld dat de meeste conventionele voetgangersdetectoren opzettelijk het gebruik van afgesloten voetgangersmonsters voor training vermijden. Dit komt doordat conventionele machine learning classificaties de occluderende objecten niet kunnen onderscheiden van de echte voetganger, wat

zal resulteren in het leren van onjuiste eigenschappen bij het trainen met geoccludeerde trainingvoorbeelden. Daarom hebben we een nieuwe kenmerkselectiestrategie ontwikkeld op basis van statistische distributies van de typische occlusieverdeling van geoccludeerde trainingvoorbeelden. Deze methode helpt de classificator om om meer kenmerken te selecteren in regio's met een lagere kans op occlusie. Deze aanpak helpt maar volgt een eerder algemene (statistische) strategie bij het selecteren van goede kenmerken voor voetgangerdetectie. Daarom hebben we deze methode verder verbeterd door de specifieke occlusiepatronen rechtstreeks in de beelden te detecteren, als convolutionele kenmerken, en zo een beter classificatieresultaat te verkrijgen. Experimentele resultaten tonen aan dat onze methode een significant lagere miss-rate bereikt voor het detecteren van geoccludeerde voetgangers, zonder dat de detectie van niet-geoccludeerde voetgangers verslechtert.

2. De tweede bijdrage gaat over fusie van RGB- en diepte-beelden voor meer geavanceerde voetgangersdetectie. De meeste bestaande RGB-D-detectiemethoden voor voetgangers volgen een gelijkaardig proces waarbij eerst afzonderlijk kenwaarna deze kenmerken dan worden samengevoegd. We stellen een alternatief voor voor deze "naïeve" manier van datafusie in classificatie: onze methode gebruikt de dieptebeelden als gidsbeelden om de convolutionele kenmerkextractie uit RGB-afbeeldingen te verfijnen. De verfijnde kenmerken die uit de echte voetgangerszone komen, zullen hogere gewichten hebben, terwijl de kenmerken van de achtergrond of occlusiegebieden lagere gewichten krijgen. Bovendien verwijdert het onderzoeken van de projectierelatie van dieptebeelden enkele valse kandidaatregio-proposities, waardoor de efficiëntie van de stroomafwaartse classificatietaken duidelijk wordt verbeterd. Experimentele resultaten tonen aan dat onze voorgestelde RGBD-methode beter presteert dan andere state-of-the-art methoden.

3. We hebben ook een gezamenlijk trainingsraamwerk voorgesteld voor gelijktijdige detectie en segmentatie van voetgangers. Door gezamenlijke training kunnen de aangeleerde functies profiteren van de supervisie van beide taken en representatiever worden voor de voetgangersdetector. Gemeenschappelijke semantische segmentatietaken vereisen echter nauwkeurige segmentatieannotaties op pixelniveau voor training, wat buitengewoon tijdrovend is om te verkrijgen. We lossen dit probleem op door zwakke segmentatiemaskers te gebruiken die automatisch worden gegenereerd door dieptebeelden. We laten zien dat deze gezamenlijke training met semantische segmentatietaak de prestaties van de voetgangerdetector verhoogt. Bovendien laten we zien dat het samenvoegen van de outputs van het classificatienetwerk en de gegenereerde segmentatiemaskers leidt tot een verdere verbetering van de detectieprestaties. We hebben uitgebreide experimenten uitgevoerd met drie RGBD-voetgangersdatasets om de effectiviteit van onze voorgestelde

methode aan te tonen. Als bijproduct verkrijgen we ook segmentatieresultaten van voetgangers van goede kwaliteit, zonder gebruik te maken van segmentatieannotaties op pixelniveau tijdens de training.

4. Onze laatste bijdrage is het versnellen van neurale netwerken voor voetgangersdetectie. We stellen hiervoor een eenvoudig maar effectief snoeicriterium op filterniveau voor. Dit criterium beoordeelt het belang van een filter door de samenhang met de volgende laag te onderzoeken. Het principe is dat voor een getraind CNN-model een belangrijk filter sterke verbindingen moet hebben met de filters in de volgende laag, anders heeft het verwijderen ervan weinig invloed op het hele netwerk. Bovendien zien we dat filters van de rekenintensieve lagen gevoeliger zijn voor snoeien. Het betekent dat snoeien op dergelijke lagen effectiever is in het comprimeren van de Floating-point Operations (FLOP's) van het model, maar het lijdt ook aan grotere nauwkeurigheidsverlies. Om dit probleem op te lossen, stellen we een FLOP-efficiëntie groep-Lasso-benadering voor om het netwerk te leren minder filters te gebruiken in de rekenintensieve lagen. Dit leidt tot betere FLOP-compressieprestaties na het snoeien. Vergeleken met de state-of-the-art methoden op verschillende populaire CNN-structuren, bereikt onze voorgestelde methode een vergelijkbare of betere nauwkeurigheid, maar met een aanzienlijk grotere FLOP-compressieverhouding.

Dit doctoraatsonderzoek heeft tot dusver geleid tot twee gepubliceerde internationale peer-reviewed tijdschriftartikelen en twee bijdragen in internationale conferenties met peer-review. Een derde tijdschriftartikel is ingestuurd en wordt momenteel beoordeeld. Van al deze publicaties was ik de hoofdauteur.

Summary

Pedestrian detection is an important but challenging task in many computer vision applications, such as autonomous driving, video surveillance and human behavior analysis. The major challenge of pedestrian detection is the variety of the appearance of humans. Different postures, observing angles, clothes and accessories can all largely change the appearance of a pedestrian, which raises a high requirement for the robustness of pedestrian detectors. Over the past decade, with the appearance of several large-scale pedestrian datasets and the development of computer vision technologies, the accuracy and reliability of modern pedestrian detectors have significantly improved. However, more detailed studies are still needed to handle some challenging situations.

The occlusion problem is an unavoidable issue for pedestrian detection. In the urban streets, a pedestrian is often partially or even fully occluded by the neighbouring pedestrians or vehicles, while the miss-rate of common pedestrian detectors will increase a lot when occlusion happens. Another problem for pedestrian detection is the low contrast in some situations. For example, when in the night or cloudy days, the poor light condition will make it very difficult for the detector to recognize pedestrians only from visible light cameras. Therefore it is necessary to fuse data from different sources (e.g. LiDAR or thermal) to enhance the robustness of pedestrian detectors. Also the efficiency problem is a big concern for pedestrian detection, because the modern detectors are commonly based on deep neural networks, which requires large computation and storage costs. It will be difficult to deploy such detectors in real life applications without optimizing the structure of networks.

In order to address the challenging problems mentioned above, this thesis focuses on three specific topics: occlusion handling, RGB-D detection and network acceleration. A more specific summary of our contributions can be highlighted in the following:

1. The first contribution of this thesis is that we propose an occlusion-robust pedestrian detection method. First, we observed that most conventional pedestrian detectors intentionally avoid using occluded pedestrian samples for training. This is because conventional machine learning based classifiers are not able to distinguish the occluding objects from the real pedestrian, which will result in learning wrong characteristics when training with occluded pedestrian samples. Therefore, we proposed a new feature selection strategy according to the average occlusion distribution of occluded pedestrian samples. The idea is to guide the classifier to

choose more features from the regions with lower probability of being occluded. This approach helps, but follows a rather general (statistical) strategy in selecting good characteristics for detecting occluded pedestrians. Therefore, we have further improved this method by directly recognizing the specific occlusion pattern of each pedestrian, and thus obtain a better classification result. Experimental results demonstrate that our method has achieved significantly lower miss-rate for detecting occluded pedestrian samples, without deteriorating the detection of non-occluded pedestrians.

2. The second contribution is about fusing RGB and depth modalities for more advanced pedestrian detection. Most of the existing RGB-D pedestrian detection methods follow a similar process, where we first extract features from RGB and depth images independently, and then fuse these features. We propose an alternative to this " naive " way of data fusion in classification: our method uses the depth images as guidance to refine the convolutional features extracted from RGB images. The refined features from the real pedestrian region will be assigned higher weights, while the features from the background or occlusion regions will have lower weights. Furthermore, by exploring the projection relationship from depth images, some false candidate region proposals are removed, which obviously improves the efficiency of the downstream classification tasks. Experimental results show that our proposed RGBD method outperforms other state-of-the-art methods.

3. We also proposed a joint training framework for simultaneous pedestrian detection and segmentation. Through joint training, the learned features can benefit from the supervision of both tasks and become more representative for the pedestrian detector. However, common semantic segmentation tasks require accurate pixel-level segmentation annotations for training, which is extremely time consuming to obtain. To solve this problem, we propose to apply weak segmentation masks automatically generated by depth images. We show that this joint training with semantic segmentation task boosts the performance of the pedestrian detector. Moreover, we show that fusing the outputs of the classification network and the generated segmentation masks leads to a further detection performance improvement. Extensive experiments have been conducted on three RGBD pedestrian datasets to demonstrate the effectiveness of our proposed method. As a byproduct, we also obtain good quality pedestrian segmentation results, without using pixel-level segmentation annotations during training.

4. The last but not least contribution of this thesis is to speed up neural networks for pedestrian detection. First we propose a simple but effective filter-level pruning criterion. This criterion assesses the importance of a filter by exploring its connection with the next layer. The principle is that for a trained CNN model, an important filter should have

strong connections with the filters in the next layer, otherwise removing it will have little influence on the whole network. Besides, we observe that filters from the computationally-intensive layers are more sensitive to pruning. It means pruning on such layers is more effective in compressing the Floating-point Operations (FLOPs) of the model, but it also suffers from bigger accuracy lost. To solve this problem, we propose a FLOPs-efficient group Lasso approach to guide the network to use fewer filters in the computationally-intensive layers, which leads to better FLOPs compression performance after pruning. Compared with the state-of-the-art methods on several popular CNN structures, our proposed method achieves similar or better accuracy, but with significantly larger FLOPs compression ratio.

This doctoral research has so far resulted in three published international peer-reviewed journal articles and two contributions in international conferences with peer-review. I am the first author of all these publications.

1

Introduction

1.1 Problem Statement

Object detection is a fundamental problem in computer vision and has attracted considerable attention in the past several decades. In particular, the more specialized problem of pedestrian detection is of great interest to both research and industry owing to its critical role in many real-world applications such as behavior analysis, video surveillance, autonomous driving and intelligent transportation. For example, in urban autonomous driving systems, since the safety of pedestrians is always the top priority, highly reliable pedestrian detection is crucial. Also many tasks such as pedestrian tracking and semantic understanding rely heavily on the performance of pedestrian detectors.

Formally, pedestrian detection needs to identify all pedestrians in an image or video and determine the location and size of each pedestrian. It must do so without reporting other objects such as trees, mannequins in shopping centers... falsely as pedestrians. When reporting results, detected pedestrians are often marked in the images by colored boxes. The main difficulties of pedestrian detection can be summarized as follows:

1. Many factors can affect the appearance of a human. Due to the flexibility of the human body, a pedestrian can assume many different postures in different motion states, such as standing, running, walking, jumping, waving and squatting down. Observing a body from different viewpoints can also result in different appearances. Also, different clothes (or the absence of clothes) and accessories, such as umbrellas, scarfs and hats, can largely change the appearance of a human.

2. In real-world application scenarios, such as autonomous driving in an urban area, many pedestrians are heavily occluded by the neighbouring pedestrians or vehicles, which seriously affects the accuracy of pedestrian detectors.

3. In some situations, low foreground-background contrast increases the difficulty of pedestrian detection. This is because some objects in the scene

may have very similar shapes, colors or textures as humans, and poor light conditions (during night or bad weather conditions) can also make it harder for the detectors to distinguish pedestrians from the background.

4. As the novel pedestrian detectors are normally complex and require large computation costs, they can hardly reach real-time processing.

With the development of technologies, larger pedestrian datasets have been collected in recent years, which help to improve the robustness of pedestrian detectors to the variance of appearances of pedestrians. However, more specialized designs are needed to handle the occlusion, low foreground-background contrast and efficiency problems. Aiming at these problems, we have proposed several state-of-the-art methods which will be introduced in the following section.

1.2 Contributions

The main novelties and contributions presented in this thesis are as follows:

- **An occlusion-robust pedestrian detection method**

  Common pedestrian detectors suffer from a serious accuracy drop when pedestrians are occluded. Intuitively, training with occluded pedestrian samples can improve the occlusion handling ability of the detector, but it also deteriorates the detection performance of the non-occluded pedestrians. This is because occluded training samples will introduce unreliable information, which affects the learning of model parameters and thus results in performance decline. Thus, most conventional machine learning based approaches intentionally avoid using occluded pedestrian samples during the training stage. In order to better utilize the information in the occluded samples, we propose a new feature selection framework, which still use conventional handcrafted features, but trains the detector to attach less importance to features that are most likely to be in occluded regions. For example, occlusion is more common in the lower body parts. Specifically, according to the spatial distribution of the occluded areas from the training samples, the detector is guided to select more features from the less occluded body parts, which are less likely to be distorted in the case of occlusion.

  Our experiments show that this approach indeed improves detection performance. However, it models occlusion probability with a single distribution for all pedestrians. A better approach is to detect occlusion for each possible candidate pedestrian directly in the image. Such an approach allows treating pedestrians differently depending on their specific degree of occlusion. We have created a better approach based on this principle. The approach is conducted with deep neural networks, which first recognizes the occlusion pattern of each pedestrian and then refines the

features accordingly. As the refined features obtain higher classification scores, it significantly reduces the detection miss rate for the occluded pedestrians.

This work resulted in one peer-reviewed journal paper [Guo 18b] and one peer-reviewed conference paper [Guo 18a].

- **A method to fuse RGB and depth images for pedestrian detection**

  Under some challenging conditions like poor illumination or complex backgrounds, it is very difficult for detectors to correctly recognize pedestrians with only RGB images. Therefore, more and more researches in recent years are focused on multi-modality pedestrian detection. We have proposed a novel method that fuses depth images (generated from LiDAR cloud points) with RGB images to get more reliable pedestrian detection results. Instead of directly fusing the features extracted from RGB and depth modalities, we propose to use the depth information to guide the refining of RGB features. Features from real pedestrian regions will have bigger weights after refining, while features from the background area and the occluding objects will be assigned smaller weights. The refined features result in higher classification scores.

  In addition, we evaluate the reliability of all candidate region proposals by exploring the depth information. By combing the pixel-wise height of each region proposal on the image, the focal length of the camera and the depth information, we can obtain an estimated height of each region proposal in real life. Then the region proposals that have abnormal heights (unlikely to be human heights) will be removed, which reduces the computation cost for the downstream classification task.

  This work resulted in one peer-reviewed conference paper [Guo 19].

- **Weak segmentation supervised deep neural networks for pedestrian detection**

  Many researchers utilize semantic segmentation as a complementary information source in pedestrian detection. However, it requires accurate pixel-level semantic segmentation annotations for training, which is extremely time-consuming to obtain. Instead of using these expensive pixel-level annotations, we proposed to use weak segmentation masks automatically generated by depth images. This enables joint semantic segmentation and pedestrian detection with only ground truth bounding boxes for training. We show that this joint training boosts the performance of the pedestrian detector. Moreover, we have observed that the generated segmentation results can also be used to indicate the existence of pedestrians that are missed by classifiers. Thus, fusing the outputs of the classification network and the generated segmentation masks brings better detection performance. Last but not least, although without the

supervision of pixel-level annotations for training, our trained segmentation subnet also creates good quality pedestrian segmentation results.

This work was published in a peer-reviewed journal [Guo 21a].

- **A filter pruning method to accelerate the deep neural networks** As modern pedestrian detectors are commonly based on deep neural networks, they require many computations and large amounts of storage. In order to adapt such complex detectors for resource-limited devices, we propose a novel model pruning technique to reduce the computational cost of convolutional neural networks. We first propose an effective filter level pruning criterion. The idea is to measure the importance of a filter according to its connections with the next layer. If a filter has weak connections with the next layer, it means the output of this filter is not important to the network, thus, this filter can be safely removed from the network. Moreover, we provide an observation that filters from the computationally-intensive layers are more sensitive to pruning. In a CNN, filters from different layers contain different amount of computations (per filter). Filters from earlier layers normally contain more computations than those from later layers. When pruning a filter from earlier layers, the FLOPs is more efficiently compressed, but the accuracy of the model also drops more. This observation inspires us to introduce more sparsity into the earlier layers during training, which will achieve better FLOPs compression performance after filter pruning. Compared with the state-of-the-art methods, our proposed method achieves a larger FLOPs compression ratio with a similar or smaller accuracy drop.

  This work was reported in a peer-reviewed journal [Guo 21b].

1.3 List of Publications

1.3.1 Publications in International Journals

- **Zhixin Guo**, Wenzhi Liao, Yifan Xiao, Peter Veelaert, and Wilfried Philips. "Weak Segmentation Supervised Deep Neural Networks for Pedestrian Detection." Pattern Recognition, Volume 119 (2021).

- **Zhixin Guo**, Yifan Xiao, Wenzhi Liao, Peter Veelaert, and Wilfried Philips. "FLOPs-efficient filter pruning via transfer scale for neural network acceleration." Journal of Computational Science, accepted (2021).

- **Zhixin Guo**, Wenzhi Liao, Yifan Xiao, Peter Veelaert, and Wilfried Philips. "An occlusion-robust feature selection framework in pedestrian detection." Sensors 18, no. 7 (2018).

1.3.2 Publications in International Conferences

- **Zhixin Guo**, Wenzhi Liao, Yifan Xiao, Peter Veelaert, and Wilfried Philips. "Deep learning fusion of RGB and depth images for pedestrian detection." In 30th British Machine Vision Conference (BMVC), pp. 1-13. 2019.

- **Zhixin Guo**, Wenzhi Liao, Peter Veelaert, and Wilfried Philips. "Occlusion-robust detector trained with occluded pedestrians." In 7th International Conference on Pattern Recognition Applications and Methods (ICPRAM), pp. 86-94. SCITEPRESS-Science and Technology Publications, 2018.

1.4 Outline

The rest of the thesis is organized as follows:

Chapter 2 provides some background knowledge for this thesis. After introducing the general steps of pedestrian detection, we explain the methods based on different features, i.e., hand-crafted features and convolutional neural network features. Then we introduce the commonly used pedestrian datasets and evaluation methods.

Chapter 3 introduces our proposed method to handle the occlusion problem for pedestrian detection. We exploit the occlusion distribution of the occluded pedestrian samples to guide feature selection strategy during training. Then we extend this method with deep neural networks, by refining the features of each pedestrian according to its occlusion pattern. Experiments demonstrate the effectiveness of our method.

Chapter 4 investigates the way of using depth information for pedestrian detection. Instead of directly fusing the features extracted separately from RGB and depth modalities, we use the depth maps as guidance to effectively reweight the RGB features, which significantly outperforms the state-of-the-arts RGBD pedestrian detection methods.

Chapter 5 focuses on using semantic segmentation to boost the performance of pedestrian detection. We propose a novel framework for joint supervision on pedestrian detection and segmentation. In order to break the dependence of expensive pixel-level segmentation annotations for training, we propose a maximum a posterior (MAP) approach to generate weak segmentation masks from depth images. Then the classification and segmentation results are fused to achieve a further performance improvement.

Chapter 6 focuses on accelerating the inference of deep neural networks for pedestrian detection. We first propose a filter pruning criterion, which assesses the importance of filters according to their connections to the previous layer. Besides, based on our observation that filters from computationally-intensive layers are more sensitive to pruning, we propose a FLOPs-efficient group Lasso approach to guide the network to use fewer filters in computationally-intensive

layers. Experiments show that our method reduces the computation costs of several popular CNN networks.

Chapter 7 presents the general conclusions of this thesis and discusses possible directions for future work.

2

Background

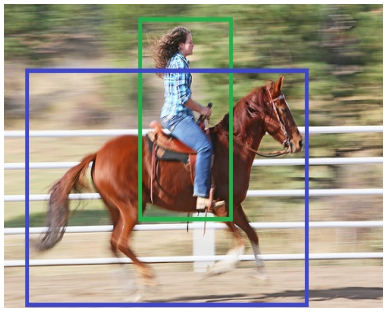
Normally, pedestrian detection is done by transferring the detection task into a binary classification task, which is aimed at determining if a pedestrian exists within a candidate window. Machine learning algorithms are used to learn classification criteria from the training data. But detection is more difficult than classification. As is shown in Fig. 2.1, detection requires not only judging the existence of the target, but also providing the location of the target. Thus, more modules are needed to cooperate with the classification module in a pedestrian detection system. In this chapter, we will introduce the general procedure of pedestrian detection. The rest of this chapter is organized as follows: Section 2.1 introduces each step of pedestrian detection. Section 2.2 and Section 2.3 focus on pedestrian detection methods based on hand-crafted features and convolutional neural network features, respectively. Some popular pedestrian datasets are introduced in Section 2.4, while the evaluation metrics for pedestrian detection are introduced in Section 2.5.

2.1 The General Steps of Pedestrian Detection

A pedestrian detector normally outputs some bounding boxes, which mark the size and location of each pedestrian in the input image. Also a score is generated simultaneously with each bounding box, which indicates the confidence of the detector to this detection result.

Although there are various pedestrian detection algorithms, they can be generally divided into three main parts: Region of Interest (ROI) generation, ROI classification and Non-Maximum Suppression (NMS). The ROI generation module generates candidate pedestrian region proposals, which are marked as blue bounding boxes in Fig. 2.2. Then the role of the ROI classification module is to judge whether an ROI contains a pedestrian or not. In Fig. 2.2, the ROI generation module outputs seven ROIs (We only use seven ROIs as an example. Normally there will be hundreds of ROIs.). After classification, three ROIs are recognized as pedestrians, while the other three are from the background. However, normally several ROIs will be generated around the real pedestrian, and they will likely to be all recognized as pedestrian by the ROI classification

(a) Classification: Panda



(b) Detection: A person and a horse, with their locations

Figure 2.1: Comparison of classification and detection.

module. In Fig. 2.2, we can find the three pedestrian ROIs are actually from the same pedestrian, but a good pedestrian detection result should only contain one bounding box which matches the location of the pedestrian best. Therefore, we need the NMS module to remove the repeated detections. More details of the three modules are introduced below.

2.1.1 ROI Generation

The goal of ROI generation is to output a group of candidate regions, which include all the pedestrians in an input image. On one hand, the number of ROIs is normally huge to make sure no real pedestrian is missed in this step, but on the other, large number of ROIs will also increase the computation cost for the following classification and NMS modules. Therefore, a good ROI generation algorithm should report all the pedestrians with the fewest candidate regions.

The sliding window algorithm is the simplest method for ROI generation: windows with different sizes will slide across the image to obtain all the possible image patches as candidate pedestrian regions. The major advantage of the sliding window algorithm is that it rarely misses real pedestrian regions, because it considers almost all the possible locations and sizes of pedestrians as candidates. But it also requires large computation time, which dramatically reduces the efficiency of the whole detection process.

In order to generate fewer ROIs with higher quality (better matches the sizes and locations of real pedestrians), more specially designed ROI generation methods are proposed, e.g.,, Selective Search [Uijlings 13] and Edge Boxes [Zitnick 14]. The Selective Search method starts by over-segmenting the input image based on the intensity of the pixels using a graph-based segmentation method [Felzenszwalb 04]. Then these initial tiny segments are iteratively merged into larger regions, which are added to the list of candidate ROIs. Hence ROIs of different sizes are generated in a bottom-up approach. The idea of Edge Boxes comes from an observation: the number of contours wholly en-

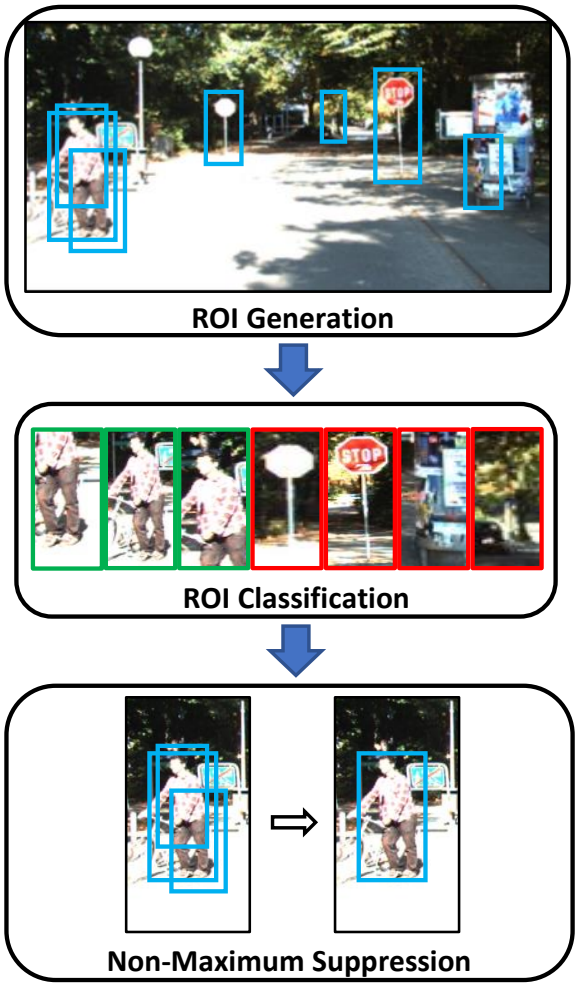


Figure 2.2: General steps of pedestrian detection

closed by a bounding box is indicative of the likelihood of the box containing an object. Thus, this method first extracts edges from the input image, and then outputs bounding boxes which tightly enclose a set of edges as ROIs.

Selective Search and Edge Boxes generate significantly fewer ROIs compared with the sliding window method. It reduces the computation cost during classification, but these methods themselves are more complex and time-consuming. To further reduce the cost of ROI generation, Faster R-CNN [Ren 15] proposed a Region Proposal Network (RPN), which used convolutional neural network (CNN) to generate ROIs. Because RPN shares the feature map with the classification network, the computation cost of ROI generation is significantly cut down. More details about Faster R-CNN and RPN are introduced in Sec-

(a) Before NMS (b) After NMS

Figure 2.3: Before and after NMS

tion 2.3.3.

2.1.2 ROI Classification

The ROI classification module of a pedestrian detector decides if each ROI contains a pedestrian or not. It usually includes two steps: feature extraction and feature classification. Feature extraction can be regarded as an abstract representation of the input image, which transfers the pixel values into features containing high-level semantic information. The original RGB values can also be used as features, but its discrimination ability is not as good as other features, which are more commonly used for classification, for instance, gradient features, edge features and the features automatically learned from data. After obtaining the feature vector of an ROI, a classifier will generate a confidence score of this feature vector belonging to a pedestrian. If the score is below a predefined threshold, this ROI is considered to be background and removed from the ROI list. Otherwise, the ROI is recognized as pedestrian and sent to the NMS module for final processing.

Generally, there are two ways to do ROI classification. One way is to first extract the original image patches from the ROIs and then compute the features of each ROI independently. However, as the ROIs are heavily overlapping, the features from the overlapped image regions will be computed repeatedly. So it is inefficient. Most of the current pedestrian detection methods adopt another way, by first computing the feature map of the whole input image, then finding the features of each ROI from the feature map according to the position of the ROI on the image. In this way, the features from the feature map can be shared by multiple ROIs, which obviously improves the efficiency of classification.

2.1.3 Non-Maximum Suppression

In the ROI generation step, the number of ROIs is normally far more than the number of real pedestrians. Then during classification, ROIs with different sizes located around a real pedestrian are all likely to be recognized as pedestrians. As is shown in Fig. 2.3a, there are multiple ROIs around every real pedestrian in the image, which is not our expected detection result. So the target of NMS is to remove the extra ROIs and leave only one ROI which matches the real

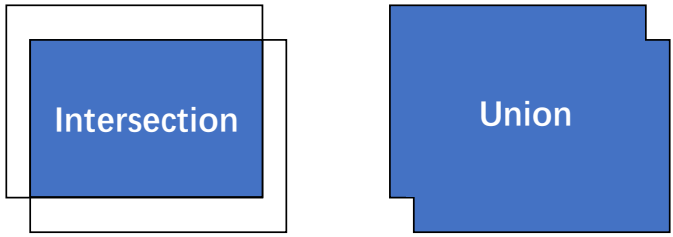


Figure 2.4: Intersection and Union

Algorithm 2.1 Non-Maximum Suppression Algorithm

Input: Candidate ROI list B, corresponding confidence scores S, overlap threshold T and output ROI list B_o

Initialize: $B_o = \emptyset$

while $B \neq \emptyset$ **do**

Step 1: Select B^1 with the highest confidence score $S^1 = \max(S)$

Step 2: Add B^1 to the output ROI list $B_o = B_o \cup B^1$

Step 3: Remove B^1 from the input ROI list $B = B - B^1$, $S = S - S^1$

Step 4: Compute the IOU of B^1 with all the ROIs from B, and denote the ROIs with IOU bigger than T as B^T

Step 5: Remove B^T from the input ROI list $B = B - B^T$, $S = S - S^T$

end while

Output: B_o

pedestrian best, as is shown in Fig. 2.3b.

One simple way to apply NMS is to measure the overlap between two ROIs. If the overlap degree exceeds some threshold value, then the ROI with a lower confidence score will be suppressed. Here the intersection over union (IOU) is used to measure the overlap, which is computed by dividing the intersection area of the two ROIs by the union area of them, see Fig. 2.4. The detailed process of NMS is shown in Algorithm 2.1.

2.2 Pedestrian Detection Based on Hand-crafted Features

2.2.1 Hand-crafted Features

Before the rise of deep learning, hand-crafted features are widely used in pedestrian detection. These features are generated by manually designed rules. Typical hand-crafted features include Haar-like feature [Viola 04], Local Binary Pattern (LBP) feature [Ojala 02], Scale-Invariant Feature Transform (SIFT) [Lowe 04] and Histogram of Oriented Gradient (HOG) fea-

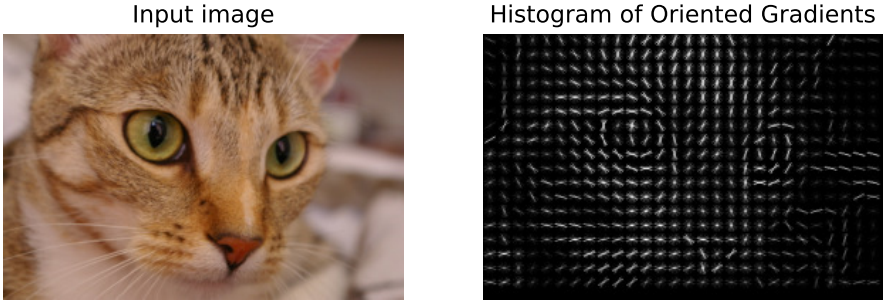


Figure 2.5: Visualization of HOG feature.

ture [Dalal 05]. A landmark research in the development of hand-crafted features is the HOG feature, which was proposed by Dalal and Triggs in 2005.

2.2.1.1 Histogram of Oriented Gradient (HOG) Feature

The basic idea of the HOG feature is to first divide the input image into equal-sized small cells, then use the distribution of the gradients from each cell as features to represent the appearance of pedestrians. Fig. 2.5 shows a visualization of the generated HOG feature from an input image. HOG feature can be created in the following steps:

Step 1: Compute the gradients of pixels. Gradients of pixels from an image are the small changes in x and y directions. Let $I(r, c)$ denote the value of the pixel in (r, c) position of the image, then the gradients in the x and y directions are:

$$G_x(r, c) = I(r + 1, c) - I(r - 1, c), \tag{2.1}$$

$$G_y(r, c) = I(r, c + 1) - I(r, c - 1), \tag{2.2}$$

where $G_x(r, c)$ and $G_y(r, c)$ represent the pixel gradient in the x and y directions, respectively. Then we can obtain the magnitude and orientation of the gradient:

$$G(r, c) = \sqrt{G_x(r, c)^2 + G_y(r, c)^2}, \tag{2.3}$$

$$\theta(r, c) = \tan^{-1}\left(\frac{G_y(r, c)}{G_x(r, c)}\right), \tag{2.4}$$

where $G(r, c)$ and $\theta(r, c)$ represent the magnitude and orientation of the gradient of pixel $I(r, c)$, respectively.

Step 2: Create the histogram of oriented gradient. Divide the image into 8×8 cells, then each cell contains 64 magnitudes and 64 orientations. We generate a histogram containing 9 bins to represent these 128 values on a cell: On the x-axis of the histogram, orientations from 0 degrees to 180

degrees are divided into 9 intervals (bins) in average ; On the y-axis, each pixel contributes its magnitude to the nearest two bins, while the contribution to each bin is decided by the orientation distance between the pixel and that bin (the nearer bin receives higher contribution). Hence the new feature of a cell is a 9-dimensional vector.

Step 3: Normalize the gradients. In order to reduce the lighting variation on local areas of the image, we normalize the gradients by taking 16×16 blocks, i.e., one 16×16 block contains four 8×8 cells. Then we have a 36-dimensional vector $F = (a_1, a_2, ..., a_{36})$ in one block, and a L2-normalization is adopted:

$$\bar{F} = (\frac{a_1}{k}, \frac{a_2}{k}, ..., \frac{a_{36}}{k}), \tag{2.5}$$

$$k = \sqrt{a_1^2 + a_2^2 + ... + a_{36}^2}, \tag{2.6}$$

where $\bar{F}$ denotes the normalized feature vector of F.

Step 4: The HOG feature. The 16×16 block slides across the image with a step of 8 to generate a group of normalized feature vectors. Then these vectors are concatenated together to generate the final HOG feature of the image.

Compared with the other hand-crafted features, the HOG feature has better representation ability for pedestrian and is more robust to lightning and size variations. However, the HOG feature only utilizes the gradient information, which leaves room for further improvement.

2.2.1.2 Aggregated Channel Features

As the representation ability of single feature is limited, Dollar et al. [Dollár 14] proposed Aggregated Channel Features (ACF) which merged several kinds of traditional hand-crafted features together. As is shown in Fig. 2.6, the ACF feature contains ten feature channels: three LUV color channels, one gradient magnitude channel and six gradient orientation channels.

The LUV color space is proposed by the International Commission on Illumination (CIE) to achieve perceptual uniformity, where L stands for luminance, while U and V represent chromaticity values of color images. The RGB color space can be converted into LUV color space in two steps:

Step 1: Convert RGB to XYZ.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.7}$$

Step 2: Convert XYZ to LUV. It requires a referent white (X_r, Y_r, Z_r):

$$\begin{aligned} L &= \begin{cases} (\frac{29}{3})^3 \frac{Y}{Y_r}, & \frac{Y}{Y_r} \leq (\frac{6}{29})^3 \\ 116(\frac{Y}{Y_r})^{\frac{1}{3}} - 16, & \frac{Y}{Y_r} > (\frac{6}{29})^3 \end{cases} \\ U &= 13L(u' - u'_r), \\ V &= 13L(v' - v'_r), \end{aligned} \tag{2.8}$$

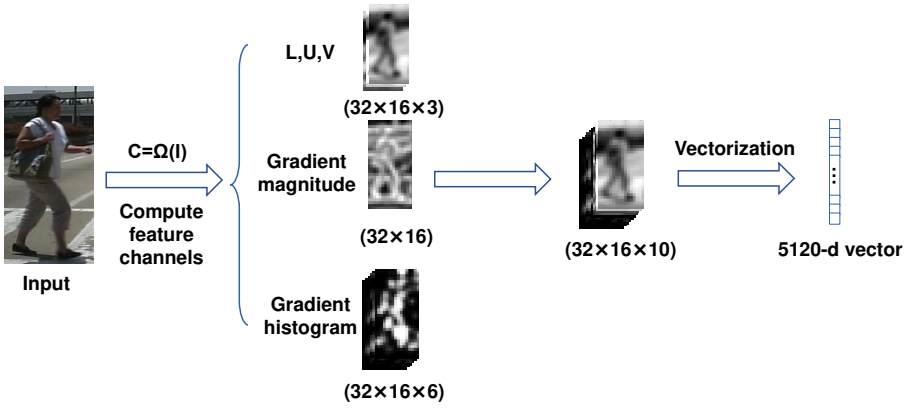


Figure 2.6: The channel features. Here we assume the size of the feature map to be 32×16, then the generated channel feature is a 5120-dimensional vector.

where

$$\begin{aligned} u' &= \frac{4X}{X+15Y+3Z}, \\ v' &= \frac{9Y}{X+15Y+3Z}, \\ u'_r &= \frac{4X_r}{X_r+15Y_r+3Z_r}, \\ v'_r &= \frac{9Y_r}{X_r+15Y_r+3Z_r}, \end{aligned} \tag{2.9}$$

Normally D65 white point is used, then (X_r, Y_r, Z_r) equals to $(0.951, 1, 1.088)$.

The computation of the gradient magnitude and orientation channels are the same as that explained in Section 2.2.1.1. In Fig.2.6, we assume the size of the feature map in each channel is 32×16, then features from the 10 channels are vectorized into a 5120-dimensional vector, which is used for classification. In [Dollár 14], ACF is combined with Decision Trees (DT) classifier for pedestrian detection, which achieves state-of-the-art performances compared with the other hand-crafted feature based methods.

2.2.2 Classification

Among the various machine learning classifiers, the commonly used are Naive Bayes (NB), Decision Tree (DT), Logistic Regression (LR), K-Nearest Neighbor (KNN), Neural Network (NN) and Support Vector Machine (SVM). Different classifiers have different characteristics and applicable conditions. In order to choose a proper classifier for some task, the size and distribution of the training data should be taken into consideration. In pedestrian detection with hand-crafted features, the most widely used classifiers are SVM and Boosted Decision Trees (BDT).

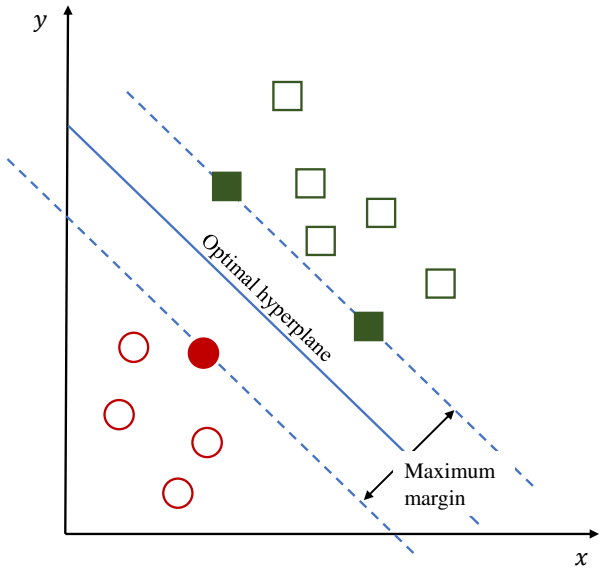


Figure 2.7: A Support Vector Machine on a 2-d feature space.

2.2.2.1 Support Vector Machine

The idea of Support Vector Machine (SVM) [Cortes 95] is to find a hyperplane in an N-dimensional space that distinctly classifies the training data. The optimal choice of the hyperplane is the one that has the maximum margin, i.e., the maximum distance between data points of both classes. As is shown in Fig. 2.7, we indicate the optimal hyperplane on a 2-dimensional feature space. In [Dalal 05], the authors used the combination of HOG+SVM for pedestrian detection, which outperformed the other solutions at that time.

However, SVM is unsuitable for large datasets because of its high computation cost and long training time [Bottou 07]. Also, it is difficult to choose a proper kernel function for SVM to handle the non-linear data.

2.2.2.2 Boosted Decision Trees

Boosted Decision Trees (BDT) is the combination of the Boosting algorithm and Decision Trees. A decision tree takes a set of input features and splits input data recursively based on those features. As is shown in Fig. 2.8, a decision tree recursively generates nodes and leaves, where data is split based on one of the input features at each node, and a class label or probability is outputted on each leave (i.e. terminal node).

But one decision tree is not capable of generating reliable classification results. In order to reduce the risk of overfitting, the Boosting algorithm is used for learning strong classifiers by combining simpler ones. In BDT, each tree is created iteratively and attempts to minimize the errors of previous trees.

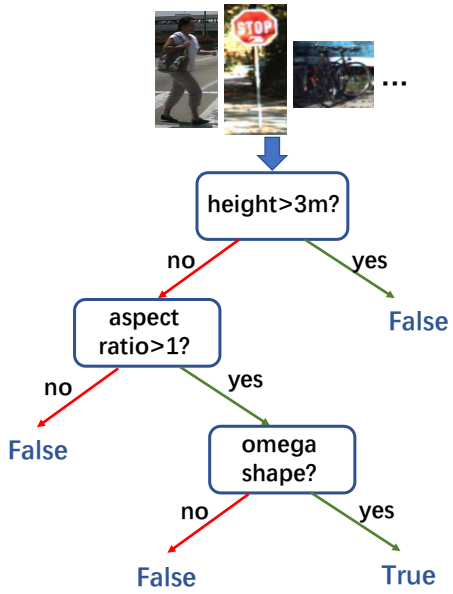


Figure 2.8: A binary decision tree as pedestrian classifier.

Algorithm 2.2 AdaBoost Algorithm

Input: training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, number of weak classifiers M

Initialization: $\omega_i^{(1)} = 1$ for $i = 1$ to N

for $t = 1$ to M **do**

Fit weak classifier t to minimize the objective function:

$$\epsilon_t = \frac{\sum_{i=1}^N \omega_i^{(t)} I(f_t(\mathbf{x}_i) \neq y_i)}{\sum_{i=1}^N \omega_i^{(t)}}$$

where $I(f_t(\mathbf{x}_i) \neq y_i) = 1$ if $f_t(\mathbf{x}_i) \neq y_i$ and 0 otherwise

$\alpha_t = \ln \frac{1-\epsilon_t}{\epsilon_t}$

for $i = 1$ to N **do**

$\omega_i^{(m+1)} = \omega_i^{(m)} \exp(\alpha_t I(f_t(\mathbf{x}_i) \neq y_i))$

end for

end for

Output: α_t

The output of a tree is given a weight relative to its accuracy. Below we introduce the general procedure of AdaBoost [Freund 97], which is the most popular Boosting algorithm.

Suppose we are given training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbb{R}^K$ and $y_i \in \{-1, 1\}$. Let $f_t(\mathbf{x}) \in \{-1, 1\}$ denotes a weak classifier, and a loss function I is

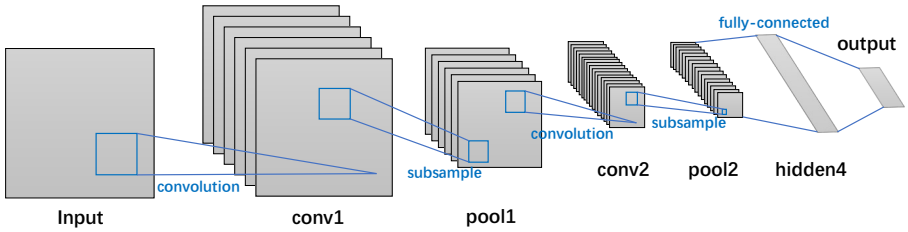


Figure 2.9: The architecture of LeNet

defined as

$$I(f_t(\mathbf{x}), y) = \begin{cases} 0, & f_t(\mathbf{x}_i) = y_i \\ 1, & f_t(\mathbf{x}_i) \neq y_i \end{cases} \tag{2.10}$$

Then, the procedure of AdaBoost is shown in Algorithm 2.2. The final classifier is based on a weighted summation of the weak classifiers:

$$F(\mathbf{x}) = \sum_{t=1}^{M} \alpha_t f_t(\mathbf{x}) \tag{2.11}$$

2.3 Pedestrian Detection Based on Convolutional Neural Networks

Unlike hand-crafted features that require elaborate design of the feature rules, CNN features can be automatically learned from the training data. The pioneering work of utilizing CNN for computer vision tasks is LeNet [LeCun 89], which was proposed in 1989 by Yann LeCun et al. The goal of LeNet is to recognize the handwritten digit (0 to 9) from images. The structure of LeNet is shown in Fig. 2.9. Given an input image, it goes through several convolutional layers, pooling layers and fully connected layers and outputs a 10-dimensional vector indicating the confidence scores for the ten classes. In recent years, with the fast development of computer hardware with high computing ability, e.g., Graphics Processing Unit (GPU), many novel and more complex CNN structures have been proposed. For instance, AlexNet [Krizhevsky 12], VGG [Simonyan 14] and ResNet [He 16]. Modern pedestrian detectors based on these CNNs have achieved a tremendous success. In the rest of this section, we will first introduce the basic components of CNN and its training process, and then explain the principle of Faster R-CNN, which is a popular CNN based object detector.

2.3.1 Basic Structure of CNN

2.3.1.1 Convolutional Layer

The convolutional layer is the key component of CNN, which extracts features from the input image (or feature map) by convolution. The convolution is performed by sliding a kernel over the image, and a single value is generated in each position of the kernel. It is calculated by multiplying together the kernel values and the underlying image pixel values, and then adding these numbers together. Suppose we are given a gray-scale image I and a two-dimensional kernel K, then the convolution can be written as:

$$O(i,j) = \sum_{m=1}^{W} \sum_{n=1}^{H} I(i+m-1, j+n-1) K(m,n), \tag{2.12}$$

where W and H represent the width and height of kernel K, respectively. $O(i,j)$ indicates the value on the (i,j) position of the output feature map O. Note that for the three-dimensional RGB input images, the convolutional kernel should also have three channels, and usually a convolutional layer contains multiple kernels to obtain rich representations of the input information.

In Eq. 2.12, the weight of the convolutional kernel $K(m,n)$ does not change at different positions of the input. This strategy is known as weight sharing, which greatly reduces the number of learnable weights in CNN. After performing convolution, a bias term is often added to the output $O(i,j)$.

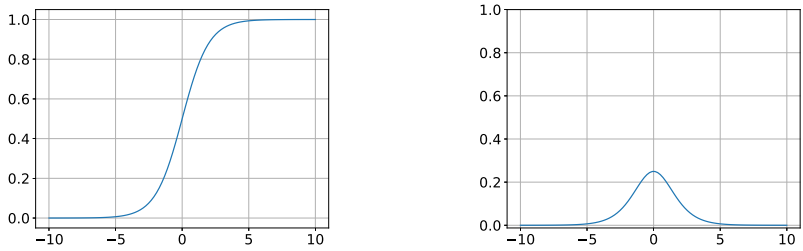
2.3.1.2 Pooling Layer

Adding a pooling layer after the convolutional layer is a typical pattern in modern CNN architectures. The pooling layer actually performs down-sampling to the input feature map. One the on hand, the down-sampled feature maps are more robust to changes in the position of the input feature map; On the other hand, it also reduces the computation load in the following CNN layers.

The Pooling Layer is specified, rather than learned. Two commonly used pooling methods are Max Pooling and Average Pooling. The Max Pooling calculates the maximum value for patches (e.g., the size of the patch will be 2×2 when the down-sampling factor is 2) of a feature map, while Average Pooling calculates the average values for feature patches.

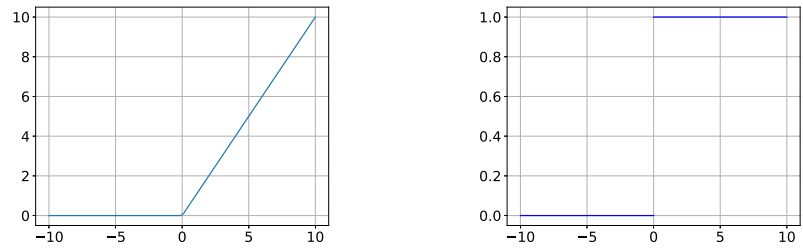
2.3.1.3 Fully Connected Layer

Fully connected layers connect every neuron in one layer and every neuron in the next layer. Normally it forms the last few layers in the CNN structure, which compiles the data extracted by previous layers to form the final output. When a fully connected layer containing $M \times N$ weights is used as the final layer, it compiles an M-dimensional feature vector into an N-dimensional score vector which indicates the classification score for N classes.



(a) Sigmoid (b) Derivative

Figure 2.10: Sigmoid and its derivative



(a) ReLU (b) Derivative

Figure 2.11: ReLU and its derivative

2.3.1.4 Non-Linearity Mapping Layer

Normally a Non-Linearity Mapping layer, also known as the Activation Function, is added after each convolutional layer. The goal of Non-Linearity Mapping is to enable the CNN to simulate any non-linear functions. Additionally, it can help to limit the output of the convolutional layer to a fixed range. The most commonly used non-linear functions are sigmoid, Tanh and Rectified Linear Units (ReLU).

The sigmoid function has an output range of 0 to 1, which suites the binary classification tasks well. It is a non-linear function with continuous gradients. The sigmoid function is shown in Eq. 2.13 and Fig. 2.10a, while the derivative of sigmoid is shown in Eq. 2.14 and Fig. 2.10b. From Fig. 2.10a and Fig. 2.10b we find on either end of the sigmoid function, the output has very small activation to the input, which may lead to vanishing gradient during training.

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \tag{2.13}$$

$$\sigma'(x) = \sigma(x)(1 - \sigma(x)), \tag{2.14}$$

In order to avoid the vanishing gradient problem, ReLU is proposed, as is shown in Eq. 2.15 and Fig. 2.11a. It is simpler in computing, but in some range of ReLU, the gradient can be zero, which prevents some weights from being updated during training. The derivative of ReLU is shown in Eq. 2.16 and Fig. 2.11b,

$$R(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{2.15}$$

$$R'(x) = \begin{cases} 1, & x > 0 \\ 0, & x < 0 \end{cases} \tag{2.16}$$

as there is no derivative at value 0 for ReLU, any value between 0 and 1 can be used as a sub-derivative in practice.

2.3.2 The Training Process of CNNs

In the beginning, all the weights of the CNN are randomly initialized (mostly from a distribution like Gaussian). Thus when a training image is put into the CNN, the output classification scores are also random, which is far from the expected classification result. But a loss function will be used to compare the output of CNN (prediction) and the ground truth label of the input image, and generate a loss value. A small loss value indicates a good prediction result of the CNN for the real class of the input image. Based on the loss function, we can compute the impact of each weight on the loss by using the chain rule. In the last step, each weight of the CNN is updated according to its impact on the loss, which leads to a smaller loss in the next forward pass process.

The loss functions can be classified into classification loss and regression loss. Commonly used classification losses are Cross-Entropy Loss and Hinge Loss, while regression losses include Mean Absolute Error (MAE), Mean Squared Error (MSE) and so on. The Cross-Entropy Loss measures classifiers which have output probability between 0 and 1. It increases when the predicted probability has a larger distance with the true label.

In binary classification problems, the Cross-Entropy function is shown below:

$$L = -(y \log(p) + (1 - y) \log(1 - p)), \tag{2.17}$$

where p and y represent the predicted probability and the true label for the input sample, respectively. When there are more than two classes, the Cross-Entropy is computed as follows:

$$L = -\sum_{c=1}^{M} I(y = c) \log(p_c), \tag{2.18}$$

where M represents the number of classes, while p_c indicates the predicted probability of the input sample belonging to class c. $I(y = c)$ is an indicator operator, which equals one when $y = c$ and zero otherwise.

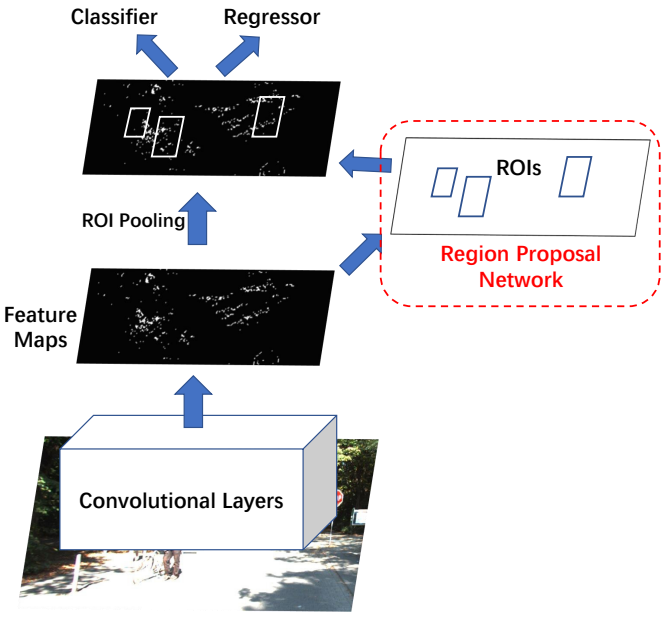


Figure 2.12: The architecture of Faster R-CNN

2.3.3 The Faster R-CNN Detector

Faster R-CNN is a classic CNN based general object detector, which is also widely used in pedestrian detection tasks. Many recently proposed pedestrian detection methods, including our proposed methods in Chapter 4 and Chapter 5, are based on a revised version of Faster R-CNN. Therefore we introduce the principle of Faster R-CNN below.

The architecture of Faster R-CNN is shown in Fig. 2.12. Firstly, the input image goes through some convolutional layers to generate feature maps of the whole image. Then the feature maps are shared between two modules. A Region Proposal Network (RPN), which is also a fully convolutional network, uses the feature map to propose ROIs. Then the classifier will use the features of each ROI (through ROI pooling on the feature map) to output the final classification results.

2.3.3.1 Region Proposal Network

The main novelty of Faster R-CNN lies in the RPN, which is totally different from previous methods which separate ROI generation from the subsequent feature extraction and classification processes. With RPN, all the modules of the detector are put into a unified framework to obtain the global optimum solution. The number of generated ROIs is also obviously smaller than that from other methods, e.g.,, Selective Search or Edge Boxes, which greatly accelerates

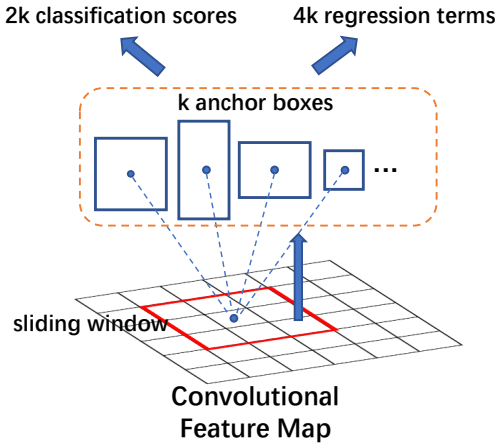


Figure 2.13: The region proposal network

the detection.

In Faster R-CNN, the RPN immediately follows the convolutional layers which generate the feature maps of the input image. The workflow of RPN is shown in Fig.2.13. A 3×3 window slides over the feature map with a step of 1. The center point of each position of the window is mapped to the original image, and k anchor boxes with different sizes and aspect ratios are generated around these points. Then the anchor boxes on each position are used for classification (judge if the target exists) and regression (revise the size and location of the bounding box).

2.3.3.2 Loss Functions

The training loss for RPN is a joint-loss of classification loss L_{cls} and regression loss L_{reg}, given by:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*), \tag{2.19}$$

where i is the index of the anchor box in the mini-batch. p_i and p_i^* represent the predicted classification score and ground-truth label of anchor i. t_i is the predicted regression information which consists of four variables $[t_x, t_y, t_w, t_h]$, and is defined as follows:

$$\begin{aligned} t_x &= (x - x_a)/w_a, \\ t_y &= (y - y_a)/h_a, \\ t_w &= \log(w/w_a), \\ t_h &= \log(h/h_a), \end{aligned} \tag{2.20}$$

where x,y,w and h denote the predicted box's center coordinates and its width and height. The regression target t_i^* is defined in the same way:

$$\begin{aligned} t_x^* &= (x^* - x_a)/w_a, \\ t_y^* &= (y^* - y_a)/h_a, \\ t_w^* &= \log(w^*/w_a), \\ t_h^* &= \log(h^*/h_a), \end{aligned} \tag{2.21}$$

In Eq. 2.20 and Eq. 2.21, variables x, x_a and x^* represent the predicted box, anchor box and ground-truth box, respectively. Same naming rules are also adopted for y, w and h.

In Eq. 2.19, the classification loss L_{cls} is a Cross-Entropy loss, which is introduced in Section 2.3.2 and can be computed by Eq. 2.17. In order to compute the regression loss L_{reg}, a smooth-L1 loss is used which is shown below:

$$L_{reg}(t, t^*) = \sum_{i \in \{x,y,w,h\}} smooth_{L_1}(t_i - t_i^*), \tag{2.22}$$

where

$$smooth_{L_1}(x) = \begin{cases} 0.5x^2, & |x| < 1 \\ |x| - 0.5, & otherwise \end{cases} \tag{2.23}$$

These two loss terms are normalized by N_{cls} and N_{reg}, and weighted by λ. N_{cls} and N_{reg} represent the mini-batch size (e.g., 256) and number of anchor locations (e.g., 2400), respectively. Thus, λ is set as 10 by default to roughly balance these two terms.

The whole architecture can be trained end-to-end. Then the loss function contains four terms in total: two classification losses from RPN and the classifier, respectively, and two regression losses from RPN and the regressor, respectively.

2.4 Pedestrian Datasets

The pedestrian dataset is an important part in pedestrian detection. Normally a pedestrian dataset contains training and test sets. The training set is used to train the pedestrian detector, and also to adjust some super parameters in the detector. The test set is used to evaluate the performance of the trained detector. A good pedestrian dataset should contain pedestrians of various appearances and postures, and also include various conditions such as different lighting and scenes. In this thesis, we have used the following three pedestrian datasets:

1. **Caltech Pedestrian Dataset:** It contains 10 hours of 640×480 30Hz video taken in an urban environment from Los Angeles. There are about 250k frames with a total of 350k annotation bounding boxes. One advantage of the Caltech dataset is that it provides rich occlusion information

(a) Caltech

(b) CityPersons

(c) KITTI

Figure 2.14: Some example images from Caltech, CityPersons and KITTI dataset.

of the pedestrians, because each annotation includes the bounding boxes of both the full body and the visible part of the pedestrian.

2. **CityPersons Dataset:** There are 5000 images with fine pixel-level annotations and 35000 pedestrian annotations in this dataset. The resolution of the images is high: 2048×1024 pixels. It also provides annotation bounding boxes for both the full body and visible part of the pedestrian.

3. **KITTI Dataset:** It provides different forms of data for several computer vision tasks such as stereo, visual odometry, depth completion and detection. For object detection, it consists of 7481 training images and 7518 test images. A total of 80256 objects are labeled.

Some example images from these datasets are shown in Fig. 2.14.

2.5 Detection Performance Evaluation

In order to evaluate the ability of a pedestrian detector, it is crucial to evaluate the quality of the detection results in an effective way. Although it is possible to draw detection results on the image and evaluate them visually, it is difficult to summarize these evaluations for the whole dataset containing thousands of images. Therefore, some specially designed metrics are required.

2.5.1 Detection Results Classification

To evaluate the quality of detection results on multiple images, we first need to judge the correctness of each detection. According to the ground-truth bounding box (gtBB), a detection bounding box (dtBB) can be classified into four categories:

1. **True Positive (TP):** The gtBB belongs to the target, and the dtBB correctly predicts it as the target.
2. **True Negative (TN):** The gtBB belongs to the background, and the dtBB correctly predicts it as the background.
3. **False Positive (FP):** The gtBB belongs to the background, but the dtBB predicts it as the target by mistake.
4. **False Negative (FN):** The gtBB belongs to the target, but the dtBB predicts it as the background by mistake.

Based on these classifications, several basic metrics are obtained: Precision, Recall, Miss Rate and False Positive Rate. These metrics measure the quality of detection results from different point of views.

1. **Precision:** It means the proportion of the correct detection results in all the detection results, which is computed as:

$$precision = \frac{TP}{TP + FP} \tag{2.24}$$

   It focuses on measuring the confidence in a given detection result.

2. **Recall:** It is the proportion of the correct detection results in all the ground truth targets, which is computed as:

$$recall = \frac{TP}{TP + FN} \tag{2.25}$$

   It focuses on measuring how many true targets are detected.

3. **Miss Rate:** It is the proportion of the undetected targets in all the real targets, see below:

$$MissRate = \frac{FN}{TP + FN} \tag{2.26}$$

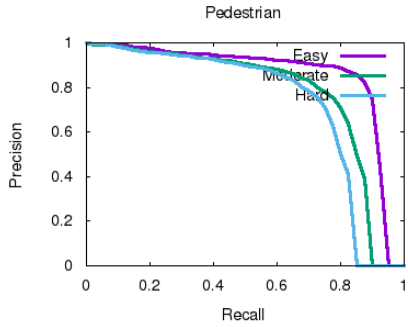


Figure 2.15: The P-R curve of our proposed method in Chapter 5.

It also focuses on the detected ratio of the whole targets. The sum of Recall and Miss Rate equals to 1: $recall + MissRate = 1$

4. **False Positive Rate:** It measures the proportion of incorrect detections in all the ground truth negatives, see below:

$$MissRate = \frac{FP}{TN + FP} \tag{2.27}$$

2.5.2 Evaluation Curves

Based on the metrics introduced in Section 2.5.1, some evaluation curves are created to evaluate the detection performance more intuitively. The most commonly used evaluation curves are Precision-Recall (P-R) curve and the Miss Rate-False Positive Per Image (MR-FPPI) curve.

A P-R curve is a graph with Recall values on the x-axis and Precision values on the y-axis. An ideal detector should report both high Precision and high Recall. However, in practice detectors need to find a good trade-off between the two. When more detection results are allowed (the lower threshold for confidence scores), it normally brings higher Recall, but as more FT detections are also included into evaluation, the Precision drops. A good P-R curve should be close to the top-right corner, which means with the increase of Recall, the Precision drops slowly. Fig. 2.15 illustrates the P-R curves of our proposed method in Chapter 5, which is submitted to the KITTI benchmark. Obviously, the method performs better on the Easy test set.

We can also compute an Average Precision (AP) value for each curve, which is a widely used evaluation metric for object detection tasks. By definition, the AP is to find the area under the P-R curve:

$$AP = \int_0^1 p(r)dr \tag{2.28}$$

In practice, different benchmark computes AP in different ways. KITTI calculates AP by averaging the Precision values on 40 uniformly distributed

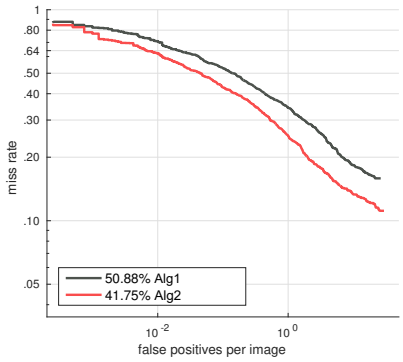


Figure 2.16: An illustration of the MR-FPPI curve.

Recall positions. Pascal VOC competition (after the year 2010) samples the curve at all unique recall values whenever the maximum precision value drops, which obtains a more accurate estimate of the area under the PR curve.

The MR-FPPI curve is proposed by Piotr Dollar, and adopted in the Caltech pedestrian dataset. As is shown in Fig. 2.16, both the x-axis and y-axis are log based. When the threshold of the classification score is lower, more FPs are allowed for each image, while the Miss Rate decreases as well. Different from the P-R curve, a lower MR-FPPI curve indicates better performance. Thus in Fig. 2.16, Alg2 outperforms Alg1.

3

Partial Occlusion Handling in Pedestrian Detection

3.1 Introduction

Although pedestrian detection has been intensively studied in the past decade, occlusion remains a challenging problem. According to a recent survey, in a video sequence captured in busy urban streets, over half of the pedestrians are occluded to various levels in typical images as seen by self driving cars. The accuracy of normal pedestrian detectors can drop significantly when a pedestrian is heavily occluded. However, many applications like self-driving vehicles require very high reliability of the pedestrian detection results.

Before the rise of deep learning, researchers focused on the deformable part-based models (DPM) [Felzenszwalb 10] to overcome occlusion issues. Instead of conventionally treating pedestrians as a whole object [Dalal 05], DPM models separate a pedestrian into different body parts. The occluded parts can then be handled properly and the influence of changed appearance caused by occlusion is eliminated. But this requires an accurate estimation of the visibility of different body parts [Wang 09, Ouyang 16], which makes the training of DPM models delicate and complex. Besides, the high computation cost of training a set of detectors from different body parts and the fusion of their detection scores further increase the difficulties of this kind of methods.

By observation the Caltech Pedestrian dataset [Dollar 12], Dollar et al. indicate that most occluded pedestrians fall into a limited number of occlusion types (7 types account for 97% of all occlusions in the dataset). Inspired by this discovery, some researchers train a set of occlusion-specific models to improve the detection of occluded pedestrians [Mathias 13]. However, this method suffers from similar problems as the DPM based methods. Training distinct detectors for different occlusion patterns is not only costly, but also difficult because of the need for a sufficient number of specific occluded pedestrian samples. In addition, a proper method to merge the results of distinct detectors is also needed, because the occlusion pattern is unknown during detection.

In practice, as many other methods we have observed that using occluded pedestrian samples for training can reduce the detection accuracy, because the detectors cannot distinguish a real pedestrian from the occluding object, which results in learning wrong characteristics of pedestrians. In order to exploit the occluded pedestrian samples in a more effective way, we propose a new feature selection strategy based on the occlusion distribution of the training samples. Experimental results demonstrate that our method achieves a better occlusion-handling performance.

Moreover, we extend our method to the more popular deep learning based models. The idea is to first compute the specific occlusion pattern of each pedestrian, and then reweight the convolutional features accordingly. Features from the non-occlusion region will be assigned a bigger weight, which results in a higher classification score for the occluded pedestrian samples. This significantly reduces the detection miss rate for the occluded pedestrians.

The rest of this chapter is organized as follows. After reviewing the related work in Section 3.2, we introduce our proposed conventional method and deep learning based method in Section 3.3 and Section 3.4, respectively. The conclusion is drawn in Section 3.5.

3.2 Related Work

Over the past decades a great effort has been made to improve pedestrian detection performance. In this section, we first discuss the development of the boosted detection framework on which we base our work, and then review the state-of-the-art occlusion-handling methods.

In 2004, Viola and Jones [Viola 04] pioneered a detection architecture that computed features very efficiently with integral images. Adaboost [Friedman 00] was used to train a cascade of decision trees, and a sliding window strategy was employed to search each potential region of the image. This successful structure was then combined with some more powerful features. Dalal and Triggs [Dalal 05] proposed the histogram of oriented gradient (HOG) feature which since then has been widely used. Some researchers combined HOG with additional features [Walk 10,Wang 09], while the integral channel features (ICF) proposed by Dollar et al. [Dollár 09a] showed excellent discriminative power. In the ICF detector, the VJ boosted structure [Viola 04] selects appropriate ICF features and train powerful classifiers. Inspired by the ICF architecture, a variety of improvements have been made to achieve better performance as well as efficiency. The FPDW [Dollár 10] detector is proposed to accelerate the detection by estimating some features across different image scales, instead of computing them explicitly. Benenson et al. [Benenson 12] move this estimation from the detection to the training stage and further improve efficiency. To strengthen the representation ability of features, SquaresChnFtrs [Benenson 13] uses all sizes of square feature pools instead of random rectangular pools and get better performance. More recently, Dollar et al. [Dollár 14] propose the ACF detector, which far outperforms its contemporary detectors. Based

on ACF, LDCF [Nam 14] demonstrates that decorrelation of local feature information is helpful. Some recent work continues to improve the detection performance by generalizing more powerful features [Zhang 15], exploring the model capacity [Ohn-Bar 16] or combining detectors with deep Convolutional Neural Network (CNN) models [Angelova 15].

As for occlusion-handling, one idea is to first estimate the visibility of different body parts of a pedestrian, and then handle the visible and occluded parts separately. Wang et al. [Wang 09] propose to use the response of HOG features of the global detector to estimate the occlusion likelihood map. The final decision is made by applying the pre-trained part detectors on the fully visible regions. To further explore the visibility correlations of body parts, Ouyang et al. [Ouyang 16] employ a deep network, which supplements the DPM detection results. In [Enzweiler 10], the authors obtain the degree of visibility by examining occlusion discontinuities extracted from additional depth and motion information. However, it is very difficult to accurately estimate the visibility of every body part, which impairs the performance of this kind of methods.

Another idea is to train a set of distinct detectors for different occlusion patterns. In [Wojek 11], a full-body DPM detector and six part-based detectors, at low and high resolution, are trained and combined to make the final decision. While in [Mathias 13], Mathias et al. train a more exhaustive set of ICF detectors (16 different occlusion patterns). By biased feature selection and reusing trained detectors, the training cost can be ten times lower compared to conventional brute-force training. Besides single-person models, some multi-person occlusion patterns are investigated to handle occlusion in crowded street scenes. Tang et al. [Tang 14] make use of the characteristic appearance pattern of person-person occlusion, and train a double-person detector for occluded pedestrian pairs in the crowd. A similar model is proposed in [Ouyang 13b] where the authors use a probabilistic framework instead of Non-maximum Suppression (NMS) to deal with strong overlaps. The main problem of these methods is the high cost for training distinct occlusion-specific detectors. In addition, it is also challenging to design an appropriate rule to fuse the results from several detectors.

In this chapter, we propose a new feature selection strategy for pedestrian detection, which guides the detector to select more features from unoccluded regions during training. Moreover, we design a deep learning based method to first recognize the occlusion pattern of each pedestrian, and then refine the features to get higher detection scores for the occluded pedestrians.

3.3 Occlusion Handling through Occlusion Distributions

3.3.1 Conventional ACF Detector

As our method is based on the conventional ACF [Dollár 14] detector (see Section 2.2.1.2 and Section 2.2.2.2), we briefly review the training process of

ACF, which helps to explain how occlusion probability is included in our proposed method. ACF employs a boosting structure that greedily minimizes a loss function for the final decision rule

$$F(m_i) = \sum_{t=1}^{N_t} \alpha_t f_t(m_i), \tag{3.1}$$

where the final classification result $F(m_i)$ is a weighted sum of N_t weak classifiers $f_t(m)$. $m_i \in \mathbb{R}^K$ denotes the feature vector of the i-th pedestrian sample, while α_t indicates the weight of each weak classifier. At each training iteration t, a decision tree $f_t(m)$ is trained and its weight α_t is optimized.

A decision tree $f(m)$ is composed of a stump $h_j(m)$ at every non-leaf node j. The tree is grown from the root node by recursively learning a stump at a time, which produces a binary decision:

$$h_j(m) = p\,\mathrm{sign}(m[k] - \tau), \tag{3.2}$$

where $m[k]$ denotes the k-th feature of m, while p and τ are the polarity element $\{\pm 1\}$ and threshold for feature $m[k]$, respectively. During the training of a stump $h_j(m)$, the goal is to optimize k, p and τ, which minimizes the classification error:

$$\epsilon = \sum \omega_i \mathbf{1}_{\{h_j(m_i) \neq L_i\}}, \tag{3.3}$$

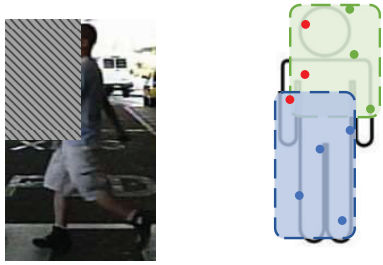
where $\mathbf{1}_{\{\ldots\}}$ is the indicator operator. $h_j(m_i)$ and L_i indicate the classification result and the true class of the input feature map m_i, respectively. ω_i is the weight of each sample, which is updated after each iteration t (the weight of the misclassified samples will be increased).

Specifically, when training the stump $h_j(m)$, the classification error of feature $m[k]$ can be rewritten as follows by combining Eq. 3.2 and Eq. 3.3:

$$\epsilon_{m_i[k]} = \sum_{m_i[k] \leq \tau} \omega_i \mathbf{1}_{\{L_i = +p\}} + \sum_{m_i[k] > \tau} \omega_i \mathbf{1}_{\{L_i = -p\}}, \tag{3.4}$$

so during training the feature selection is conducted by selecting the most optimal feature $m[k]$ and its corresponding parameters p and τ which minimize the classification error: $\{k, p, \tau\} = \arg\min\ \epsilon_{m[k]}$.

However, when we introduce occluded pedestrian samples into training, this feature selection criterion concerning only the classification error will not be sufficient. The selection is susceptible to occlusion as it will use features from occluded pedestrian regions. For example, each time a feature in the occluded region of a pedestrian is selected, it harms the learning of pedestrian characteristics because it is very different from a pedestrian but is eventually regarded as a pedestrian feature. In the next section we will propose a more robust feature selection criterion under occlusion situations.



(a) (b)

Figure 3.1: Pedestrian samples are manually occluded in the top-left corner with non-person image patches randomly cut from negative samples. Some features in the occlusion region (marked as red) are selected by weak classifiers but they are not reliable.

3.3.2 Proposed Method

In the ACF detector, the final decision is made by a combination of weak classification results. As is shown in Fig. 3.1b, each weak classifier will select several features from the feature map. The idea is that we can improve the detection of occluded pedestrians by guiding the classifiers to choose more reliable features, which are located in the areas with a lower probability of being occluded.

As we are more concerned about the location of features, we redefine $m_i \in \mathbb{R}^{C\times W\times H}$ as a three-dimensional feature map of the i-th pedestrian sample, where C, W and H represents the number of channels, width and height of the feature maps (feature maps from all the candidate regions are resized into equal size, e.g. $10 \times 16 \times 32$). Then during training the classification error function of one feature (see Eq. 3.4) can be rewritten as:

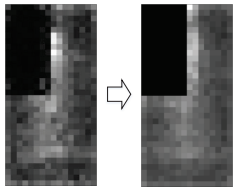
$$\epsilon_{m(c,x,y)} = \sum_{m_i(c,x,y)\leq\tau} \omega_i \mathbf{1}_{\{L_i=+p\}} + \sum_{m_i(c,x,y)>\tau} \omega_i \mathbf{1}_{\{L_i=-p\}}, \tag{3.5}$$

where $m(c, x, y)$ represents the feature located in the (x, y) position of the feature map on the c-th channel.

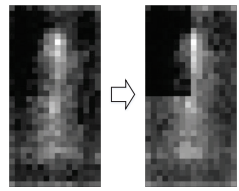
In order to explain the proposed method clearer, we first assume that all the training samples have the same known occlusion region (Fig. 3.1a) in Section 3.3.2.1, then we extend this assumption to a more realistic situation and propose a biased feature selection strategy in Section 3.3.2.2. Last but not least, we explain the proposed feature selection strategy to the real situation to handle occluded pedestrians in Section 3.3.2.3.

3.3.2.1 Simple Situation: Known Occlusions

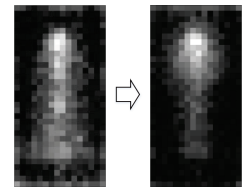
To illustrate the proposed method, we first assume that all samples have the same occlusion region in the top-left corner (the occlusion region occupies 1/4 of



(a) Under simple situation: avoid selecting features from the occluded area during training



(b) Under complex situation: lower the chances of selecting features from the occluded area druing training



(c) Under real situation (the proposed method): select features according to the occlusion distribution of the training samples.

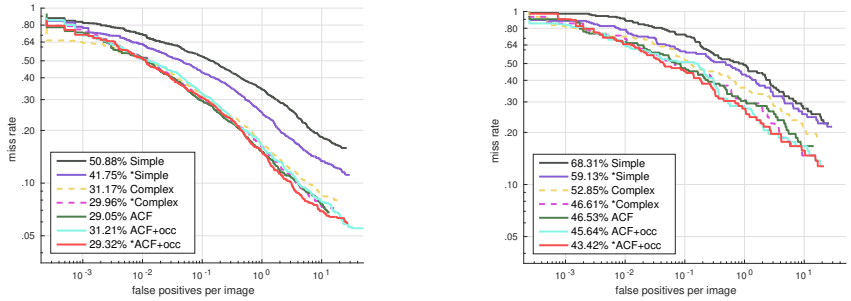
Figure 3.2: We create three assumptions to simulate the occlusion situations of the training samples: (a) simple situation: all the training samples have the same occlusion region (top-left corner) (b) complex situation: 20% of the training samples are occluded in the same area (c) real situation: each training sample can have occlusions at any area. Under each assumption, we show how the distribution of selected features is changed after we adopt new feature selection strategies. Features located in the brighter area are more frequently selected by the model during training.

the image area, see Fig. 3.1a). This is easy to handle because we are certain that features that are located in the occluded region do not represent characteristics of pedestrians. Thus if the features selected by weak classifiers are in the top-left corner of the feature map (red pixels in Fig. 3.1b), they must be unreliable. What we need to do is to just restrict the locations of features, forbidding the selection of features in the occluded region. So the new classification error function for feature $m(c, x, y)$ is defined as:

$$\epsilon'_{m(c,x,y)} = \begin{cases} \epsilon_{m(c,x,y)}, & x > \frac{W}{2} \vee y > \frac{H}{2} \\ +\infty, & x \leq \frac{W}{2} \wedge y \leq \frac{H}{2} \end{cases} \tag{3.6}$$

where $\epsilon_{m(c,x,y)}$ indicates the classification error function defined in Eq. 3.5. Hence during training a feature is selected by minimizing the new classification error: $c, x, y = \arg\min\ \epsilon'_{m(c,x,y)}$.

We simulate this simple case by covering all the training samples (including positive and negative samples) with non-pedestrian image patches in the top-left corner. These image patches are randomly cut from the negative samples. In Fig. 3.2a we can see the distribution of the selected features of the model trained with our manually occluded samples. As expected, although the detector selects most of the features from the non-occluded region, a few features from the top-left corner are also used. Therefore, we also train a new model in which we forbid the selection of features in the top-left corner. In Fig. 3.3, *simple* and **simple* indicate the performance of the original ACF model and the new ACF model without using the features of occluded regions, respectively, while **simple* outperforms *simple* both in the Reasonable and the Partial Oc-



(a) Reasonable (b) Partial occlusion

Figure 3.3: Performance of models trained in different situations. The lower, the better. (a) Reasonable: less than 35% occluded pedestrians (including full-visible ones) (b) Partial occlusion: less than 35% occluded pedestrians (excluding full-visible ones)

clusion cases. It demonstrates that when training with occluded pedestrian samples, we can improve the detection by avoiding the selection of unreliable features.

However, in this case it is easy to confirm the locations of unreliable features because the occlusion regions of all the training samples are assumed to be fixed. In the next subsection, we will extend this assumption to a more complex and realistic one.

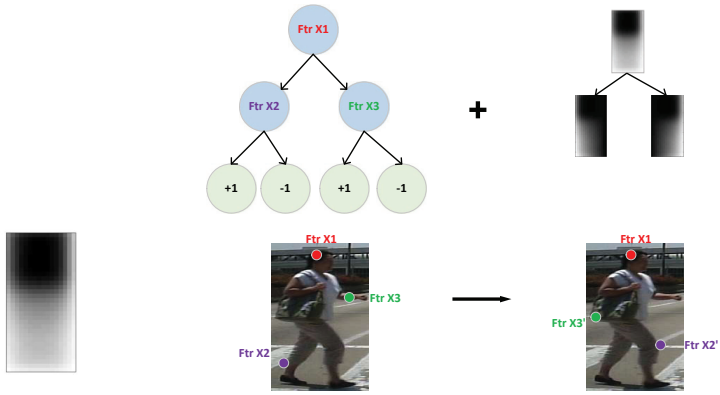
3.3.2.2 Complex Situation: Training with Sample Mixtures

Now we assume a more complex situation: Only 20% of the training samples are occluded, and still only the top-left corner of these samples are fully occluded, as is explained in Section 3.3.2.1. In this case, the features located in the top-left area of the feature map can no longer be discarded directly, because they are not always occluded in all the samples, and they contain useful information in the rest 80% unoccluded training samples. Therefore, we improve the feature selection criterion defined in Eq. 3.5 by taking into account the occlusion probability of a feature among the training samples. In short, if two features from the input feature map have very similar classification errors during training, we prefer the one located in the area with lower probability of being occluded.

The new classification error of feature $m(c, x, y)$ for this case is defined as:

$$\epsilon'_{m(c,x,y)} = \epsilon_{m(c,x,y)}(1 + \lambda P(c, x, y)), \tag{3.7}$$

where $P(c, x, y)$ indicates the occlusion probability of feature $m(c, x, y)$, while λ acts as a constant weight factor to control the impact of the occlusion cost. In this case, because 20% of the training samples are occluded in the top-left corner, $P(c, x, y)$ is computed as:



(a) Occlusion distribution map. Brighter region indicates higher occlusion probability

(b) feature X2 and X3 selected by orignial training process are replaced by X2' and X3' according to the occlusion probability distribution in each decision node

Figure 3.4: Proposed method

$$P(c,x,y) = \begin{cases} 0 & x > \frac{W}{2} \vee y > \frac{H}{2} \\ 0.2 & x \leq \frac{W}{2} \wedge y \leq \frac{H}{2} \end{cases} \tag{3.8}$$

In Eq. 3.7, in order to ensure that the classification error is always the prior consideration, we set the λ value much smaller than 1 (here we have made multiple experiments and finally set $1/25$ as the λ value. A bigger λ will overweight the influence of occlusion and sometimes sacrifice some very discriminative features) so that a feature with low occlusion probability will be preferred to a feature that is barely better but has much higher occlusion probability.

In Fig. 3.2b, after adopting the new classification error function (Eq. 3.7) during training, the trained model selects fewer features from the top-left corner of the feature map. The detection performances are shown in Fig. 3.3. We use *complex* and **complex* to represent the detection performances of the original model (using Eq. 3.5 as classification error) and the new model (using Eq. 3.7 as classification error) on the Caltech dataset, respectively. **complex* shows an improvement from 31.17% to 29.96% (lower log-average miss rate indicates better performance) in the Reasonable case (Fig. 3.3a) and an improvement from 52.85% to 46.61% in the Partial Occlusion case (Fig. 3.3b). In addition, we find that the new model **complex* has achieved comparable performance to that of the default ACF model trained with non-occlusion pedestrian samples (29.05% and 46.53%).

So far we have only focused on the artificially occluded samples whose occlusion regions are fixed. For real pedestrian samples, we need to handle the occlusion which may occur in any part of a person.

3.3.2.3 Real Situation

Now we come to the real situation. Unlike the artificially occluded samples used in Section 3.3.2.1 and 3.3.2.2, the occlusion regions of real pedestrian samples are unfixed. We adopt the same feature classification error function of Eq. 3.7 proposed in Section 3.3.2.2, but for real pedestrian samples we need to compute the $P(c, x, y)$ of Eq. 3.7 according to the occlusion information from the training samples.

We use the Caltech pedestrian dataset, which provides bounding box annotations for both the full body and the visible part of each pedestrian. Thus we can easily recognize if a feature $m_i(c, x, y)$ is located in the occluded area for each sample m_i. Let N_{all} and $N_{(x,y)}$ denote the number of pedestrian samples and the number of samples that have occlusions on the location (x, y), then the occlusion probability of the features can be computed as:

$$P(c, x, y) = \frac{N_{(x,y)}}{N_{all}}, \tag{3.9}$$

hence by combining Eq. 3.7 and Eq. 3.9, a feature is selected by minimizing the new classification error: $c, x, y = \arg\min \epsilon'_{m(c,x,y)}$.

Fig. 3.4a shows the probability distribution map of occlusions for all the training samples in Caltech pedestrian dataset. From the map we notice that the occlusion distribution is not uniform. The lower part of a pedestrian is more likely to be occluded while the head region suffers the least from occlusion. This result conforms with our common sense that the head region is often the most visible part, because most obstacles are located on the ground.

Fig. 3.4b shows how our proposed method impacts the selection of features. With the old feature classification error function (Eq. 3.4), feature X1, X2 and X3 are selected by the weak classifiers. Then we calculate the occlusion distribution map of each splitting node and employ Eq. 3.7 and Eq. 3.9 to select more robust features. For example, feature X2 is replaced by a more reliable feature X2' which is less likely to be occluded according to the occlusion distribution map. In Fig. 3.2c, we see that with the proposed method, the new model is more biased to select the features from the area with lower probability of being occluded (for example, the head area).

Fig. 3.3 clearly shows how our proposed method improves the detection of occluded pedestrians. We first introduce occluded pedestrian samples and train a model named as *ACF+occ*. It shows an improvement of the average miss-rate from 46.53% to 45.64% for the Partial Occlusion cases (Fig. 3.3b). This demonstrates that the introduction of occluded samples in the training process improves the detection of occluded pedestrians. Unsurprisingly, there is also a reduction of performance in the Reasonable case (Fig. 3.3a). However, when we use our method to train a new model (named as **ACF+occ*), there is further improvement in both the Reasonable and Partial Occlusion cases. **ACF+occ* successfully eliminates the impact of occlusion samples and achieves a performance comparable to the default *ACF* model for the Reasonable case, while in Partial Occlusion case the average miss rate further reduces to 43.42%.

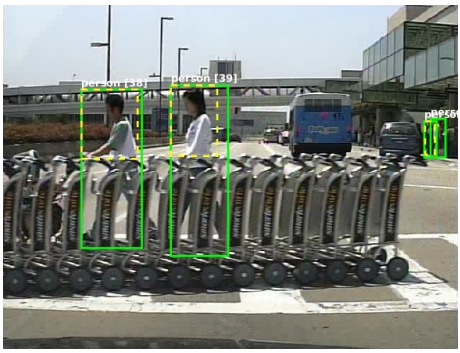


Figure 3.5: Caltech pedestrian dataset

3.3.3 Experiments

Now that we have proposed a new method of utilizing occluded pedestrian samples, in this section we demonstrate its effectiveness with several experiments. The experiments are divided into two parts. In the first part, we exhaustively explore the potential of ACF models and obtain our best detector trained with non-occlusion pedestrian samples. In the second part, we further improve the performance by introducing occluded pedestrians and training them with the proposed method.

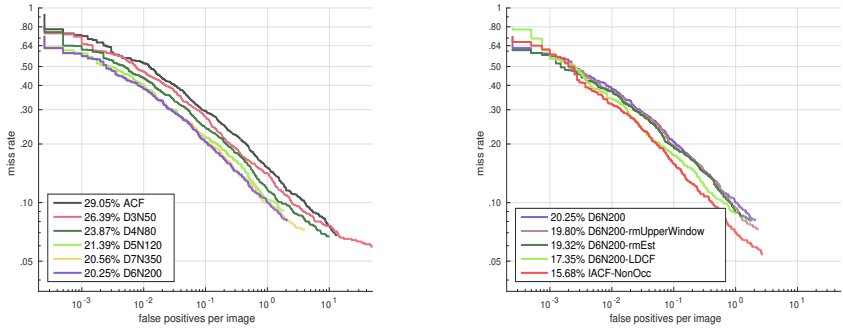
In both parts of the experiments, we double the default feature size to 32×64 pixels per channel. Positive samples in most experiments are obtained by sampling the Caltech video data with a skipping step of 10, while a smaller skipping step (more dense sampling) is also used in some cases. We employ some of the modifications proposed in [Ohn-Bar 16]: a scaling (factor 1.1) is used to augment the number of positive samples by 3 (scaling in horizontal, vertical and both directions), while the randomness handling is also employed to make the results more reliable.

3.3.3.1 Caltech Pedestrian Dataset

In our experiments, we use the Caltech Pedestrian dataset [Dollar 12], which is currently one of the most popular pedestrian detection benchmarks. It consists of 250k labeled frames with 350k annotated bounding boxes. In particular, each partially occluded pedestrian is annotated with two bounding boxes (see Fig. 3.5), which indicate the full body (in green) and the visible part (in yellow), respectively. This visible region information will be used for training with occluded pedestrians in Section 3.3.2.3.

3.3.3.2 Training with Non-occlusion Samples

We investigate the optimal training parameters by gradually increasing the maximum depth of the decision trees. For larger model capacity (deeper trees),



(a) (b)

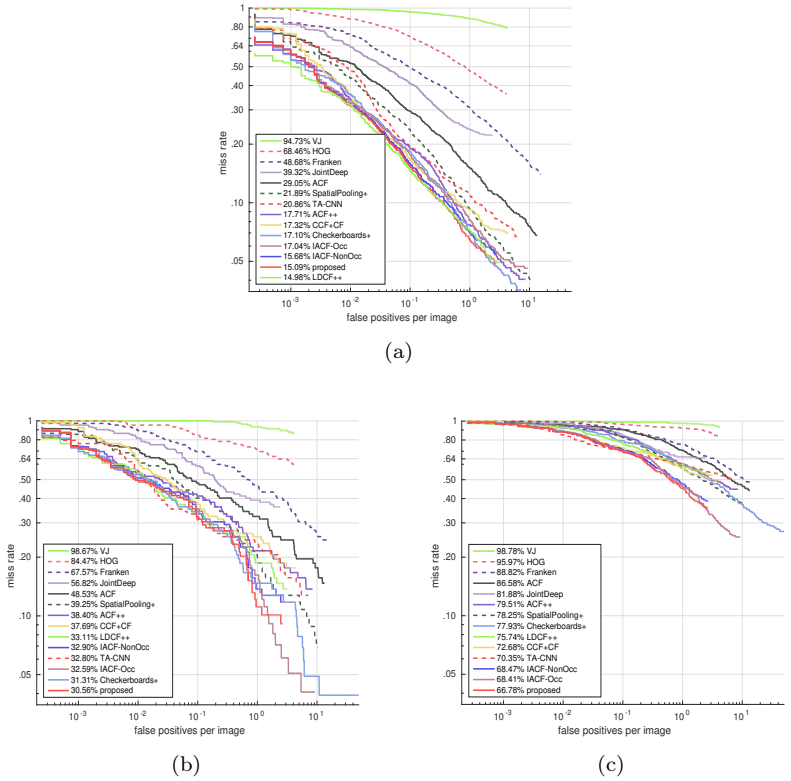
Figure 3.6: Performance of models trained with fully visible pedestrian samples.

a bigger collection of training samples is needed to explore the full potential of the model.

We start our experiment with a tree depth equal to 3 and gradually increase the depth to 7 with more training samples (see Fig. 3.6a). At the start, we train a maximum depth 3 model with 50k negative samples, named *D3N50* (D indicates the maximum depth while N indicates negative samples of the model), which already outperforms the ACF detector trained with default settings (maximum depth 5 with 50k negative samples). We conduct multiple experiments to find the optimal data size under a maximum tree depth. When additional data does not lead to an obvious improvement, we consider the model to be saturated. In this way, tree depths from 3 to 7 are evaluated with 50k, 80k, 120k, 200k and 350k negative samples respectively. For deeper trees (depth 6 and 7), more dense sampling with a skipping step of 4 is employed to enlarge the number of positive samples.

In Fig. 3.6a we observe a quick saturation of performance: when the maximum depth reaches 6, additional data seems to help little, even with deeper trees. Thus we fix *D6N200* as our baseline setting and further improve it in Fig. 3.6b.

In the experiments, we also take advantage of some prior knowledge. Instead of exhaustively searching the image with sliding windows, we remove those candidates from the upper 1/3 of the image, because it is impossible for pedestrians in the real world to appear in that area. It obviously reduces the calculation and avoids some false positive detections in the non-pedestrian regions (*D6N200-rmUpperWindow*). Furthermore, we obtain *D6N200-LDCF* and *D6N200-rmEst* by employing feature decorrelation filtering [Nam 14] and removing feature estimation as suggested in [Ohn-Bar 16]. With all the above modifications, we obtain our best performance detector trained only with non-occlusion samples named *IACF-NonOcc* (improved ACF detector trained with non-occlusion samples), which is comparable with the state-of-the-art ACF



(a) (b) (c)

Figure 3.7: Comparison with state-of-the-art methods in (a) Reasonable, (b) Partial Occlusion and (c) Heavy Occlusion cases

based method LDCF++ [Ohn-Bar 16], see Fig. 3.7a).

3.3.3.3 Training with Occluded Samples

In Fig. 3.7, we evaluate the performance of our models: *IACF-NonOcc*, *IACF-Occ* and *proposed* by comparing with some state-of-the-art methods. The *IACF-NonOcc* is trained only with full-visible pedestrian samples, as explained in Section 3.3.3.2. Then we introduce occluded pedestrian samples into the training stage and obtain *IACF-Occ*. In order to show the influence of the occluded samples, all the parameters and modifications of *IACF-NonOcc* are kept unchanged. At last, the *proposed* model achieves the best performance by employing our new training method proposed in Section 3.3.2.

The methods we use as comparison include *VJ* [Viola 04], *HOG* [Dalal 05], *Franken* [Mathias 13], *JointDeep* [Ouyang 13a], *ACF* [Dollár 14], *SpatialPooling+* [Paisitkriangkrai 16], *TA-CNN* [Tian 15b], *ACF++* [Ohn-Bar 16], *LDCF++* [Ohn-Bar 16], *CCF+CF* [Yang 15] and *Checkerboards+* [Zhang 15]. We obtain the detection results of the above methods from the website of Cal-

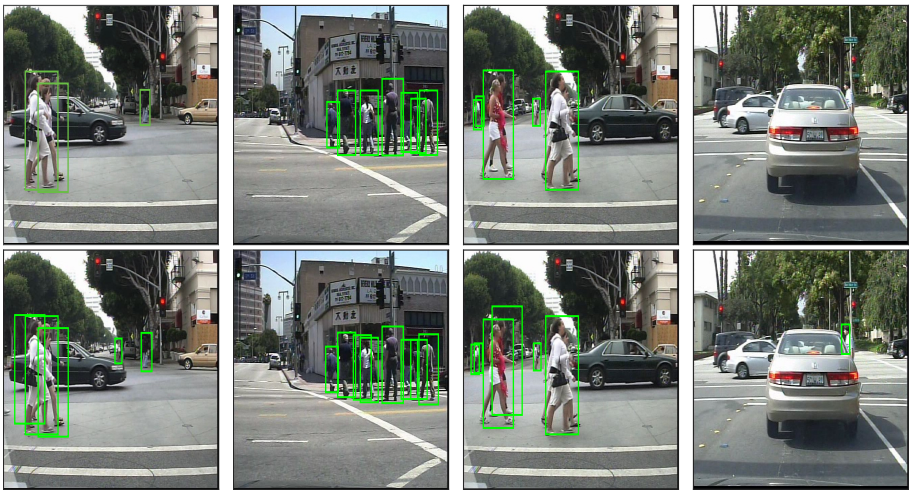


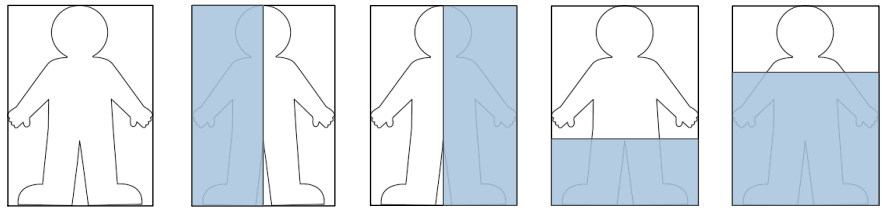
Figure 3.8: Detection results of *IACF-NonOcc* (first row) and *proposed* (second row) on Caltech Pedestrain dataset. For comparison both the detectors are kept with same number of false positives, while the *proposed* model successfully recognised more occluded pedestrians.

tech Pedestrian dataset.

In the commonly used Reasonable case (Fig. 3.7a), we observe an obvious performance decline of nearly 2% (15.68% to 17.04%) after we introduce the occluded samples during the training stage (*IACF-Occ*), while the proposed method (*proposed*) successfully eliminates this drop and even slightly outperforms the baseline *IACF-NonOcc* (it reaches 15.09% compared with 15.68%).

The occlusion test cases represent the strong occlusion handling ability of our proposed model. In the partial occlusion case (Fig. 3.7b), the introduction of occluded samples slightly improves the performance from 32.90% to 32.59%, while the proposed model further obtains the best result of 30.56%. More impressive results appear in the heavy occlusion case (Fig. 3.7c), which achieves a significant improvement of nearly 10% (from 75.74% to 66.78%) over the state-of-the-art ACF based model LDCF++, while the efficiency remains the same (only the selected features and thresholds are changed during detection). Some detection results of the Caltech Pedestrian dataset are presented in Fig. 3.8. We observe a more robust detection of occluded pedestrians with our method.

Although our proposed method in this section achieves better detection performances for the occluded pedestrians, one major disadvantage of this method is that it applies an average occlusion distribution to all the pedestrians without considering the specific occlusion condition of each pedestrian. In the next section, we will propose a new deep learning based approach to fix this problem.



(a) Non-occlusion (b) Left (c) Right (d) Bottom 1/3 (e) Bottom 2/3

Figure 3.9: Five occlusion patterns. The region marked with blue indicates the occluded part of the human body.

3.4 Occlusion Handling through Occlusion Patterns

Typically, machine learning based detectors treat all the pedestrians equally. Pedestrian samples with different occlusion conditions will go through a CNN network to generate an n-dimensional feature vector F and produce a classification score S, which can be computed in Eq. 3.10:

$$S = \sum \omega_i F_i, i = 0, 1, ..., n \tag{3.10}$$

where ω_i represents the weights of the classification network (for convenience we assume the classification network to be a 1-layer fully connected neural network), which determines the contribution of feature F_i to the final classification score. For an occluded pedestrian sample, features from the unoccluded area should be assigned higher weights to enhance their contribution to the classification score. However, after training, the weights of the classification network are fixed, which limits its ability to adapt to different occlusion patterns. Therefore, we propose to reweight the features instead. We will first recognize the occlusion pattern of each pedestrian, and then reweight the features according to its occlusion pattern.

As is shown in Fig. 3.9, we classify all the occlusion patterns into five categories: fully visible (non-occlusion), left-occlusion, right-occlusion, bottom 1/3 occlusion and bottom 2/3 occlusion.

3.4.1 Network Structure

Our proposed occlusion handling framework is shown in Fig. 3.10. It is based on the popular Faster R-CNN [Ren 15] detector, which is illustrated in the upper part of Fig. 3.10. A CNN network (e.g. VGG16 net [Krizhevsky 12]) takes an RGB image as input and extracts the convolutional feature of the whole image. Then a region proposal network (RPN) is used to generate regions of interest (ROIs). After cropping the features of each ROI from the feature map, a ROI-Pooling layer is applied to generate same-length convolutional features for each

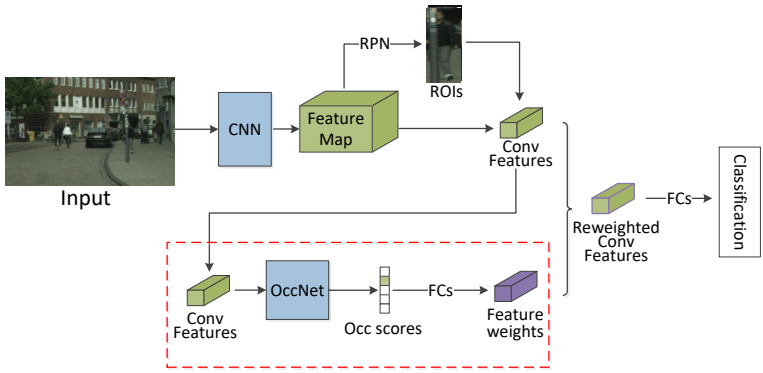


Figure 3.10: Overview of our occlusion handling network.

ROI. Then the features will go through a classification network, usually some fully connected layers, to produce the classification results and bounding box regression information.

We add a subnet (see the dashed bounding box in Fig. 3.10) on the baseline Faster R-CNN structure to recognize the occlusion pattern of each ROI and reweight its convolutional features accordingly. The OccNet contains one convolutional layer and two fully connected (FC) layers, which outputs a five-digit occlusion score vector (according to the occlusion patterns defined in Fig. 3.9) to indicate the occlusion pattern of the input candidate pedestrian. For example, when the ROI contains a pedestrian occluded on the left part, the expected output of the OccNet is $(0, 1, 0, 0, 0)$. Then the output occlusion score vector goes through several FC layers and is reshaped into a feature weights vector which is used to reweight the original convolutional features.

3.4.2 Network Training

As is explained in Section 3.4.1, the role of the OccNet is to recognize the occlusion pattern of the input pedestrian. Thus during the training process, normal pedestrian annotations containing only the coordinates of the pedestrians are not enough, and we need the ground truth occlusion pattern labels for the training samples to train the OccNet. In this subsection, we will first explain how we generate the ground truth occlusion pattern labels, and then introduce the joint training process of the whole network.

3.4.2.1 Generating Occlusion Pattern Labels

In order to generate ground truth occlusion pattern labels for training, we use the annotations from the CityPersons dataset [Zhang 17], which provides the bounding box annotations for both the full body and the visible part of each pedestrian. Each annotation from the CityPersons dataset contains eight numbers: $(x, y, w, h, x_v, y_v, w_v, h_v)$, where x, y, w and h represent the top-left

corner, width and height of the full body, respectively, while x_v,y_v,w_v,h_v represents the corresponding information for the visible part of the body. In order to compute the occlusion pattern label p ($p \in \{0,1,2,3,4\}$) of each pedestrian sample, we compare the matching degree of the pedestrian with the five occlusion pattern templates (see Fig. 3.9).

Let o_i denote the Intersection over Union (IOU) between the visible part of the pedestrian and the visible part of the i-th occlusion pattern template. We compute o_i according to the eight numbers from the annotation. For example, when computing o_2 (the matching degree of the pedestrian sample with the right-occluded template), the coordinates of the visible part of the pedestrian and that of the occlusion template are (x_v, y_v, w_v, h_v) and $(x, y, \frac{w}{2}, h)$, respectively. Then the upper bound and lower bound of the intersection on the direction of x-coordinate are $\min(x+\frac{w}{2}, x_v+w_v))$ and $\max(x, x_v)$, respectively. Thus we obtain the width of the intersection $w_I^{[2]}$ in Eq. 3.11:

$$w_I^{[2]} = \max(\min(x + \frac{w}{2}, x_v + w_v) - \max(x, x_v), 0), \tag{3.11}$$

then we compute the height of the intersection $h_I^{[2]}$ in the same way:

$$h_I^{[2]} = \max(\min(y + h, y_v + h_v) - \max(y, y_v), 0), \tag{3.12}$$

the areas of the intersection and the union can be computed as follows:

$$s_I^{[2]} = w_I^{[2]} \times h_I^{[2]}, \tag{3.13}$$

$$s_U^{[2]} = w_v \times h_v + \frac{w}{2} \times h - s_I^{[2]}, \tag{3.14}$$

Thus we obtain o_2 in Eq. 3.15:

$$o_2 = \frac{s_I^{[2]}}{s_U^{[2]}}. \tag{3.15}$$

After computing o_i for all the five occlusion pattern templates, the value of p is set as the index of the occlusion pattern with biggest o_i:

$$p = \arg\max_i(o_i), \tag{3.16}$$

so the new annotation we used for training contains nine numbers: $(x, y, w, h, x_v, y_v, w_v, h_v, p)$.

3.4.2.2 Joint Loss Function of the Network

One advantage of this method is the whole framework can be trained end-to-end. The optimization target is a joint loss function containing five terms:

$$\begin{aligned} L = &\lambda_1 L_{RPNcls} + \lambda_2 L_{RPNbbox} + \lambda_3 L_{FRCNNcls} \\ &+ \lambda_4 L_{FRCNNbbox} + \lambda_5 L_{OCC}, \end{aligned} \tag{3.17}$$

Table 3.1: Accuracy of occlusion pattern detection results on CityPersons dataset.

Patterns	Non-occlusion	Occlusion	left	right	bottom 1/3	bottom 2/3
Number	3157	1474	270	304	558	342
True Positive	2984	1328	232	250	517	329
Accuracy	94.5%	90.1%	85.9%	82.2%	92.6%	96.3%

where the first four terms remain the same as the Faster R-CNN losses explained in Section 2.3.3, including two classification losses (L_{RPNcls}, $L_{FRCNNcls}$) and two bounding box regression losses ($L_{RPNbbox}$, $L_{FRCNNbbox}$) for RPN and classification networks. L_{OCC} is the multinomial logistic loss between the output occlusion score vector from the OccNet and the generated occlusion pattern labels (see Section 3.4.2.1).

3.4.3 Experiments

3.4.3.1 Implementation Details

The whole framework is trained on the CityPersons dataset [Zhang 17], which contains 2975 training images and 1575 test images. The total number of pedestrians amounts to 35000. In order to improve the detection performance of the baseline Faster R-CNN detector, we remove the fourth max-pooling layer to double the size of the convolutional feature maps. We train for 50k iterations, starting from a learning rate of 0.001 and reducing it by 10 at 20k and 40k iterations. Other settings of the model are unchanged as in the original Faster R-CNN paper [Ren 15]. Training on a single GTX 1070 takes 12 hours.

3.4.3.2 Recognizing Occlusion Patterns

In this subsection, we evaluate the output occlusion pattern results from the OccNet. The evaluation is conducted on a video sequence from the CityPersons dataset, which is captured in a busy urban street scenes in Stuttgart. The accuracy of each occlusion pattern is shown in Table 3.1.

In Table 3.1, we get 4631 pedestrian detection results in total, including 3157 pedestrians recognized as non-occlusion and 1474 pedestrians recognized as partial occlusion. The accuracy of non-occlusion and occlusion samples are 94.5% and 90.1%, respectively. For left and right occluded samples, the accuracies are 85.9% and 82.2%, respectively, while bottom occluded samples have a higher accuracy (92.6% and 96.3% for bottom 1/3 and 2/3 occluded, respectively). Some occlusion pattern detection results of the video sequence are shown in Fig. 3.11.

3.4.3.3 Comparing with State-of-the-art Methods

We also compare our proposed method with the state-of-the-art methods on the CityPersons test set, which is shown in Table 3.2. The performances are evaluated with three test cases: 1) The Reasonable case includes pedestrians

Figure 3.11: Some occlusion pattern detection results from a video sequence in the CityPersons dataset, where different occlusion patterns are marked with different colors. Green: non-occlusion, Blue: left-occlusion, Purple: right-occlusion, Yellow: bottom 1/3 occlusion, Red: bottom 2/3 occlusion.

Table 3.2: Comparison with state-of-the-art methods on CityPersons dataset. Here we use Average Miss Rate for evaluation, and a smaller value indicates better performance.

Methods	MR(Reasonable)	MR(Occlusion)	MR(All)
MS-CNN	13.32%	51.88%	39.94%
FasterRCNN	12.97%	50.47%	43.86%
Reputation Loss	11.48%	52.59%	39.17%
Proposed	11.45%	**42.72%**	38.97%
AdaptiveNMS	11.40%	46.99%	**38.89%**
OR-CNN	**11.32%**	51.43%	40.19%

higher than 50 pixels and with a visibility of more than 65% of the body. 2) The Occlusion case includes pedestrians higher than 50 pixels and with a visibility from 20% to 65%. 3) The All case includes pedestrians higher than 20 pixels. OR-CNN obtains the best (lowest) miss rate of 11.32% on the Reasonable case, while AdaptiveNMS achieves the best miss rate of 38.89% on the All case. Our proposed method achieves the best performance of 42.72% on the Occlusion case, which is obviously better than the others. It demonstrates the effectiveness of our method on the occlusion-handling problem. Furthermore, our method also performs well on the non-occlusion cases, which achieves very similar performances with the best miss rates on the Reasonable and All cases.

3.5 Conclusions

In this chapter, we proposed two methods to handle the challenging occlusion problem in pedestrian detection.

The first method is based on the conventional methods which uses hand-crafted features. Limited by the representation ability of the conventional machine learning models (such as SVM and random forests), we find most conventional methods do not use occluded pedestrian samples for training. This is because when training with occluded pedestrian samples, the detector can not distinguish real pedestrian from the occluding objects, and therefore it will learn some wrong characteristics from occlusions and reduce the accuracy of detection. In order to make better use of the occluded pedestrian samples, we propose a new feature selection strategy during training, which guides the detector to choose more features from the regions with lower probabilities of being occluded. Our method achieves better occlusion-handling performance.

We also explore our method on deep learning based models. Instead of using an average occlusion distribution of all the training samples, we first recognize the occlusion pattern of each pedestrian, and then accordingly reweight its convolutional features. The idea is to assign a bigger weight to features from non-occlusion regions, which results in a higher classification score for the occluded pedestrians. Therefore, the detection miss-rate of the occluded pedestrians is significantly reduced. The whole framework can be trained end-to-end.

The research presented in this chapter has resulted in one conference paper [Guo 18a] and one journal paper [Guo 18b].

4

Pedestrian Detection in RGB-D Images

4.1 Introduction

In the previous chapter, we proposed two occlusion-handling methods for pedestrian detection on both hand-crafted feature based models and deep learning based models, but only RGB images captured from visible cameras are used in these methods. In practice, it is also very helpful to resort to other modalities of data (e.g. thermal or depth) to handle challenging conditions such as partial occlusions and insufficient illumination.

Recent work has shown that depth images could be used as a good complementary data source to RGB images [Premebida 14, Gupta 14, Hoffman 16, Ophoff 19]. Unlike RGB images, depth images will not provide fine appearance details of the objects, but rather silhouettes shapes and equally importantly their distance to the sensors. In addition, depth images show a better description of the relationship of the spatial positions of the objects in a scene, which enriches the representations of each target object. For example in Fig. 4.1, if we apply a cluster analysis on the depth region of the pedestrian in the green bounding box, we can clearly distinguish the pedestrian from the occluding object in front of it and the background behind it. Depth images captured with other sensors (e.g. LiDAR) are also not affected by bad lighting conditions, in contrast to RGB images from visible-light cameras, which are heavily affected.

Looking back at the development of RGBD pedestrian detection research, depth information was used as additional hand-crafted features in traditional computer vision techniques. Detectors trained with the combination of depth and RGB features showed an improved performance [Gavrila 07, Wu 11, Wang 12]. After the rise of Convolutional Neural Networks (CNNs) [Krizhevsky 12, Simonyan 14, He 16], most RGBD object detection frameworks apply two parallel CNN network streams for RGB and depth modalities, and fuse them in early or late layers [Gupta 14, Hoff-

Figure 4.1: An RGB and depth image pair from KITTI dataset. With a cluster analysis of the depth region in the bounding box, the person is distinguished from the occluding object and the background, which can be utilized to boost the detection performance.

man 16, Ophoff 19, Eitel 15, Ertler 17]. However, the above work did not explore the geometric information of the depth images, which can be very useful. Some researchers take advantage of the depth images to reduce the searching space of candidate regions [Benenson 12, Jafari 14], but the spatial relationship illustrated in Fig. 4.1 is still not well exploited.

In this chapter, we focus on how to better utilize depth images for pedestrian detection. We demonstrate that by refining RGB features with the guidance of depth information, the detection performance can be significantly improved, even without directly extracting features from depth images. Our major contribution is threefold:

- First, we propose to use depth images to guide the reweighting of the convolutional features extracted from RGB images. It helps the classification network to pay more attention to the features from the target pedestrian instead of on the non-pedestrian regions (e.g. occluding objects, background). This framework shows a significant improvement on pedestrian detection, both for the visible pedestrians and partially occluded ones.

- Second, we compute the sizes of all the candidate pedestrian proposals according to the depth information, and filter out those candidates with unreasonable size. It obviously improves the efficiency of our proposed RGBD detector.

- Third, our proposed RGBD detector significantly outperforms the baseline method Faster R-CNN and the state-of-the-art RGBD detection algorithms on the KITTI dataset.

4.2 Related Work

With the rise of CNNs, significant improvements have been made in object detection [Krizhevsky 12, Simonyan 14, Girshick 14, Girshick 15, Ren 15, He 16]. One pioneering work using deep CNNs for pedestrian detection was proposed in [Sermanet 13], which combined CNNs with unsupervised multi-stage feature learning. Some methods [Angelova 15, Hosang 15, Tian 15b, Tian 15a] follow the RCNN [Girshick 14] structure, which integrates external proposals with CNN framework and outperforms the traditional hand-crafted feature based pedestrian detectors [Dollár 14, Nam 14] by a significant margin. Later, Fast-RCNN [Girshick 15] and Faster-RCNN [Ren 15] further improved the RCNN framework by sharing computations during feature extraction and region proposal generation. Some adaptations of the Faster-RCNN framework [Zhang 17, Zhang 16] have reached state-of-the-art results in pedestrian detection. In this chapter, an adapted Faster-RCNN framework that takes RGB and depth images as inputs is used in our experiments.

Depth images have been intensively studied as a complementary data source for object detection. Early works [Spinello 11, Jafari 14, Choi 11] extended hand-crafted features from RGB data with depth information to provide a richer representation. [Spinello 11] and [González 13] designed HOG-like features in the depth channel and trained an SVM classifier for pedestrian detection. Depth is also used to reduce the search space of candidate regions [Camplani 16, Jafari 14, Zhang 13]. In most CNN based methods, features are extracted from RGB and depth images independently and fused together for further classification. Gupta et al. [Gupta 14] proposed to encode depth image into a three-channel (height above ground, horizontal disparity and angle to gravity) depth image, and used a CNN network pre-trained on RGB images to extract depth features. Tanguy et al. [Ophoff 19] exhaustively trained several models to explore the optimal fusion layer of RGB and depth features in CNN networks. Gupta et al. [Gupta 16] transferred supervisions from RGB images to depth images to achieve a richer representation. Instead of equally extracting features from RGB and depth images, our method takes advantage of the depth modality to refine the RGB features, which outperforms the existing RGBD detection methods.

4.3 Proposed Method

An overview of our proposed framework is illustrated in Fig. 4.2. Two subnets are added to the default Faster-RCNN structure to take advantage of the depth image: Depth Guided Feature Reweighting Net and ROI Filtering net. The depth region of each ROI is put into the reweighting net, where a binary map is produced to indicate the real pedestrian areas and guides the reweighting of the RGB convolutional features for better classification scores. Depth images are also used in the ROI generation process: In the depth guided ROI filtering net, the size of each ROI will be computed according to the depth information. ROIs

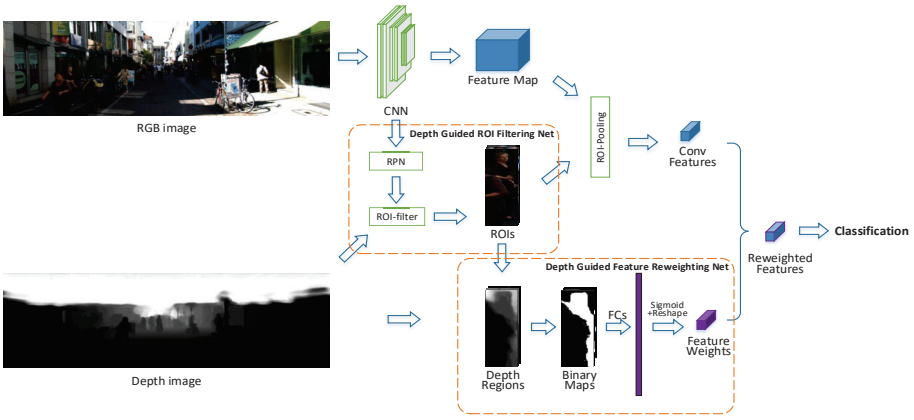


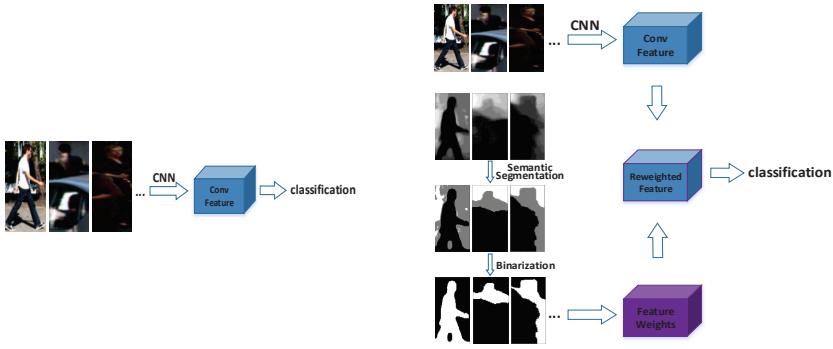
Figure 4.2: The architecture of our proposed RGBD pedestrian detector. It is based on a Faster-RCNN structure with two subsets: (1) Depth Guided Feature Reweighting Net, which produces the weights for the convolutional feature of each ROI. (2) Depth Guided ROI Filtering Net, which evaluates each ROI with the perspective projection relationship. The reweighted feature are used in classification.

with reasonable size will be kept for classification, while others are removed from the candidate list.

4.3.1 Depth Guided Feature Reweighting

In this section, we adopt ideas similar to those in Section 3.4, i.e., that the convolutional features of each pedestrian can be refined according to external guide information to improve detection performance. The main difference between this section and Section 3.4 is what is used as the guidance information. In Section 3.4, we used occlusion pattern information as the guidance to enhance the features from the unoccluded regions, whereas in this section we utilize depth information to guide the feature refining process.

In Fig. 4.3, we compare the common convolutional features and our depth-guided reweighted features. Normally, all the candidate pedestrian proposals will go through a CNN to generate convolutional features for classification. However, the weights of CNN are fixed after training, which makes it difficult to adapt to different pedestrian samples during the inference process. For example, the features from the non-pedestrian areas (e.g. occluded region, background) should contribute less to the final classfication score, while the features from the pedestrian areas should be assigned bigger weights to enhance the detection performance. Thus we propose the depth guided feature reweighting net, which utilizes depth information to indicate the real pedestrian areas and then reweight the convolutional features accordingly. The flowchart of the reweighting net is illustrated in Fig. 4.3b. The depth image is used to guide the generation of feature weights in three steps:



(a) typical CNN classification process (b) CNN classification with reweighted features

Figure 4.3: The comparison of the typical CNN classification process with our method. We reweight the convolutional features according to the depth information, which increase the contribution of features from the pedestrian.

Semantic Segmentation: We apply a K-means (k equals three to handle occlusion situations) clustering for the depth region of each ROI to segment it into three parts. Normally the top depth layer, which is nearest to the camera and has the smallest depth value, represents the pedestrian (see the first sample in Fig. 4.3b). But when the pedestrian is partially occluded (see the second and third samples in Fig. 4.3b), the occluding object becomes the top depth layer, while the pedestrian and background occupy middle layer and bottom layer, respectively. Therefore, it is unreliable to decide the true pedestrian segment according to its depth values, and we propose a simple but effective strategy to select the pedestrian segment in the next step.

Binarization: The goal of binarization is to generate a binary map that marks the pixels from the pedestrian region with 1 and all other pixels with 0. To realize this, we propose to choose the segment which is nearest to the center of the head region, which we define as the top 1/3 of each ROI, see Fig. 4.4. Note that we only consider the pixels from the head region, which has the smallest probability of being occluded. $(\frac{w}{2}, \frac{h}{6})$ represents the center point of the head region, where w and h indicate the width and height of the ROI region, respectively. $D(m)$ represents the average euclidean distance between the head center and all the pixels from segment m. It is computed by Eq. 4.1:

$$D(m) = \frac{1}{N^m} \sum_{i=1}^{N^m} \sqrt{(x_i^m - \frac{w}{2})^2 + (y_i^m - \frac{h}{6})^2}, m = 1, 2, 3 \tag{4.1}$$

where (x_i^m, y_i^m) indicates the location of the i-th pixel from the m-th segment, while N^m indicates the total pixel numbers from the m-th segment. Pixels from the segment with the smallest $D(m)$ will be set as 1, while the other pixels are all set to 0.

Weights Learning: The binary maps will go through two fully connected

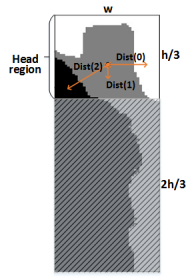


Figure 4.4: Computing the average distance between the head center and the pixels from layer m. Only pixels from the head region are taken into account.

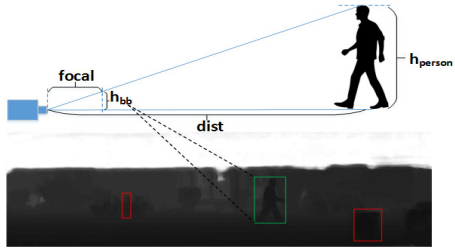


Figure 4.5: The projection relationship of a camera. We can compute the distance between the pedestrian and the camera through depth information, and further estimate the height of the pedestrian.

layers to generate the weight vector, which is followed by a sigmoid computation and a reshape operation to get the final weight cube. It will have the same size with the convolutional feature. After a element-wise multiplication, the reweighted feature will be sent to the classification network.

In the next section, we will further use depth map to eliminate unreasonable candidates during the generation of ROIs.

4.3.2 Depth Guided ROI Filtering

In the typical Faster-RCNN structure, an RPN is used to generate ROIs for further classification. In order to make sure true positives will not be lost at this stage, the number of ROIs is normally large. On the one hand, a large number of ROIs harm the efficiency of the detector. On the other hand, some high score false positive ROIs are still difficult to recognize during the classification. In this section we propose to take advantage of the depth information to efficiently reject some unreasonable (too high to be a human) ROIs.

As is shown in Fig. 4.5, the height of the bounding box h_{bb} can be computed by Eq. 4.2:

$$h_{bb} = \frac{focal}{dist} \cdot h_{person} \tag{4.2}$$

where $dist$ is the distance between the pedestrian and camera, while $focal$ and h_{person} represent the focal length of the camera and the real height of the pedestrian, respectively. According to the depth value, $dist$ can be computed by comparing with the maximum range of depth map:

$$dist = \frac{depth}{255} \cdot range \tag{4.3}$$

where the $depth$ value of the pedestrian is computed by the same method described in Section 4.3.1, and $range$ equals to 80 meters in our experiments. In Eq. 4.2, we assume the range of h_{person} to be [1m, 2m], which should include

most of the pedestrian heights in real life. The bounding boxes with a height out of this range are considered to be unreasonable, and removed from the ROI candidates.

4.4 Experiments

In this section, we will first introduce the implementation details. Then we will show some experiments of the two subnets in our framework. In the end, we will compare our method with the baseline method and some state-of-the-art RGBD detection methods.

4.4.1 Implementation Details

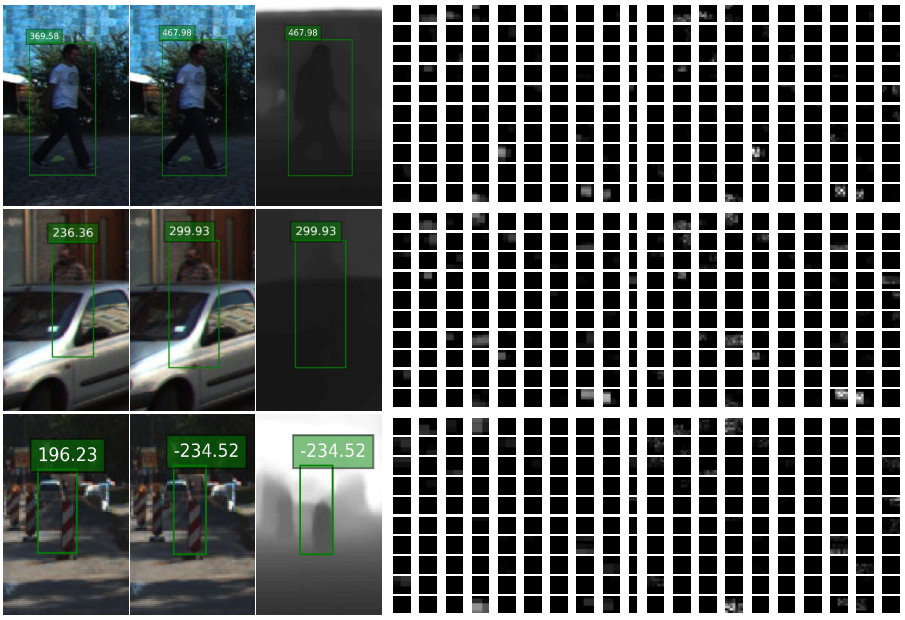
Our RGBD pedestrian detection framework is based on an adapted version of Faster-RCNN: In order to adapt to the shape of pedestrians, we use four aspect ratios of [1, 1,5, 2, 2.5] and five scales of [1, 2, 4, 8, 16] in the RPN, which allows more human-like regions and small regions during the ROI generation. We use the VGG16 net pre-trained on ImageNet as the feature extraction network, but the last max-pooling layer is removed, which reduced the feature stride from 16 pixels to 8 pixels to further improve the detection of small pedestrians. To evaluate the performance of our adapted Faster-RCNN detector, we apply a 3-fold cross validation in the KITTI training set, which shows the average precision (AP) improvements from 61.5% to 67.3%. This adapted Faster-RCNN will be one of our baseline methods.

The depth images are generated by the method proposed by [Dimitrievski 18], and are computed from the LiDAR cloud points.

4.4.2 Experiments on Feature Reweighting

In order to evaluate our feature reweighting method, we compare the detection results of our baseline Faster-RCNN (denoted as FRCNN) structure and the Faster-RCNN with our Depth Guided Feature Reweighting Net (denoted as FR+RW).

Fig. 4.6 shows three detection samples, including a fully visible pedestrian, a partially occluded pedestrian and a false detection. We find that after depth-guided feature reweighting, our FR+RW method significantly increases the classification score of the true positive detections (row 1 and row 2), while reduces the classification scores of the false positive detection (row 3). We attribute it to larger contributions of the features from the pedestrian region, while the feature reweighting does not boost the non-pedestrian detections in a reasonable way. The convolutional features before and after reweighting are visualized in Fig. 4.6d and Fig. 4.6e. Here we show the feature channels 162-262 from the 512 channel features extracted from the VGG16 net. More detection results of multiple pedestrians are shown in Sec. 4.4.4.



(a) FRCNN (b) FR+RW (c) FR+RW (d) feature before reweight (e) feature after reweight

Figure 4.6: Three detection samples from FRCNN and our FR+RW method. After the feature reweighting guided by the depth image, our FR+RW method generates more distinguishable classification scores for true positive and false positive detections: two true positive samples from row 1 and row 2 get higher classification scores, while the score of the false positive sample from row 3 significantly drops.

4.4.3 Experiments on ROI Filtering

In default settings of FRCNN, the RPN will select at most 300 ROIs for further classification, which requires much computation. We compare the top 30 ROIs generated from default RPN and RPN with our ROI filtering network (denoted as FRCNN+filter) in Fig. 4.7. We find that with the ROI filtering network, the generated ROIs will be mostly focused on the pedestrian regions. It allows us to use fewer ROIs during the region proposal generation. In our following experiments, we choose to use the top 100 ROIs for classification.

4.4.4 Comparison with Other Methods

Here we compare our proposed method with several baseline methods:
FRCNN: Our adapted version of Faster-RCNN (see Sec. 4.4.1 for detailed settings). This method is used as our baseline RGB pedestrian detector and denoted as FRCNN.
FRCNN+depth: This is a modified version of FRCNN which also uses depth

(a) FRCNN (b) FRCNN+filter

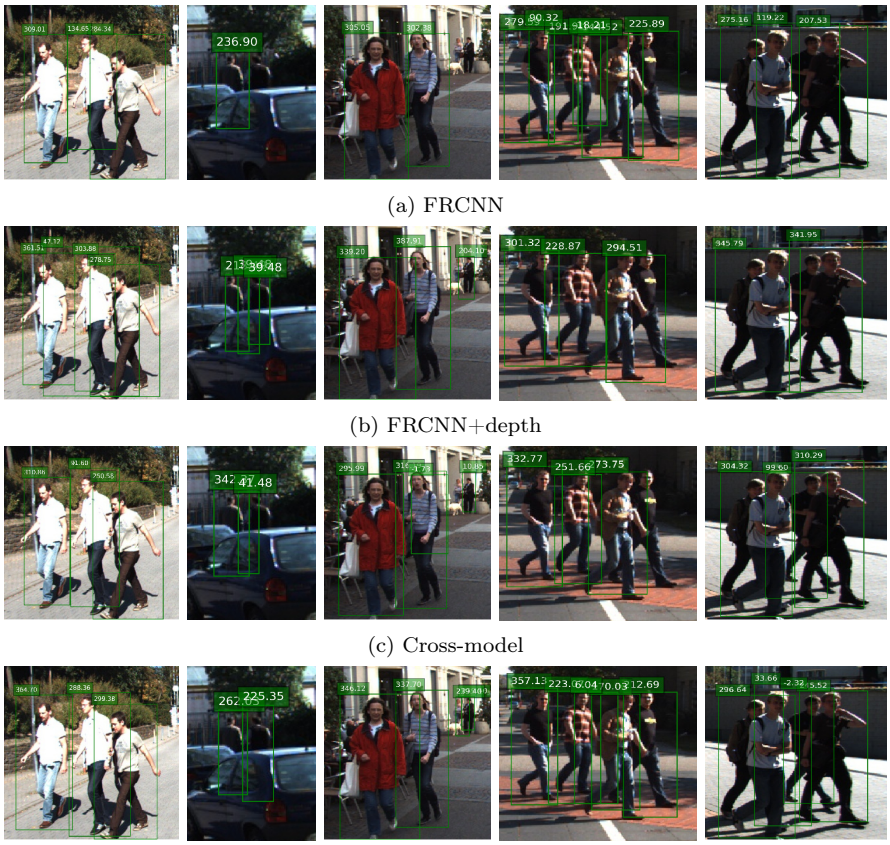
Figure 4.7: The top 30 ROIs generated from the RPN of FRCNN and our FRCNN+filter method. The ROIs generated from our method have higher quality, which are mostly located at pedestrian regions.

data. We have implemented it ourselves for the purposes of a fair comparison, i.e., a comparison to a method that also uses depth data. For the FRCNN+depth method, we train two parallel CNN branches to independently extract features from RGB and depth modalities and fuse them in later layers. We compared different fusion points for all the five convolutional layers in VGG16 net and found fusion after layer-3 performed best, which was consistent with the results in [Ophoff 19, Konig 17]. So in our experiment, we fuse the layer-3 RGB and depth features by adding a concatenation layer, which follows with a 1x1 convolution filter to connect to the fourth convolutional layer. This structure is used as our baseline RGBD pedestrian detector and denoted as FRCNN+depth.

Cross-modal: In our experiments we also compare with the RGBD object detection method [Gupta 16]. The middle layer features from the RGB modality are used as supervision for learning rich representations from depth modality. This method is used as another baseline RGBD detector and denoted as Cross-modal.

Proposed: Our RGBD pedestrian detector illustrated in Fig. 4.2, which is based on the FRCNN structure and contains two subnets: Depth Guided Feature Reweighting Net and ROI Filtering Net.

We evaluate the above methods on the KITTI dataset. The detection results are evaluated based on three levels of difficulty (Easy, Moderate and Hard), which are defined by different heights, occlusion levels and truncation percentages of the pedestrians. Since the ground truth labels of the test set are not available, in our experiments, we randomly split the training set (7481 images) into three sets and apply a 3-fold cross-validation for evaluation.



(a) FRCNN

(b) FRCNN+depth

(c) Cross-model

(d) Proposed

Figure 4.8: The comparison results of FRCNN, FRCNN+dep, Cross-model and our proposed method. The proposed method outperforms the other methods, especially under occlusion conditions.

Method	Easy	Moderate	Hard
FRCNN	72.8%	67.3%	61.8%
FRCNN+dep	74.2%	69.8%	63.7%
Cross-modal	82.6%	72.6%	65.1%
Proposed	**83.5%**	**75.8%**	**73.2%**

Table 4.1: Comparison between the four methods. Our proposed method outperforms the others, especially in Moderate and Hard cases.

Table 4.1 shows that by naively fusing RGB and depth features in CNN networks, the FRCNN+depth method achieves an obvious improvement, but it is small compared to specifically designed RGB+D methods. We attribute this to the characteristics of the KITTI dataset, because most of the images

are captured in the daytime and with sufficient luminance, which diminishes the advantage of the depth features when they are simply fused with the RGB features. Then we find the Cross-modal achieves a better improvement in all three test cases, while our proposed method further outperforms Cross-modal by 0.9%, 3.2% and 8.1% in Easy, Moderate and Hard test cases, respectively. It shows that our method works well under more difficult conditions (e.g. occlusion), which is also proved in the detection results in Fig. 4.8.

4.5 Conclusions

In this chapter, we propose a novel RGBD pedestrian detection method to better exploit the potential of depth modality. Most of the existing RGBD methods choose to independently extract features from RGB and depth images, and then fuse them for detection. These methods achieve better detection performance compared with RGB detectors, but the improvements are limited according to our experiments. We believe this is because the depth features are more powerful than the RGB features only under challenging situations, e.g., insufficient luminance or low contrast between pedestrian and background. Thus simply extracting features from depth images and fusing them with the RGB features may not be the optimal way. Inspired by this idea, we propose to take advantage of depth information to enhance the RGB features. Specifically, we produce binary maps from the depth images, which mark the real pedestrian areas of each pedestrian sample, and then reweight the RGB features by enhancing the features from pedestrian areas and suppressing features from the non-pedestrian areas. In addition, the depth image is also used to estimate the real height of each pedestrian candidate, then the candidates with unreasonable heights will be removed before classification. We report state-of-the-art performance on the KITTI dataset, which outperforms the existing RGBD object detection methods.

The research work discussed in this chapter has resulted in one peer-reviewed conference paper [Guo 19].

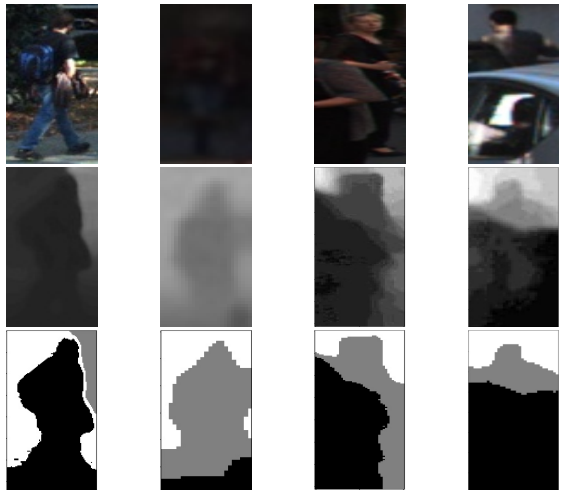
5

Weak Segmentation Supervised Deep Neural Networks for Pedestrian Detection

5.1 Introduction

Recent work has shown that the performance of pedestrian detection can be improved by utilizing semantic segmentation information [Fidler 13, Liu 18a, Zhu 20]. Comparing with the original RGB modality, exploiting semantic information helps to better distinguish pedestrians from the occluding object or the background, especially under some challenging conditions (see Fig. 5.1). Fidler et al. [Fidler 13] extract richer features from semantic segmentation masks and improve the performance of a deformable part-based model. In [Liu 18a], the authors propose to train a convolutional neural network (CNN) including two branches of RGB features and semantic segmentation features and fuse them in later stages for classification. However, during training these methods typically require the supervision from accurate pixel-level segmentation masks, which are very expensive. Furthermore, many pedestrian datasets lack pixel-level segmentation annotations, such as Caltech [Dollár 09b], EPFL [Bagautdinov 15b] and WiderPerson [Zhang 19]. And because of the high cost of labeling pixel-level segmentation annotations, some datasets (e.g. KITTI [Geiger 12]) provide far fewer segmentation annotations compared with bounding box annotations.

Depth images can be captured using several technologies, e.g., time of flight cameras or LiDAR, and provide another source for segmentation of objects, including pedestrians. It is not easy, but still possible to segment clear silhouettes of the objects in depth data [Dimitrievski 18]. Also, depth data provides information on the spatial location of objects in a scene, and specifically on occlusion. For example in Fig. 5.1c-5.1d, according to the depth information,



(a) (b) (c) (d)

Figure 5.1: Some challenging pedestrian examples are shown in the first row: (a) confusing background, (b) insufficient illumination, (c-d) occlusion. The depth images in the second row provide more clear silhouettes of the pedestrians, which can be used as a complementary cue for detection. The third row shows the result of cluster analysis applied on the depth images. This is what we call "semantic information."

we can clearly distinguish the pedestrian from the occluding object in front of it and the background behind it. A specific class of methods, called RGBD pedestrian detection [Spinello 11, Eitel 15, Zhou 17, Ophoff 19, Guo 19], already use depth information as an additional feature on top of RGB. Most of these methods [Spinello 11, Eitel 15, Zhou 17, Ophoff 19] directly fuse depth features with RGB features and show that this effectively boosts the detection performance. Some researchers take advantage of the depth images to reduce the search space of candidate regions [Camplani 16, Zhang 13], e.g., eliminating proposals located in the background region, but the spatial relationship illustrated in Fig. 5.1c-5.1d is not well exploited.

In this chapter, we propose a new method to jointly optimize the pedestrian detection and segmentation tasks. During training, the segmentation network is supervised by weak segmentation masks which are automatically generated from depth images. Experimental results show that joint training of the detection and segmentation tasks leads to an obvious performance improvement. Moreover, we observe that the output of the segmentation branch is helpful for improving the classification results, especially under some difficult conditions such as confusing backgrounds and occlusion. The fusion of the segmentation results with the detection results brings a further boost of the detection performance. We refer to our method as Weak Segmentation Supervised Network (WSSN) in the rest of this chapter.

The contributions of this chapter can be summarized as:

- We propose a novel RGBD pedestrian detection framework, which is jointly trained end-to-end with detection and segmentation tasks. The main advantage of this framework is we do not require accurate segmentation annotations for training. Instead, the segmentation network is supervised by weak segmentation masks which are automatically generated from depth images by a maximum a posterior (MAP) approach.
- We propose a fusion strategy to fuse the outputs of the segmentation and classification results. The detection performance is improved, especially under challenging conditions like insufficient illumination and occlusion.
- We outperform state-of-the-art RGBD pedestrian detectors on three RGBD pedestrian datasets.
- We obtain pedestrian segmentation results of good quality, without using accurate pixel-level segmentation annotations during training.

5.2 Related Work

In order to utilize semantic segmentation for detection, most methods [Fidler 13, Liu 18a, Xie 18] train a parallel network to automatically generate segmentation masks and fuse them with the classification network. Liu et al. [Liu 18a] integrated RGB features and semantic features together and used boosted forests for hard example mining during training. Du et al. [Du 17] used the generated masks in a post-processing manner to suppress false positive proposals in the background. Xie et al. [Xie 18] proposed a feature enrichment unit in each layer and merged the multi-scale segmentation information with the RGB features. However, these methods all require the supervision from pixel-level segmentation annotations for training, which are very time-consuming to obtain. In order to address the problem of lacking pixel-level annotations, some researchers [Brazil 17, Cao 19b] proposed to use bounding boxes as weak segmentation annotations for training. In contrast, our proposed WSSN uses depth images to automatically generate more accurate segmentation masks by an MAP approach. More accurate segmentation results also help to further improve the detection performance in our method.

Depth images are also widely used in recent pedestrian detection research [Spinello 11, Zhou 17, Eitel 15, Ophoff 19, Guo 19, Zhang 20]. Spinello et al. [Spinello 11] designed a hand-crafted feature descriptor for depth modality and combine it with the Histogram of Oriented Gradients (HOG) feature of RGB for pedestrian detection. In most CNN based methods [Eitel 15, Ophoff 19, Zhou 17], RGB and depth features were extracted independently and fused together in different stages for classification. Ophoff et al. [Ophoff 19] exhaustively explored the optimal fusion layer of RGB and depth features in CNN networks. Zhou et al. [Zhou 17] proposed to extract region proposals from depth data and fuse the classification scores from RGB

and depth data. Our proposed method in Chapter 3 generates a binary segmentation mask for each region proposal from the depth information and uses it to reweight the convolutional RGB features, while in this chapter, we have made many substantial improvements:

- We propose a new method to generate semantic segmentation masks. In order to pick the true pedestrian segment among all candidate segments, the previous method simply picks the segment in the center position. Instead, in this chapter, we adopt an MAP approach combining observations and prior knowledge. As a result, the generated segmentation mask is much more accurate. See Section 5.3.1.

- Our previous method uses the segmentation masks to reweight the learned convolutional RGB features in both training and inference. However, the quality of segmentation may directly impact the feature reweighting during inference, which limits the performance of the previous method. Instead, this chapter proposes a completely different framework named WSSN, which jointly optimizes the detection and segmentation tasks during training. The learned features will benefit from the supervision of both tasks and become more representative for the detector. It outperforms the previous method as is demonstrated in Section 5.4.

- In the previous method, as the quality of segmentation masks are often poor, they are only used to slightly reweight the RGB convolutional features. Instead, in this chapter, the jointly trained segmentation network generates pedestrian segmentation results of good quality, see Section 5.4.3.5. It also enables us to further improve the detection performance by fusing the outputs of detection and segmentation networks, see Section 5.3.3.

- More extensive experiments have been conducted in this chapter. We evaluate our proposed method in three RGBD pedestrian datasets. In addition, we compare the results to more state-of-the-art RGBD pedestrian detection methods. Experimental results in Section 5.4 show that our proposed WSSN outperforms the existing RGBD pedestrian detection methods, including the previous method.

- Lastly, we perform a comprehensive ablation study in this chapter to demonstrate the effectiveness of different components of our WSSN.

5.3 Proposed Method

Our proposed WSSN is based on the Faster R-CNN [Ren 15] architecture, which is illustrated in Fig. 5.2. The backbone CNN (VGG-16 [Simonyan 14]) extracts features from the input images, and then sends the feature map to the region proposal network (RPN) to generate the region of interests (ROIs) of

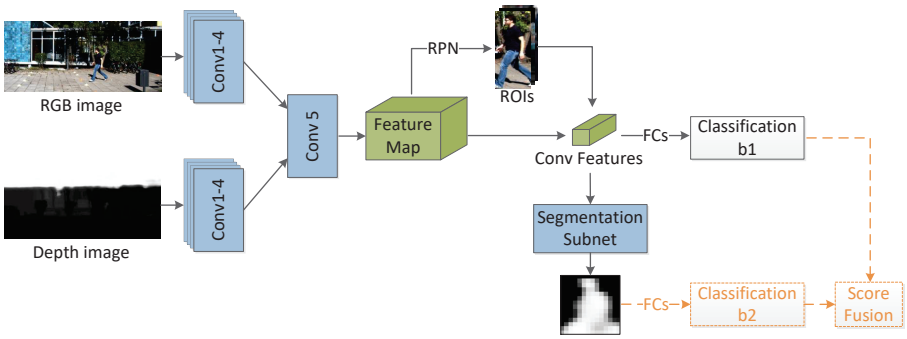


Figure 5.2: Our proposed WSSN adopts a straightforward structure, which is based on the Faster R-CNN with three main modifications: (1) An additional parallel input branch for depth images. (2) A semantic segmentation subnet to generate segmentation masks (3) A score-fusion module to fuse the segmentation and classification results. One main advantage of WSSN is that we do not need accurate pixel-level segmentation annotations for training, instead, we reuse the depth images to automatically generate weak segmentation masks, which are used to supervise the training of the segmentation subnet.

the image. The features from the ROIs are cropped from the feature map and converted into equal-length features through a ROI-Pooling layer and used for classification, generating the classification results and bounding box regression informations. Comparing with the original Faster R-CNN, our framework has the following modifications:

1. We add a parallel input branch for depth images and fuse the features at the fourth convolutional layer (see details in Section 5.4.3.2). It works together with other modifications which are introduced in Section 5.4.3.1 to significantly improve the detection performance of Faster R-CNN and provides a very challenging RGBD baseline detector.

2. We add a semantic segmentation subnet, which contains two convolutional layers with 1x1 kernel. Each layer is followed by a deconvolutional layer to increase the resolution of the generated segmentation masks (from 7x7 to 28x28). We adopt this four-layer design because a shallower network will diminish the accuracy of segmentation, while a deeper network could reduce the influence of segmentation on the shared features.

3. We add a classifier b2 which decides if the segmentation result contains a pedestrian or not. This decision is independent of the main pedestrian detector b1. Then we fuse the output of the two classifiers to reach a final decision on the presence of a pedestrian.

We train the overall network end-to-end using a combined loss criterion which is introduced in Section 5.3.2. An important contribution is how we train

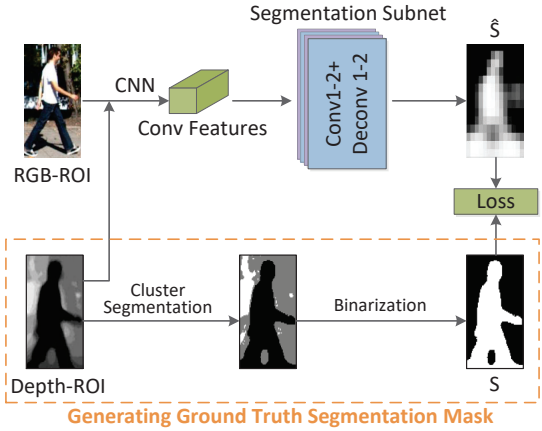


Figure 5.3: The training process of the semantic segmentation subnet. For each ROI, the depth values are clustered and binarized into a segmentation mask map S according to a Bayesian classifier, while the fused RGB and depth features are put into our segmentation subnet and generate mask $\hat{S}$. Cross-entropy loss is used as the loss function during training.

this framework. Instead of using the expensive pixel-level segmentation annotations, we use weak segmentation masks automatically generated by depth images to supervise the training of our segmentation subnet (see Fig. 5.3). The method of generating the ground truth segmentation masks will be explained in Section 5.3.1. The main idea is to exploit the proposed region to facilitate an initial segmentation by clustering, and then to analyze this segmentation, taking into account depth information, to determine which of the segments corresponds to the real pedestrian.

5.3.1 Generating Ground Truth Segmentation Masks for Training

The desired segmentation masks are binary, representing pixels belonging to a real pedestrian segment by '1' and all other pixels by '0'. The bottom stream in Fig. 5.3 shows the process to generate ground truth segmentation masks.

The first step is to cluster the depth values of each ROI into a few segments. Typically, an ROI region containing a pedestrian can be divided into two depth layers (foreground and background). However, in case of pedestrian occlusion, there can be a third depth layer with pixels covering the occluding object. Therefore we opt for a segmentation into three segments, as this can handle both occlusion and non-occlusion cases. (More experiments to explore the optimal cluster numbers are conducted in Section 5.4.3.4.) Typically a fully visible pedestrian will appear in the closest depth layer (see Fig. 5.1a), while a partially occluded pedestrian appears in the in-between depth layer (see Fig.

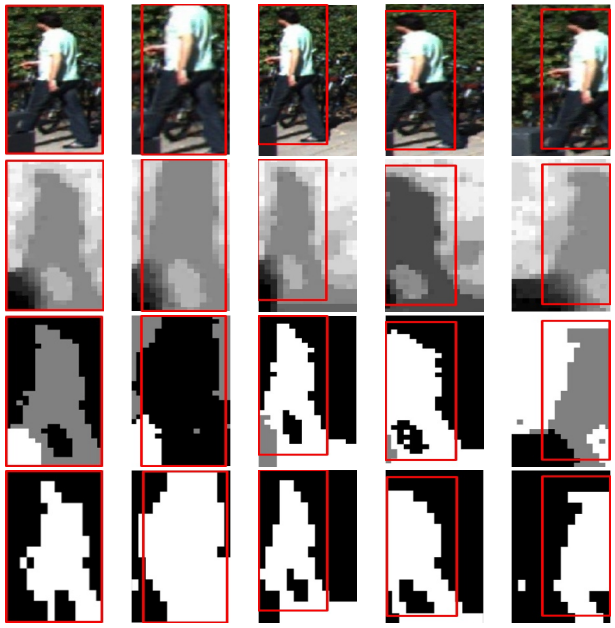


Figure 5.4: We use the IOU of each segment with the ground truth bounding box as a measure to decide which segment corresponds to pedestrian. We have five ROIs (a-e) generated from the region proposal network, while the four rows from top to bottom are RGB, depth, segmentation and mask maps of each ROI, respectively. The segmentation images in the third row illustrate that the segment (three segments in different colors) which has biggest IOU with the ground truth bounding box (in red) is more likely to be pedestrian.

5.1b, 5.1c and 5.1d). Therefore we develop a Bayesian classifier which takes the three-layer segments as input and outputs the segment index c (c=1, 2 or 3) which is estimated to correspond to the pedestrian.

Let F_c denote the event that the c-th segment covers a pedestrian, and we have the following four observations: roi denotes the ROI region generated from the RPN, bb denotes the ground truth bounding box (here we treat bb also as an observation, because we are discussing a method only used at training time), seg_c denotes the c-th segment of roi, and d_c denotes the average depth of the c-th segment. We compute an MAP estimate of the most likely F_c among F_1, F_2 and F_3, from the four observations. However, we do not use the four observations directly, but first summarize them into two variables iou_c (see Eq. 5.1) and h_c (see Eq. 5.2): iou_c represents the Intersection-over-Union (IOU) of seg_c and bb, while h_c represents the estimated real height of the pedestrian in seg_c. In the following, we will explain these two variables in detail.

As illustrated in Fig. 5.4, although most of the ROIs generated from the

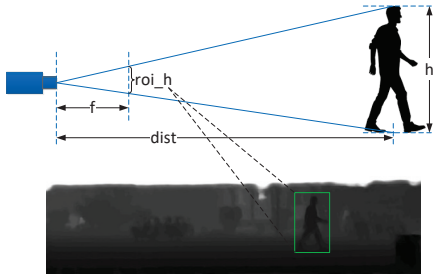


Figure 5.5: The projection relationship of a camera. We can estimate the height of the person h in the real world through the height of the ROI roi_h, the focal length of the camera f and the distance between the person and camera $dist$.

RPN do not have a good alignment with the ground truth bounding box, iou_c is an appropriate measure to check the agreement between a segment and the ground truth bounding box. It is defined as:

$$iou_c = \frac{o_c}{u_c} \tag{5.1}$$

where o_c and u_c represent the overlap and union area of seg_c with bb. If multiple ground truth bounding boxes exist within the ROI, only the one with the biggest IOU is counted. When there is no overlap of the ROI with any ground truth bounding boxes, this means this ROI is a false positive and for all c=1, 2 and 3, $iou_c = 0$. In this case the output segmentation mask map will be all-zero.

Estimating h_c and comparing it to the distribution of typical real-world person height can help us to decide if the segment seg_c contains pedestrian. To compute h_c, we assume seg_c contains a pedestrian, and we compute the height of the pedestrian according to the depth information and the camera, which is shown in Fig. 5.5 and eq. 5.2:

$$h_c = \frac{dist_c}{f} roi\_h \tag{5.2}$$

where f denotes the focal length of the camera, while roi_h denotes the height (in pixels) of the ROI. $dist_c$ represents the distance between the pedestrian and the camera in real world. As the depth value is uniformly normalized between [0,255], $dist_c$ can be computed according to the depth information of each seg_c:

$$dist_c = \frac{d_c}{255} range \tag{5.3}$$

where d_c is an observation defined earlier, while $range$ represents the maximum range of the depth map (80 meters in our experiments).

According to Bayesian theory, we compute the conditional probability of F_c

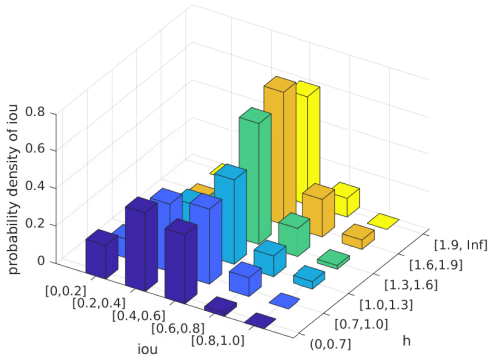


Figure 5.6: The probability density of iou based on h. It is computed from 1000 randomly selected ROIs with a real pedestrian segment, which are generated by the RPN. Then we quantize h into six intervals and iou into five intervals and get this histogram of distribution. We can find that a segment that contains a pedestrian of 1.3-1.9 meters has a higher IOU with the ground truth bounding box.

given iou_c and h_c as follows:

$$P(F_c|iou_c, h_c) = \frac{P(F_c)P(iou_c, h_c|F_c)}{P(iou_c, h_c)} \propto P(iou_c, h_c|F_c) \tag{5.4}$$

where we assume only one segment belongs to a pedestrian, while the three segments have the same prior probability of belonging to a pedestrian. Then $P(iou_c, h_c|F_c)$ is computed as:

$$P(iou_c, h_c|F_c) = P(h_c|F_c)P(iou_c|F_c, h_c) \tag{5.5}$$

In order to compute the likelihood $P(h_c|F_c)$ in Eq. 5.5, we assume h_c to follow the Gaussian distribution $N(\mu, \sigma^2)$:

$$P(h_c|F_c) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}(\frac{h_c - \mu}{\sigma})^2\right) \tag{5.6}$$

where we set the average height μ to be 1.7 and the standard deviation σ as 0.2.

In order to estimate the likelihood $P(iou_c|F_c, h_c)$ in Eq. 5.5, we analyzed 1000 randomly selected ROIs containing real pedestrians. Each ROI is clustered into three segments, and then we manually select the real pedestrian segment (assisted by Eq. 5.6) of each ROI and compute its estimated height (Eq. 5.2-5.3) and IOU (Eq. 5.1). Lastly we quantize the estimated heights and IOUs of the real pedestrian segments and get the histogram for the distribution, which is shown in Fig. 5.6.

According to Eq. 5.1-5.6, we can compute $P(F_c|iou_c, h_c)$, and we output segment c through $\hat{c} = \arg\max_c(P(F_c|iou_c, h_c))$. The pixels from segment $\hat{c}$ are marked as 1 in the mask map, while the rest of the map are marked as 0.

5.3.2 Joint Training of the Segmentation and Detection Network

The whole network is trained by minimizing the following joint loss function with five terms:

$$\begin{aligned} L =& \lambda_1 L_{RPNcls} + \lambda_2 L_{RPNbbox} + \lambda_3 L_{FRCNNcls} \\ &+ \lambda_4 L_{FRCNNbbox} + \lambda_5 L_{SEG} \end{aligned} \tag{5.7}$$

where the first four terms remain the same as the Faster R-CNN losses defined in [Ren 15], including two classification losses (L_{RPNcls}, $L_{FRCNNcls}$) and two bounding box regression losses ($L_{RPNbbox}$, $L_{FRCNNbbox}$) for RPN and classification networks. L_{SEG} is the pixel-level cross-entropy loss between the generated segmentation mask S and predicted mask $\hat{S}$:

$$L_{SEG} = \frac{1}{HW} \sum_{x,y} l(S_{x,y}, \hat{S}_{x,y}) \tag{5.8}$$

where H and W represent the size of the segmentation mask map, while l is the cross-entropy loss function. In our experiments, we set all $\lambda_i = 1$.

5.3.3 Decision-level Fusion

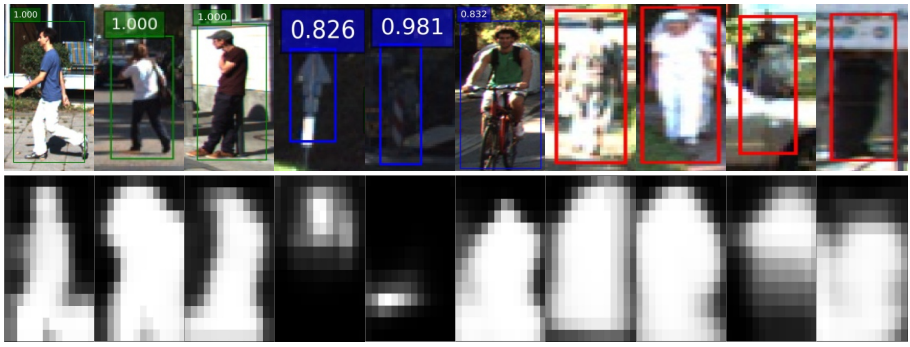
In Section 5.3.2, we propose a multi-task training framework, which enables convolutional features to learn new information from the segmentation task. However, during inference, the generated segmentation masks are not fully exploited. Thus in this section, we demonstrate the segmentation masks can be used as a complementary information with the classification results.

We have checked some errors from our detector, which is illustrated in Fig. 5.7d-5.7j. Basically most false positives come from some human-like objects or cyclists, while most false negatives are due to indistinguishable background (Fig. 5.7g), low resolution (Fig. 5.7h) and occlusions (Fig. 5.7i-5.7j). However, we find the segmentation results sometimes disagree with the classification results. For example, although the detector fails to detect the pedestrian in Fig. 5.7g-5.7j, the pedestrian is correctly segmented. Thus, if we fuse the results of segmentation and classification, some false negative examples are expected to have higher score, which will reduce the miss-rate of our detector.

We directly add two fully connected layers after the generated masks to get a new classification result, which is shown in the dashed parts of Fig. 5.2. To train this new detector, the joint loss function in Eq. 5.7 is updated by adding two new losses from the segmentation branch:

$$\begin{aligned} L =& \lambda_1 L_{RPNcls} + \lambda_2 L_{RPNbbox} + \lambda_3 L_{FRCNNcls} \\ &+ \lambda_4 L_{FRCNNbbox} + \lambda_5 L_{SEG} + \lambda_6 L_{SEGcls} + \lambda_7 L_{SEGbbox} \end{aligned} \tag{5.9}$$

where L_{SEGcls} and $L_{SEGbbox}$ indicate the classification loss and bounding box regression loss from the segmentation masks, while the other five losses remain unchanged as in Eq. 5.7. In experiments, we set λ_6 and λ_7 as 0.5.



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 5.7: Some detection results are shown in the first row, where the true positive, false positive and false negative detections are represented by green, blue and red bounding boxes, respectively. The output of our segmentation network is shown in the second row. Although some detections are missed by the detector, their segmentation masks still indicate the existence of pedestrian, which inspires us to fuse the segmentation and classification results.

Consider we have M detections from the classification branch and N detections from the segmentation branch, which are denoted as $D^i_{cls}, i = 1, 2, ..., M$, and $D^j_{seg}, j = 1, 2, ..., N$. In order to fuse the confidence scores s^i_{cls} and s^j_{seg}, we first remove the D^j_{seg} with $s^j_{seg} < 0.6$, then compute the IOU of each remaining D^j_{seg} with D^i_{cls}. The maximum IOU of D^j_{seg} with all the detections D^i_{cls} is denoted as iou_j. The steps are as follows:

1. When D^j_{seg} has maximum IOU with D^i_{cls} and $iou_j >= 0.5$, $s^i_{cls} = \max(s^i_{cls}, s^j_{seg})$. In this case, the confidence score of D^i_{cls} is updated by the higher score of the two branches.

2. When D^j_{seg} has no overlap with D^i_{cls} or the maximum $iou_j < 0.5$, we add this new detection D^j_{seg} into D^i_{cls}.

The updated D^i_{cls} are the fused results. More details are shown in Section 5.4.3.3.

5.4 Experiments

In this section, we first introduce the implementation details of our proposed WSSN, then we compare WSSN with the state-of-the-art RGBD pedestrian detection methods on three datasets. Lastly, we conduct a number of ablation experiments to verify the effectiveness of the different components of WSSN.

5.4.1 Implementation Details

Our WSSN is based on an adapted version of Faster R-CNN [Ren 15]. The VGG-16 [Simonyan 14] net pre-trained on the COCO dataset [Lin 14] is used as the backbone network, while a parallel input branch is added for fusing depth features with RGB features, as is shown in Fig. 5.2. Based on our experiments in Section 5.4.3.2, we fuse the RGB and depth features after the fourth convolutional layer (conv4_3 in VGG-16 net). We use roialign [He 17a] instead of roipool [Girshick 15] for a better alignment between the ROI and the extracted features. The whole framework is trained end-to-end with a combined loss as in Eq. 5.9.

We train our WSSN on the KITTI dataset [Geiger 12], which consists of 7481 training images and 7518 test images. As the annotations of the test set are not available, we split the training set of 60% and 40% for training and validation. Although depth images are not provided in the KITTI dataset, we generate them from the provided LiDAR cloud points data according to [Dimitrievski 18]. Since the input depth image only consists of one channel, which is different from the first layer of the VGG-16 net pretrained on RGB datasets, we decided to randomly initialize the weights of the depth branch in the first convolutional layer during training, but still reuse the pretrained weights to initialize the rest layers, as is in [Ophoff 19]. In order to improve the detection performance for small targets, we remove the fourth max-pooling layer of VGG-16 and enlarge the scale (shorter edge) of the input image from 375 to 500, both in training and inference. We train for 70k iterations, starting from a learning rate of 0.001 and reducing it by ten at 40k and 60k iterations. Other settings of WSSN are unchanged as in the original Faster R-CNN paper [Ren 15]. Training on a single GTX 1080Ti takes 18 hours.

5.4.2 Comparison with State-of-the-art Methods

5.4.2.1 Experiments on the KITTI Dataset

We compare our proposed method with other state-of-the-art RGBD detection methods [Zhou 17, Eitel 15, Zhang 20, Guo 19, Ophoff 19, Gupta 16] on the KITTI validation set. We follow the official evaluation metrics of KITTI, which evaluate the detection results based on three levels of difficulty: Easy, Moderate and Hard. The difficulty levels are defined by bounding box height, occlusion level and truncation level. For example, the Easy case evaluates the pedestrians of at least 40 pixels height, at most 0% occlusion and at most 15% truncation, while Moderate and Hard cases evaluate those smaller, more occluded and more truncated pedestrians. Average Precision (AP) over the IOU threshold of 0.5 is adopted for evaluation.

We take several measures to ensure a fair comparison. All the models are trained on the KITTI training set, and the scales of the input images are all set to 500 for both training and inference. Although Multimodal [Eitel 15], Crossmodal [Gupta 16] and YOLO-rgbd [Ophoff 19] are general RGBD object detection methods, we adapt them to detecting pedestrians as in Section 5.4.1.

Table 5.1: Comparison with state-of-the-art RGBD detection methods on KITTI validation set.

Method	Easy	Moderate	Hard
DepRoI [Zhou 17]	80.8%	70.1%	61.2%
YOLO-rgbd [Ophoff 19]	86.1%	73.3%	67.0%
Multimodal [Eitel 15]	87.8%	76.4%	71.0%
RFB [Zhang 20]	86.5%	77.8%	72.4%
Crossmodal [Gupta 16]	88.9%	78.1%	73.4%
DepReweight [Guo 19]	88.5%	78.5%	74.0%
FRCNN-rgb [Ren 15]	87.6%	75.9%	71.7%
FRCNN-dep [Ren 15]	80.5%	65.5%	59.8%
FRCNN-rgbd [Ren 15]	88.7%	77.5%	73.1%
WSSN (Proposed)	**89.5%**	**80.9%**	**75.2%**

Additionally, DepRoI [Zhou 17] and RFB [Zhang 20], which are designed for detecting the upper part of persons, are modified into detecting the full body. We use VGG-16 [Simonyan 14] pretrained on COCO dataset as backbone for all the methods (except YOLO-rgbd [Ophoff 19] which uses Darknet-19 [Redmon 17]). As our WSSN and several comparison methods [Eitel 15, Guo 19, Gupta 16] are based on Faster R-CNN, we also train three individually adapted Faster R-CNN models as in Section 5.4.1 with RGB features, depth features and RGB+depth features, which are referred to as FRCNN-rgb, FRCNN-dep and FRCNN-rgbd, respectively. The FRCNN-rgbd fuse the RGB and depth features in the same way as in our WSSN (see Section 5.4.3.2).

Table 5.1 shows the comparison results of our proposed WSSN with the state-of-the-art RGBD detection methods on the KITTI validation set. Due to the good generalization ability of the VGG-16 backbone pretrained on the COCO dataset and our modifications introduced in Section 5.4.1, our adapted version of Faster R-CNN [Ren 15] trained on RGB features is already a very powerful pedestrian detector, which achieves AP of 87.6%, 75.9% and 71.7% in Easy, Moderate and Hard cases, respectively. FRCNN-rgbd and Multimodal [Eitel 15] separately fuse the RGB and depth features at the fourth convolutional layer as well as fully connected layers, which obtain a slight performance improvement. It indicates that naively fusing depth features with RGB features at some layer only brings limited benefits. In contrast, Crossmodal [Gupta 16] utilizes the learned representations from RGB modal to supervise the learning of depth features and obtains a better performance. Our previous work DepReweight [Guo 19] proposes to reweight the RGB features according to the depth information, which achieves better performance compared with Crossmodal [Gupta 16] in Moderate and Hard cases. DepRoI [Zhou 17] does not perform well in our experiment, because the region proposals are only estimated by projecting a fixed-size person region from candidate depth pixels to the 2d image plane. Compared with the RPN from FRCNN-rgb which achieves a recall of 0.95, the recall from DepRoI is only 0.82, which significantly

Table 5.2: Comparison with state-of-the-art methods on KITTI test set. In the "Input" column, "P" and "D" indicate point clouds and depth images, respectively.

Method	Easy	Moderate	Hard	Input	Runtime
F-ConvNet [Wang 19]	83.6%	72.9%	67.1%	RGB+P	0.47s
MS-CNN [Cai 16]	85.7%	74.8%	68.9%	RGB	0.40s
FJTU-HW [Zhang 18]	87.1%	75.8%	69.8%	RGB	0.85s
FFNet [Zhao 20]	87.1%	75.8%	69.8%	RGB	1.07s
MHN [Cao 19a]	87.2%	75.9%	69.5%	RGB	0.39s
Aston-EAS [Wei 19]	86.7%	76.0%	70.0%	RGB	0.24s
ECP [Braun 19]	85.9%	76.2%	70.5%	RGB	0.25s
WSSN (Proposed)	84.9%	76.4%	71.8%	RGB+D	0.37s
RRC [Ren 17]	85.9%	76.6%	71.4%	RGB	3.60s
F-PointNet [Qi 18]	**89.8%**	**80.1%**	**75.0%**	RGB+P	0.17s[1]

degrades the detection performance. In the end, our proposed WSSN utilizes the multi-task training and decision level fusion, which obviously improves the detection performance and outperforms state-of-the-art RGBD detection methods. Visual comparisons are shown in Fig. 5.8.

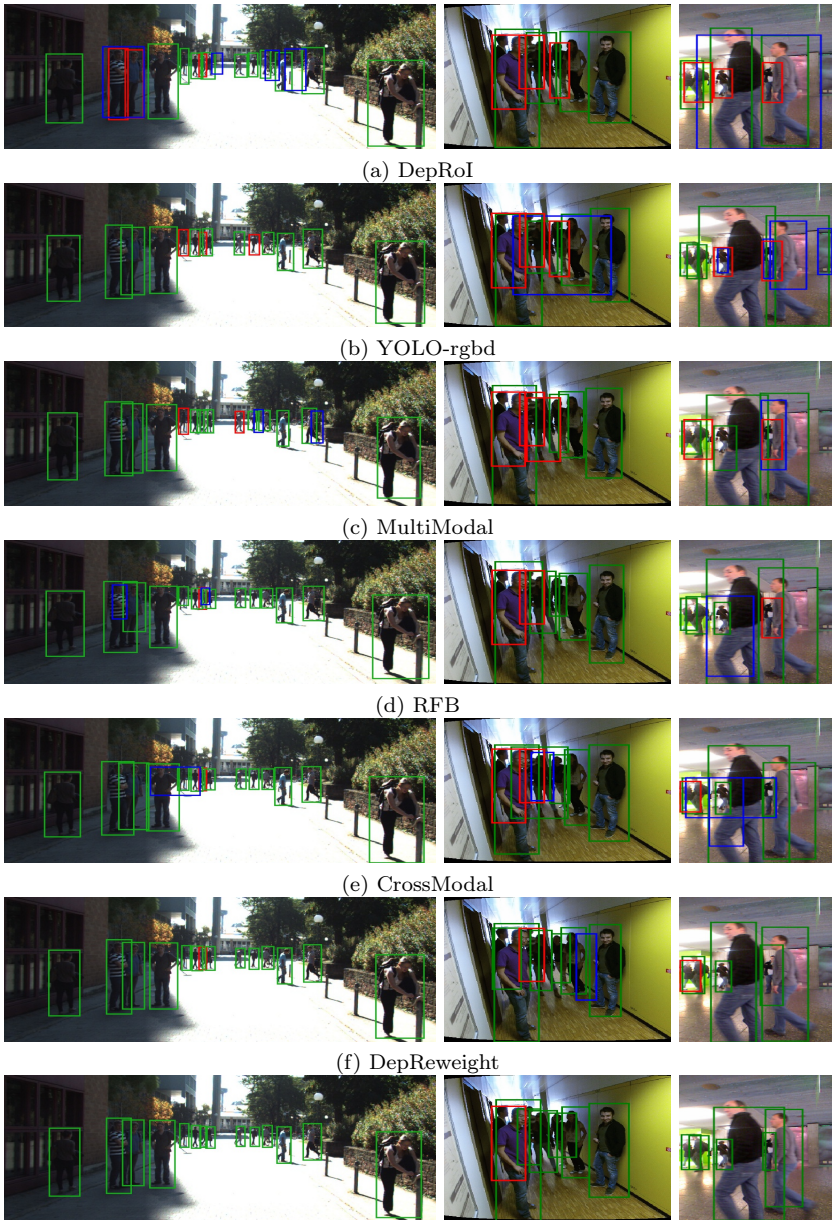
We also submitted the results of our WSSN to the challenging KITTI leaderboard [2] (shown in Table 5.2) and got a competitive performance of 84.9%, 76.4% and 71.8% in Easy, Moderate and Hard test cases, respectively. By the time we submit this paper, our method ranks 3rd among the published methods on the KITTI leaderboard (It is ranked by the performance on Moderate case, and only the methods linked to a real publication are counted). In table 5.2, although we have used depth information in our method, the inference speed of our method is still comparable with other methods.

5.4.2.2 Experiments on the EPFL Dataset and RGBD People Dataset

In order to evaluate the generalization ability of all the RGBD detection methods, we conduct experiments on another two RGBD pedestrian datasets: the EPFL dataset [Bagautdinov 15a] and the RGBD People dataset [Spinello 11]. EPFL dataset contains 5140 RGB-D image pairs, which are captured in two scenarios of lab and corridor, while RGBD People dataset contains 3399 RGB-D image pairs, which are captured by three kinect sensors in a university hall. In order to follow the KITTI evaluation metrics introduced in Section 5.4.2.1, we redefined the height of each difficulty level, because the pedestrians in these two datasets are mostly higher than those in KITTI. We put the highest 25% and shortest 25% persons in Easy and Hard cases, and leave the rest for Moderate case. We use the revised annotations [Ophoff 19] for the EPFL dataset, which are more accurate and include highly occluded persons.

[1] The performance of this method depends on a 2D object detector. The processing time of the 2D detector is not included in the runtime.

[2] http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=2d



(a) DepRoI

(b) YOLO-rgbd

(c) MultiModal

(d) RFB

(e) CrossModal

(f) DepReweight

(g) WSSN (Ours)

Figure 5.8: Detection results of our WSSN and the state-of-the-art RGBD detection methods on KITTI (left column), EPFL (middle column) and RGBD people (right column) dataset. The true positive, false positive and false negative examples are depicted with green, blue and red colors, respectively. Note that all these results are generated from RGB+D image pairs.

Table 5.3: Comparison with state-of-the-art RGBD detection methods on the EPFL dataset.

Method	Easy	Moderate	Hard
DepRoI [Zhou 17]	77.8%	58.4%	50.2%
YOLO-rgbd [Ophoff 19]	80.5%	62.3%	52.0%
Multimodal [Eitel 15]	82.8%	65.2%	52.6%
RFB [Zhang 20]	82.1%	66.8%	53.9%
Crossmodal [Gupta 16]	85.5%	68.3%	56.1%
DepReweight [Guo 19]	85.9%	69.8%	58.0%
WSSN (Proposed)	**87.9**%	**72.1**%	**60.2%**

Table 5.4: Comparison with state-of-the-art RGBD detection methods on the RGBD people dataset.

Method	Easy	Moderate	Hard
DepRoI [Zhou 17]	65.8%	56.1%	50.2%
YOLO-rgbd [Ophoff 19]	68.1%	59.8%	54.0%
Multimodal [Eitel 15]	72.3%	60.2%	56.9%
Crossmodal [Gupta 16]	74.3%	62.5%	58.3%
RFB [Zhang 20]	72.1%	63.8%	57.6%
DepReweight [Guo 19]	73.5%	64.1%	60.2%
WSSN (Proposed)	**75.6**%	**66.3**%	**62.5%**

The comparison results are shown in Table 5.3 and Table 5.4. Our WSSN also outperforms other RGBD detection methods, which achieves AP of 72.1% and 67.3% respectively on the moderate case of the EPFL and RGBD people dataset. Visual comparisons are shown in Fig. 5.8.

5.4.3 Ablation Studies

We further evaluate several variants of WSSN to verify the effectiveness of each individual component. The experiments are performed on the KITTI evaluation set, using the same settings as described in Section 5.4.2.1.

5.4.3.1 Adaptations of Faster R-CNN

As is mentioned in Section 5.4.1, we modify the default Faster R-CNN [Ren 15] to adapt it to pedestrian detection. Table 5.5 shows the evaluation results of these modifications. We use 4 aspect ratios of [1, 1.5, 2, 2.5] and 5 scales of [1, 2, 4, 8, 16] for the RPN anchors (referred to as 'More proper anchors' in Table 5.5), which better fit the shape of human and allow more smaller targets. Then we remove the fourth max-pooling layer of VGG-16 (referred to as 'Finer feature stride' in Table 5.5), which reduces the feature stride from 16 to 8. It helps the network to generate more accurate locations of targets, especially for the smaller ones. The input image scale is also increased from 375 to 500,

Table 5.5: Evaluations of the modifications of Faster R-CNN on the KITTI validation set

Modifications						
Default settings [Ren 15]	√	√	√	√	√	√
More proper anchors		√	√	√	√	√
Finer feature stride			√	√	√	√
Input up-sampling				√	√	√
Roialign [He 17a]					√	√
COCO pretrained						√
Easy	78.1%	79.8%	81.6%	83.5%	83.8%	**87.6%**
Moderate	62.8%	63.1%	67.9%	70.4%	71.1%	**75.9%**
Hard	54.7%	56.3%	60.4%	65.1%	66.9%	**71.7%**
Runtime	0.095s	0.098s	0.113s	0.160s	0.164s	0.165s

which further improves the detection results of small pedestrians. In addition, we adopt roialign [He 17a] instead of roipool [Girshick 15], which provides a better alignment of the extracted features with the corresponding ROI. Lastly, we use the weights pretrained on the COCO dataset, instead of on ImageNet, to initialize the VGG-16 during training, which significantly improves the detection performance of our finetuned model. With an acceptable sacrifice of the inference speed (processing time increases from 0.095s to 0.165s for each frame), our adapted Faster R-CNN achieves a significant improvement of 9.5%, 13.1% and 17.0% on Easy, Moderate and Hard cases on the KITTI validation set. This adapted version of Faster R-CNN is adopted as the backbone of our WSSN.

5.4.3.2 Fusing RGB+D

In order to explore the most proper fusion point for the RGB and depth features, we separately train five Faster R-CNN models with two input branches, where the features of the two branches are fused after the first (conv1_2, referred to as 'Conv1' in Table 5.6) to fifth (conv5_3) convolutional layer of VGG-16. As is mentioned in Section 5.4.1, we randomly initialize the weights of the first convolutional layer in the depth branch, while reuse the pretrained weights to initialize the other convolutional layers before the fusion layer.

As is shown in Table 5.6, fusion after the fourth convolutional layer (conv4_3) brings the best detection performance, which improves the AP by 1.6% in the Moderate case compared with the model trained only with RGB input. This conclusion is different from another work [Konig 17] which fuse the RGB and infrared images after the third (conv3_3) convolutional layer of VGG-16. We attribute it to the different characteristics of depth and infrared modality, while our COCO pretrained weights may also support the feature fusion of higher semantic levels better. We also notice this result conforms with the conclusion in [Ophoff 19], where the authors claim fusing towards the mid to late layers performs better. Therefore, we adopt this fourth layer fusion

Table 5.6: Performance (AP) of Faster R-CNN models trained with different RGB+D fusion layer.

Fusion Layer	Easy	Moderate	Hard	Runtime
Only RGB	87.6%	75.9%	71.7%	0.165s
Conv1	87.9%	76.1%	71.6%	0.173s
Conv2	87.9%	76.3%	71.4%	0.192s
Conv3	88.0%	77.0%	72.5%	0.226s
Conv4	**88.7%**	**77.5%**	**73.1%**	0.247s
Conv5	88.3%	76.6%	72.1%	0.260s

Table 5.7: Effectiveness of our joint training and decision level fusion components.

Models	Easy	Moderate	Hard	Runtime
FRCNN-rgbd	88.7%	77.5%	73.1%	0.247s
+Joint-training	**89.8%**	79.1%	74.6%	0.338s
+Decision level fusion	89.5%	**80.9%**	**75.2%**	0.344s

strategy in our WSSN.

5.4.3.3 Joint Training and Decision-level Fusion

In order to demonstrate the effectiveness of our joint training and decision level fusion methods, we train an RGBD Faster R-CNN model as described in Section 5.4.3.1 and Section 5.4.3.2, which is referred to as 'FRCNN-rgbd'. Then we add the segmentation subnet and jointly train the whole network with the loss function in Eq. 5.7, which is referred to as '+Joint-training'. Finally we train another model with the loss function in Eq. 5.9 and add the decision level fusion mechanism to the jointly trained model during inference, which is referred to as '+Decision level fusion'. The three models are compared in Table 5.7.

Our multi-task training method improves the AP of all three cases from 88.7%, 77.5% and 73.1% to 89.8%, 79.1% and 74.6%, respectively. It demonstrates that by simultaneously optimizing segmentation and classification tasks, the shared features become more sophisticated and helpful for the pedestrian detector. Although score fusion does not improve the performance in the Easy test case (We believe this is because the sole classification results are already very accurate for easy examples), it works for both Moderate and Hard cases, which brings a further improvement of 1.8% and 0.6%, respectively. The inference time increases from 0.247s to 0.344s, but our method still achieves a good accuracy-speed trade-off comparing with other state-of-the-art methods (see Table 5.2).

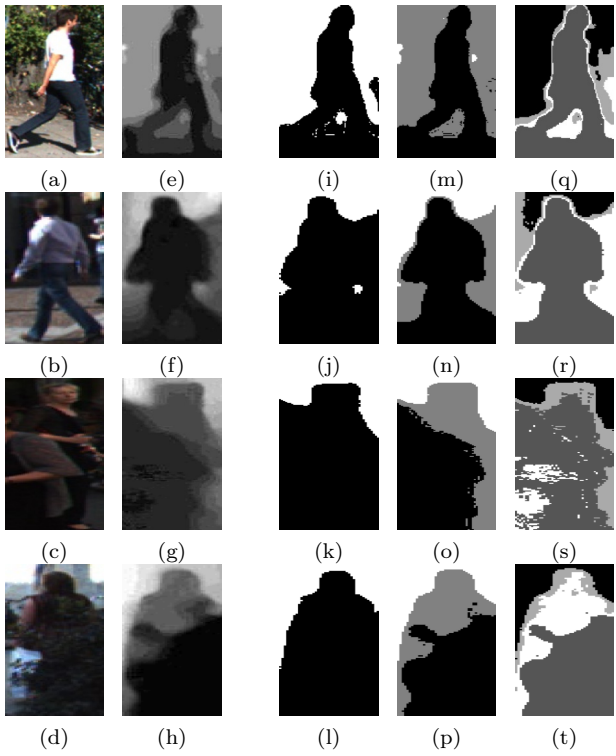


Figure 5.9: Cluster ROIs into different number of segments. Column 1-2: RGB and depth information of ROIs; Column 3-5: segmentation results with two, three and four cluster centers.

5.4.3.4 Number of Clusters for Segmentation

As was introduced in Section 5.3.1, the segmentation masks used for training are generated by two steps: Firstly the depth values of each ROI are clustered into a few segments, then our Bayesian classifier will determine which segment corresponds to real pedestrian and generate the binary mask. Here we explore the optimal number of clusters that is used in the first step.

Fig. 5.9 illustrates the segmentation results for different number of clusters. We find that adopting three clusters is the most proper, because the three-means clustering works for both cases with and without occlusion. In cases without occlusion (Fig. 5.9m and Fig. 5.9n), since the pedestrian area has a smaller intra-cluster depth variance than the background, the depth pixels of the pedestrian area are more likely to be clustered into one segment, leaving the background area to be clustered into two segments. In cases with occlusion (Fig. 5.9o and Fig. 5.9p), having three clusters helps to generate a more accurate segmentation, which separates the pedestrian from both background and the occluding object. Increasing the number of cluster centers to four may cause

Table 5.8: Comparison of different cluster numbers. 'Accuracy' represents proportion of ROIs which are properly clustered and correctly classified by our Bayesian classifier. The 'Accuracy' will influence the quality of the generated segmentation annotations used for training, and therefore influence the detection performance of our WSSN.

Number of clusters	Accuracy	Easy	Moderate	Hard
two	97.4%	90.0%	79.7%	74.4%
three	94.2%	89.5%	80.9%	75.2%
four	83.7%	88.2%	78.4%	73.5%

an over-segmentation of the pedestrian region, as is shown in Fig. 5.9s and Fig. 5.9t.

We evaluate the influence of the imposed number of clusters on the accuracy of the output of our Bayesian classifier, as is shown in Table 5.8. An accurate classification means the ROI is correctly clustered (neither mixing the pedestrian with background as in Fig. 5.9j, nor overly segmenting the pedestrian region as in Fig. 5.9s and Fig. 5.9t) and the real pedestrian segment of this ROI is correctly selected by our Bayesian classifier. We randomly generate 1000 ROIs containing real pedestrian from the output of RPN during training, then we manually select the real pedestrian segments for all the ROIs and compare with the output results of our Bayesian classifier to get the accuracy. Table 5.8 shows that when there are two clusters, 97.4% of ROIs are properly clustered and correctly classified by our Bayesian classifier, while the accuracy slightly drops by 3.2% for the three clusters. Because of the lower quality of four-means clustering and more candidate segments to be selected, the accuracy obviously decreases by 10.5% when there are four clusters. The accuracy of the classification results will directly influence the quality of the generated segmentation mask used for training and therefore influence the performance of the jointly trained detector.

We also evaluate the detection performance of the models trained with different number of clusters in Table 5.8. Because of the low accuracy, the model trained with four clusters performs worst. Although the model trained with two clusters achieves better performance in the Easy case, the model trained with three clusters performs best in Moderate and Hard cases. It proves that using three clusters help the detector to better handle occlusions. In conclusion, we choose to use three clusters during the training of our WSSN.

5.4.3.5 Pedestrian Segmentation Results

Although the quality of segmentation is not our main focus, we show some segmentation results from our WSSN to demonstrate the effectiveness of our segmentation subnet.

In Fig. 5.10, we compare the segmentation results of our WSSN with two methods: 1) A box-level segmentation supervised method [Cao 19b], which uses bounding boxes as the ground truth segmentation masks during train-

(a) Test images from KITTI

(b) Segmentation results from box-level segmentation supervision

(c) Segmentation results from WSSN

(d) Segmentation results from Mask R-CNN

Figure 5.10: Comparison of pedestrian segmentation results (under same detection results) from WSSN and other two methods, which are supervised by box-level segmentations and accurate manually annotated segmentations, respectively. Our WSSN significantly outperforms the box-level segmentation supervised method, and obtains comparable segmentation results with the state-of-the-art segmentation method Mask R-CNN.

ing. 2) A state-of-the-art segmentation method Mask R-CNN [He 17a], which use accurate segmentation annotations during training. Our WSSN obviously outperforms the box-level segmentation supervised method, and is comparable with the mature segmentation method Mask R-CNN. Comparing with Mask R-CNN, our WSSN mostly fails at segmenting the legs of pedestrians. This is because the segmentation masks which we use for training are automatically generated from depth images, while the legs of pedestrian and the ground have very similar depth values, and therefore difficult to be separated, as is shown in Fig. 5.1a and Fig. 5.9n. Considering we do not use any manually annotated segmentation masks during training, these segmentation results are encouraging.

5.5 Conclusions

In this chapter, we proposed a novel RGBD pedestrian detection framework, which enables the joint optimization of pedestrian detection and segmentation tasks. We achieve not only state-of-the-art RGBD pedestrian detection performance, but also pedestrian segmentation results of good quality, without using accurate pixel-level segmentation annotations for training. Instead, we use weak segmentation masks automatically generated from depth images to supervise the joint training of the network. Moreover, the generated segmentation map is also fused with the classification results to achieves further performance improvement. This is inspired by our observation that the segmentation map can often indicate the existence of a pedestrian, which can be used to complement the classifier. The proposed method outperforms existing RGBD pedestrian detection methods on three datasets. We have also conducted extended ablation studies to demonstrate the effectiveness of different modules of the proposed method.

Since our proposed method adopts a very straightforward structure, it still has potential for improvements by utilizing more sophisticated networks or other techniques (e.g. feature pyramid). It is however beyond the scope of this chapter and we will leave it for our further work. Our method can also be easily integrated into other state-of-the-art RGB-based methods to achieve a further improvement.

The work has resulted in one peer-reviewed journal paper [Guo 21a].

6

FLOPs-efficient Filter Pruning via Transfer Scale for Neural Network Acceleration

6.1 Introduction

In the previous chapters, we proposed several novel methods for pedestrian detection. Most of these methods are based on deep neural networks, which require very high computation and storage costs, and are therefore difficult to be deployed on resource-constrained devices without specialized optimization of the network structures. Many different methods like low-rank factorization [Sainath 13, Li 18], knowledge distillation [Bucilua 06, Zhu 18] and network pruning [Han 15a, Li 16] have been used for model compression and acceleration. Among them, network pruning has gained extensive attentions due to its good compatibility and performance [Choudhary 20]. In this chapter, we focus on network pruning to compress the large-scale deep neural networks.

Network pruning can be roughly divided into two categories: unstructured pruning and structured pruning. Unstructured pruning is also known as weight pruning [Han 15b, Louizos 17], which aims to remove redundant connections of the neural network. Although weight pruning is the most fine-grained pruning method, it always results in sparse weight matrices in the network, which requires more specialized hardware or libraries to obtain compression or speedup in real applications [Choudhary 20]. In contrast, structured pruning methods [Li 16, Liu 17, He 19, Huang 18] prune at the level of filters or even layers. They preserve the regular structure of the neural network, and are therefore more preferred for model compression and acceleration [Liu 18b].

The basic idea of filter pruning is to select and remove the least important filters of the neural network, which should not lose too much accuracy. In order

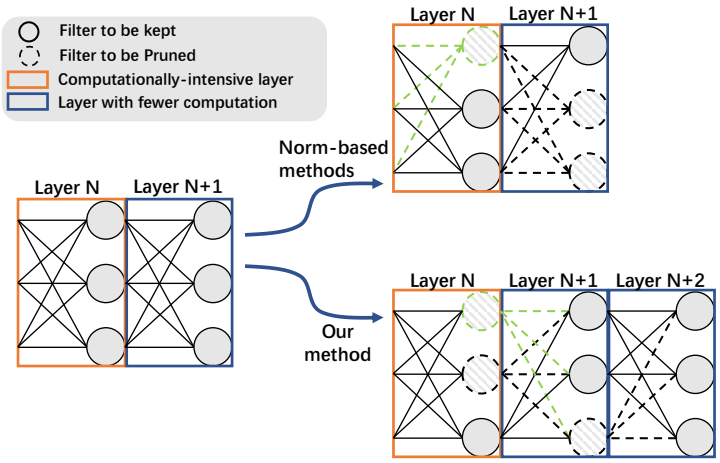
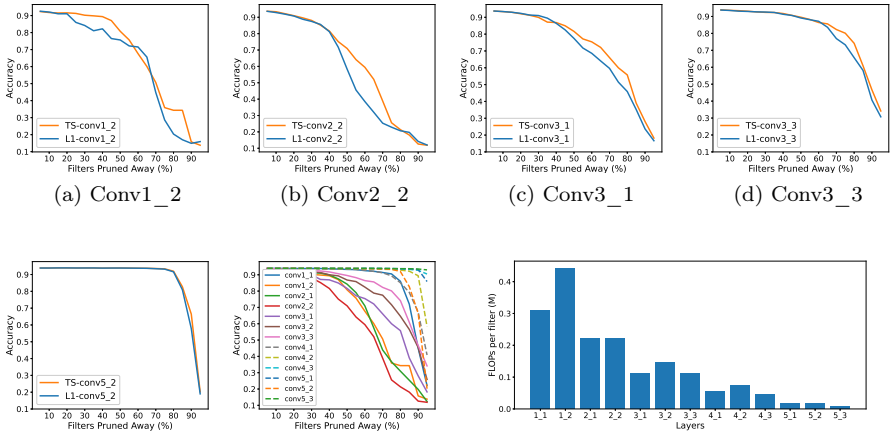


Figure 6.1: An illustration of our proposed method. Suppose we are going to prune three filters from Layer N and Layer N+1, while Layer N is more computationally-intensive. Our method has two main characteristics: 1) We measure the importance of filters according to the filters in the next layer, because filters with stronger connections with the next layer will have larger influences in the network. 2) We propose a FLOPs-efficient group Lasso approach to suppress more filters from computationally-intensive layers. This is inspired by our observation that filters from computationally-intensive layers are more sensitive to pruning, which makes it more difficult to reach higher FLOPs compression without sacrificing accuracy. Here the computationally-intensive means larger computation cost of each filter. Our method obtains a significantly better FLOPs compression ratio.

to measure the importance of filters, different criteria have been proposed. Li et al. [Li 16] and He et al. [He 18] utilize $L1$-norm and $L2$-norm of the filter weights as the pruning criteria, respectively. Hu et al. [Hu 16] explore the output of filters on a large dataset and prune those with high percentage of zero outputs. Yu et al. [Yu 18] propose to minimize the reconstruction error of the last layer to determine which filters to keep. Among them, the norm-based metrics [Li 16, He 18] are simplest and data-independent, but there are two main problems: 1) As the existence of activation functions, the activations from large norm filters can still fall into zero areas. 2) The impact of a filter in the network flow is not considered.

To address the above mentioned problems, we propose a novel data-independent criterion to measure the importance of filters. It is as simple as the norm-based metrics but is more effective. Instead of analyzing each filter by its own weights, we utilize the filters in the next layer to assess the filters in the current layer, which is illustrated in Fig. 6.1. The idea is that if one filter has weak connections with the filters in the next layer, the transfer scale (TS) of its feature map will be small, and therefore removing the filter will not influ-



(a) Conv1_2 (b) Conv2_2 (c) Conv3_1 (d) Conv3_3

(e) Conv5_2 (f) Sensitivity based on TS (g) FLOPs per filter in different layers

Figure 6.2: (a-e) Comparison of L1 and TS on different layers of VGG16 trained on CIFAR-10. It is evaluated by pruning the top x% filters according to L1 and TS, respectively. Then accuracies of the pruned models are reported. TS obviously outperforms L1 as a filter importance criterion. (f) Sensitivity of different layers based on TS. (g) FLOPs per filter in different layers. It is observed that filters with more FLOPs are also more sensitive to pruning. Thus, although pruning these filters leads to a better compression performance, it also brings a larger accuracy drop.

ence the whole network. We refer to this proposed criterion as TS, and more details of TS are explained in section 6.3.2. Fig. 6.2a-6.2e demonstrates that TS is a better pruning metric compared with the conventional $L1$-norm based metric [Li 16]. For similar pruning ratios, TS based pruning has an obvious smaller influence on the model accuracy.

An important observation from Fig. 6.2f and Fig. 6.2g is that filters from computationally-intensive layers are also more sensitive to pruning. Here computationally-intensive refers to more Floating-point Operations (FLOPs) required for each filter (see Eq. 6.8). This observation indicates a dilemma for filter pruning: When pruning from computationally-intensive layers, it leads to better FLOPs compression ratio, but it is also more difficult to preserve the accuracy of the model. Existing methods [Li 16,Liu 17] normally choose to prune fewer filters from the sensitive layers, which is not efficient in FLOPs compression. Therefore, we propose a FLOPs-efficient group Lasso approach for TS (FETS), which guides the network to use fewer filters in computationally-intensive layers during training. It helps the network to reduce the dependence on the computationally-intensive filters, which obviously facilitate the following pruning process.

Our main contributions can be summarized as:

- We demonstrate that the importance of filters can be measured by the weights of filters in the next layer. Hence we propose a simple but effective filter pruning criterion TS, which outperforms previous norm-based metrics, and is data-independent.
- We propose a FLOPs-efficient group Lasso approach for TS, which guides the network to use fewer computationally-intensive filters and leads to a better FLOPs compression ratio.
- We report state-of-the-art pruning results on several modern neural network structures on two datasets.

6.2 Related Work

Weight Pruning. Many previous works [LeCun 90, Carreira-Perpinan 18, Han 15b, Louizos 17, Ye 19] focus on pruning the individual weights of filters. Han et al. [Han 15b] propose to iteratively prune filter weights with small magnitudes, which is further incorporated with other techniques like quantization [Hubara 17] to obtain higher compression performance [Han 15a]. Louizos et al. [Louizos 17] introduce L0-norm regularization to obtain a sparse network structure. Ye et al. [Ye 19] randomly prune weights with small gradients during the network training. As we mentioned in Section 6.1, weight pruning leads to irregular sparse structures, which is difficult to leverage the off-the-shelf deep learning libraries (e.g. Basic Linear Algebra Subprograms).

Filter Pruning. Filter pruning [Li 16, He 18, He 19, Hu 16, Wang 18, Luo 20] is more preferred because of its better compatibility. Li et al. [Li 16] propose a heuristic method to prune by the $L1$-norm of filter weights, while He et al. [He 18] use $L2$-norm as the criterion and prune the selected filters in a soft manner. He et al. [He 19] rethink the *smaller-norm-less-important* rule and propose to prune the relatively redundant filters by geometric median. These methods measure the importance of a filter by analyzing its own weights, which is more efficient and does not depend on input data. However, focusing on the filter's own weights may cause neglect of the filter's influence in the whole network. Moreover, Hu et al. [Hu 16] use the average percentage of zero output to recognize unactivated filters in the network. Wang et al. [Wang 18] propose to find redundant feature maps by subspace clustering. Lin et al. [Lin 20] propose to prune filters with low-rank feature maps. It is worth noting that ThiNet [Luo 17] has a similar idea with us to measure the filters of current layer by the next layer, but they measure by minimizing the next layer's feature reconstruction error, which is still dependent on input data. In contrast, we claim that the weights of filters from the next layer are already powerful indicators of importance, which can be used to estimate the influence of filters in the current layer. In addition, it is easy to cooperate with other sparse techniques to achieve a higher compression ratio, which is introduced next.

Group Sparsity. Some methods [Wen 16, He 17b, Huang 18, Liu 17] introduce group sparsity [Yuan 06] into training to enforce the whole filter weights

to be zero, which smooths the following pruning process. He et al. [He 17b] propose to use Lasso regression based channel selection and least square reconstruction to iteratively prune each layer. Liu et al. [Liu 17] utilize a sparse scaling factor from the batch normalization layer to slim the network structure more efficiently. However, we observe that it is more difficult to introduce sparsity into computationally-intensive layers (see Section 6.4.3.2). This is similar to our observation in Fig. 6.2 that filters from computationally-intensive layers are also more sensitive to pruning, which confirms the necessity to take the computation intensity of different layers into account. Therefore, we propose a FLOPs-efficient group Lasso on TS to enforce the network to use fewer filters from the computationally intensive layers, which leads to a better FLOPs compression.

6.3 Proposed Method

6.3.1 Notations

Here we introduce the notations we used in this chapter. Given a CNN model with N convolutional layers, we use L^i to denote the i-th layer. The filters from L^i can be represented as a set of 3-D filters $W^i = \{w_1^{(i)}, w_2^{(i)}, ..., w_{n_i}^{(i)}\}$, where the j-th filter is $w_j^{(i)} \in \mathbb{R}^{n_{i-1} \times c_i \times c_i}$. n_{i-1} and n_i represent the input and output channels of L^i, respectively. In particular, n_0 represents the number of channels for the input data. c_i denotes the kernel size. The output feature maps from L^i can be represented as $M^i = \{m_1^{(i)}, m_2^{(i)}, ..., m_{n_i}^{(i)}\}$, where the j-th feature map is $m_j^{(i)} \in \mathbb{R}^{u_i \times v_i}$. u_i and v_i denote the height and width of the feature maps in L^i, respectively.

6.3.2 Transfer Scale

Our basic idea is to measure the influence of filter $w_j^{(i)}$ according to the contribution of its feature map $m_j^{(i)}$ in the next layer L^{i+1}. We evaluate the contribution of $m_j^{(i)}$ according to the average output generated from $m_j^{(i)}$ for all the filters $w_j^{(i+1)}$ in L^{i+1}, which is denoted as $C(m_j^{(i)})$:

$$C(m_j^{(i)}) = \frac{1}{n_{i+1}} \sum_{k=1}^{n_{i+1}} m_j^{(i)} * w_{k,j,\cdot,\cdot}^{(i+1)}, \tag{6.1}$$

where $*$ represents the convolution operation in CNNs, and $w_{k,j,\cdot,\cdot}^{(i+1)}$ represents the weights from the j-th channel of the k-th filter in layer L^{i+1}. It can be rewritten as:

$$C(m_j^{(i)}) = m_j^{(i)} * \frac{1}{n_{i+1}} \sum_{k=1}^{n_{i+1}} w_{k,j,\cdot,\cdot}^{(i+1)}. \tag{6.2}$$

Since $C(m_j^{(i)})$ is only related to the j-th input channel of all the filters in L^{i+1}, we propose to measure the importance of $w_j^{(i)}$ according to the mean absolute weights from the corresponding channels of W^{i+1}, which is referred to as TS and defined as:

$$TS(w_j^{(i)}) = \frac{1}{n_{i+1}} \sum_{k=1}^{n_{i+1}} \sum_{p=1}^{c_{i+1}} \sum_{q=1}^{c_{i+1}} \left| w_{k,j,p,q}^{(i+1)} \right| . \tag{6.3}$$

According to Eq. 6.1 and Eq. 6.3, we have:

$$\lim_{TS(w_j^{(i)}) \to 0} C(M^i) = C(M^i - \{m_j^{(i)}\}) , \tag{6.4}$$

where $C(M^i) = \sum_{j=1}^{n_i} C(m_j^{(i)})$, and $C(M^i - \{m_j^i\})$ represents the output of L^{i+1} generated from all features maps M^i except $m_j^{(i)}$. Eq. 6.4 indicates that when $TS(w_j^{(i)}) = 0$, the filter $w_j^{(i)}$ has no influence in the network and therefore can be removed safely. As is shown in Fig. 6.2a-6.2e, TS is a better criterion for measuring the importance of filters. In Section 6.4.3.1, more experiments have been conducted to verify the effectiveness of TS.

In order to prune more filters without affecting the model accuracy too much, we need more filters to have a TS value close to zero. Therefore we introduce a group sparsity regulation for TS which will be explained next.

6.3.3 FLOPs-efficient Group Lasso

In order to facilitate the pruning procedure, we impose a group Lasso sparsity regulation on TS during training, which will push filters to have weaker connections to all the filters in the next layer. One important point is that we will apply a more intense regulation to the filters containing more FLOPs. Because filters from the computationally-intensive layers are more sensitive to pruning, and when a normal group regulation [Wen 16, Liu 17] is adopted, more sparsity will appear on the less computationally-intensive layers (see Section 6.4.3.2). This leads to a smaller FLOPs compression because pruning filters in computationally-intensive layers can effectively remove more computation from the network. Therefore, we propose a FLOPs-efficient group Lasso approach. The proposed optimization target is as follows:

$$\mathcal{L} = \mathcal{L}_D(f(x, W), y) + \lambda_r R(W) + \lambda_g \sum_{i=1}^{N} \beta_g^{(i)} R_g(W^{i+1}) , \tag{6.5}$$

where the total loss $\mathcal{L}$ is composed of three parts: the data loss $\mathcal{L}_D(\cdot)$ and two regularization terms $R(\cdot)$ and $R_g(\cdot)$. λ_r and λ_g are used for balancing the regulation terms with the data loss. x and y denote the input data and ground truth labels, respectively. $f(x, W)$ represents the output of the network. $R(\cdot)$ represents the L_2-norm which is applied to every filter weight from W. The

group Lasso regulation term is $R_g(\cdot)$, which is applied to specific weight groups. According to Eq. 6.3, we group the weights of filters as follows:

$$R_g(W^i) = \sum_{k=1}^{n_{i-1}} \sqrt{\sum_{j=1}^{n_i} \sum_{p=1}^{c_i} \sum_{q=1}^{c_i} (w_{k,j,p,q}^{(i)})^2} \,. \tag{6.6}$$

where the weights from the same channels of the filters in each layer are put into one group. During training, all the weights from some groups are pushed to smaller values at the same time, which guides W^i to use information from fewer filters from W^{i-1}. Then the unused filters can be safely removed by pruning.

We determine $\beta_g^{(i)}$ according to the amount of computation (FLOPs) required for each filter in W^i. The FLOPs of the whole model B can be computed as the summation of the FLOPs in each layer:

$$B = \sum_{i=1}^{N} n_{i-1} c_i c_i u_i v_i n_i \,, \tag{6.7}$$

but computing the FLOPs of one filter is a little more complicated, because removing one filter from W^i will reduce the FLOPs in both L^i and L^{i+1}. Therefore, we define the FLOPs of one filter as the reduced FLOPs in the model when removing it, which is computed as follows:

$$B(W^i) = n_{i-1} c_i c_i u_i v_i + c_{i+1} c_{i+1} u_{i+1} v_{i+1} n_{i+1} \,. \tag{6.8}$$

Fig. 6.2g shows the computation results of $B(W^i)$ in the VGG-16 [Simonyan 14] network. Removing filters from the shallower layers will have better efficiency in reducing FLOPs, because the size of feature maps (u_i and v_i) are significantly smaller in deeper layers. So we obtain β_g^i by normalizing $B(W^i)$, which forces the network to use fewer filters from these computationally-intensive layers:

$$\beta_g^{(i)} = \sqrt{\frac{B(W^i)}{\min(B(W^1), B(W^2), ..., B(W^N))}} \,. \tag{6.9}$$

Here we use the square root to limit the range of β_g^i, or the training will have difficulty in converging because of a big imbalance between different layers.

6.4 Experiments

In this section, we evaluate our proposed FETS with several popular network structures (VGG [Simonyan 14], ResNet [He 16] and DenseNet [Huang 17]), on both small and large image classification datasets: CIFAR-10 [Krizhevsky 09] and ImageNet [Russakovsky 15]. The CIFAR-10 dataset consists of $50k$ training images and $10k$ test images of 10 classes, while ImageNet has a much larger

scale, which contains 1.28 million training images and $50k$ validation images of 1000 classes.

6.4.1 Implementation Details

We implemented our proposed method with PyTorch [Paszke 17]. We use Stochastic Gradient Descent (SGD) as our optimizer for training, with a batch size of 64 and an initial learning rate of 0.1. We set weight decay (λ_r in Eq. 6.5) and momentum as 0.0005 and 0.9, respectively. On CIFAR-10, training takes 160 epochs, with a learning rate reduction by 0.1 at 50% and 75% of the total training epochs. On ImageNet, training takes 60 epochs, the learning rate is reduced by 0.1 at the 20th and 40th epochs. For VGG-16 we choose $\lambda_g = 10^{-5}$, and for ResNet and DenseNet $\lambda_g = 3 \times 10^{-5}$.

The networks are firstly trained from scratch with our FLOPs-efficient group Lasso approach (see Section 6.3.3), then we prune the trained models based on TS (see Eq. 6.3). As for VGG-16, all the filters will be globally ranked according to TS, then we set a target pruning ratio to determine the percentage of filters to be pruned, which will at the same time generate a new pruned network structure. While for ResNet and DenseNet, due to the existence of short-cut connections, we prune all the layers with the same percentage to keep their structures unchanged. We copy the weights of the corresponding filters from the trained model to the pruned model. Then the pruned model is finetuned to recover its ability.

From our observations, finetuning the pruned model with the same epochs as training the large model is not enough to reach the convergence. Therefore, we follow Liu et al. [Liu 18b] and finetune the small pruned model according to its FLOPs compression ratio. If the pruned model achieves a FLOPs compression of k times, we finetune the pruned model for $\sqrt{k}$ times of the epochs used in training the large model.

6.4.2 Experimental Results

6.4.2.1 Evaluation Protocols

FLOPs and the number of parameters are two widely used criteria for evaluating the pruning methods. In this paper, we are more focused on compressing the FLOPs of models, which has a large impact on the inference speed [Choudhary 20]. We also report the number of parameters of the pruned model, which is compressed at the same time during the pruning procedure. We compare the accuracies of pruned models with similar FLOPs (or FLOPs of pruned models with similar accuracies) in our experiments. The **compression ratio** (CR) of a model is defined as FLOPs (or number of parameters) before pruning divided by FLOPs (or number of parameters) after pruning.

Method	Baseline Acc.	Pruned Acc.	Acc. Drop	Parameters(CR)	FLOPs(CR)
SSS [Huang 18]	93.90%	93.80%	0.10%	8.1M(1.85×)	210.5M(1.50×)
L1 [Li 16]	93.25%	93.40%	**-0.15%**	5.4M(2.77×)	206.6M(1.56×)
GAL [Lin 19]	**93.96%**	93.77%	0.19%	**3.4M(4.41×)**	189.5M(1.65×)
Slimming [Liu 17]	93.94%	93.89%	0.05%	5.1M(2.94×)	174.3M(1.8×)
FETS(Ours)	93.94%	**94.08%**	-0.14%	**3.4M(4.41×)**	**156.8M(2.0×)**
HRank [Lin 20]	**93.96%**	92.34%	1.62%	2.6M(5.58×)	108.6M(2.88×)
Slimming [Liu 17]	93.94%	93.23%	0.71%	2.8M(5.35×)	105.3M(2.97×)
SSS [Huang 18]	93.90%	93.15%	0.75%	5.3M(2.83×)	98.5M(3.42×)
FETS(Ours)	93.94%	**93.60%**	**0.34%**	**0.8M(18.72×)**	**76.52M(4.1×)**
Slimming [Liu 17]	93.94%	91.82%	2.12%	1.6M(9.36×)	80.4M(3.9×)
HRank [Lin 20]	**93.96%**	91.23%	2.73%	1.8M(8.31×)	73.7M(4.26×)
FETS(Ours)	93.94%	**92.48%**	**1.46%**	**0.4M(37.45×)**	**52.2M(6.0×)**
SSS [Huang 18]	93.90%	91.25%	2.65%	4.9M(3.02×)	51.0M(6.15×)
FETS(Ours)	93.94%	92.08%	1.86%	**0.3M(49.9×)**	**39.2M(8.0×)**

Table 6.1: Pruning Results of VGG-16 on CIFAR-10. Results are divided into three groups by FLOPs compression ratio (CR). "Acc. Drop" indicates the accuracy drop of the pruned model compared with the original model. Smaller "Acc. Drop" value indicates better performance, while negative value means the performance is improved after pruning.

6.4.2.2 Results on CIFAR-10

VGG-16. In Table 6.1, we evaluate different pruning methods on the VGG-16 network. All the pruning results are divided into three groups according to the FLOPs CR. Under a low CR range from 1.5 to 2.0, the performance drops are small for all the methods, which indicates the VGG-16 network is redundant for CIFAR-10. The proposed FETS and L1 [Li 16] both achieve an accuracy improvement, but our FETS has an obviously higher baseline accuracy and larger FLOPs CR. GAL [Lin 19] achieves same Parameter CR of 4.41 with FETS, but the accuracy and FLOPs CR are worse. When the FLOPs CR increases, our FETS performs much better than all the other methods. We achieve a FLOPs CR of 8.0 with only an accuracy drop of 1.86%. In contrast, HRank [Lin 20] and Slimming [Liu 17] compress the FLOPs by about 4 times with an accuracy drop of 2.73% and 2.12%, respectively. Although without any special designs for compressing the number of parameters, we still achieve surprisingly high parameter CR. Under similar FLOPs CR of 4.0, we obtain a parameter CR of 18.72, which is much higher than HRank [Lin 20] (8.31) and Slimming [Liu 17] (9.36). When FLOPs CR reaches 8.0, the number of parameters are decreased by almost 50 times. We attribute it to our pruning strategy for VGG-16: We allow the whole layer to be pruned if all the filters of that layer are below the TS threshold. In summary, our FETS outperforms other methods by a remarkable margin on the VGG-16 network.

ResNet-56. In Table 6.2, we evaluate different pruning methods on ResNet-56. With a FLOPs CR of 2.2, our FETS obtains an accuracy improvement of 0.05%. GAL [Lin 19] achieves a slightly better improvement of 0.12%, but its baseline accuracy (93.26%), FLOPs CR (1.6) and parameter CR (1.13) are lower. When the FLOPs CR becomes higher, the accuracy decrease

Method	Baseline Acc.	Pruned Acc.	Acc. Drop	Parameters(CR)	FLOPs(CR)
L1 [Li 16]	93.04%	93.06%	-0.02%	0.73M(1.16×)	90.9M(1.38×)
GAL [Lin 19]	93.26%	93.38%	**-0.12%**	0.75M(1.13×)	78.4M(1.60×)
NISP [Yu 18]	93.04%	93.01%	0.03%	0.49M(1.74×)	70.9M(1.77×)
Slimming [Liu 17]	93.50%	93.54%	-0.04%	0.53M(1.59×)	64.0M(1.96×)
HRank [Lin 20]	93.26%	93.17%	0.09%	0.49M(1.73×)	62.72M(2.00×)
FPGM [He 19]	**93.59%**	93.49%	0.10%	-	59.4M(2.11×)
FETS(Ours)	93.58%	**93.63%**	-0.05%	**0.33M(2.57×)**	**57.0M(2.20×)**
GAL [Lin 19]	93.26%	91.58%	1.68%	0.29M(2.93×)	49.99M(2.51×)
Slimming [Liu 17]	93.50%	91.29%	2.21%	0.28M(3.02×)	33.0M(3.80×)
HRank [Lin 20]	93.26%	90.72%	2.54%	0.27M(3.15×)	32.52M(3.86×)
FETS(Ours)	**93.58%**	**92.08%**	**1.50%**	**0.21M(4.05×)**	**31.6M(3.97×)**
FETS(Ours)	**93.58%**	91.01%	2.57%	**0.14M(5.97×)**	**20.8M(6.02×)**

Table 6.2: Pruning Results of ResNet-56 on CIFAR-10

Method	Baseline Acc.	Pruned Acc.	Acc. Drop	Parameters(CR)	FLOPs(CR)
GAL [Lin 19]	**94.81%**	**94.61%**	0.20%	0.67M(1.55×)	182.92M(1.54×)
HRank [Lin 20]	**94.81%**	94.24%	0.57%	0.66M(1.58×)	167.41M(1.69×)
Variational [Zhao 19]	94.11%	93.16%	0.95%	0.42M(2.48×)	156.0M(1.81×)
Slimming [Liu 17]	93.89%	94.35%	**-0.46%**	0.35M(2.97×)	120.0M(2.35×)
FETS(Ours)	94.31%	**94.61%**	-0.30%	**0.33M(3.15×)**	**113.7M(2.48×)**
HRank [Lin 20]	**94.81%**	93.68%	1.13%	0.48M(2.17×)	110.1M(2.56×)
GAL [Lin 19]	**94.81%**	93.23%	1.58%	0.26M(4.00×)	80.89M(3.49×)
FETS(Ours)	94.31%	**93.81%**	**0.50%**	**0.21M(4.84×)**	**64.2M(4.39×)**

Table 6.3: Pruning Results of DenseNet-40 on CIFAR-10

of FETS is slower than others. With a FLOPs CR of 3.97, the accuracy drop of FETS is only 1.5%, while Slimming [Liu 17] and HRank [Lin 20] have a drop of 2.21% and 2.54%, respectively. When we prune more aggressively to a FLOPs CR of 6.02, the accuracy only drops by 2.57%. In terms of Parameter CR, FETS also performs better than other methods, but the leading advantage is not as big as on VGG-16. We attribute it to different pruning strategy we adopt for ResNet-56, where we prune same percentage of filters for each layer, without pruning on the layer-level. In summary, our FETS also outperforms others on ResNet-56.

DenseNet-40. Table 6.3 shows the comparison results on DenseNet-40. Compared with GAL [Lin 19], HRank [Lin 20] and Variational [Zhao 19], our FETS achieves better FLOPs CR of 2.48 with an accuracy improvement of 0.30%. Although Slimming [Liu 17] obtains a bigger accuracy improvement of 0.46%, its baseline accuracy is obviously lower than others, which leaves more potential for improvement. After pruning (and finetuning), FETS achieves better accuracy of 94.61% than Slimming [Liu 17] (94.35%). When pruning with higher FLOPs CR, the accuracy drop of FETS is obviously smaller than others. It demonstrates our FETS can also work well on DenseNet-40.

Method	Baseline Top1 Acc.	Pruned Top1 Acc.	Baseline Top5 Acc.	Pruned Top5 Acc.	Top1 Acc. Drop	Top5 Acc. Drop	Parameters(CR)	FLOPs(CR)
SSS [Huang 18]	76.12%	71.82%	92.86%	**90.79%**	4.30%	**2.07%**	15.6M(1.63×)	2.33B(1.76×)
GDP [Lin 18]	75.13%	70.93%	92.30%	90.14%	4.20%	2.16%	-	1.57B(2.61×)
Taylor-FO-BN [Molchanov 19]	**76.18%**	71.69%	-	-	4.49%	-	7.9M(3.22×)	1.34B(3.05×)
GAL [Lin 19]	76.15%	69.31%	**92.87%**	89.12%	6.84%	3.75%	10.2M(2.50×)	1.11B(3.68×)
ThiNet [Luo 17]	72.88%	68.42%	91.14%	88.30%	4.46%	2.84%	8.7M(2.94×)	1.10B(3.72×)
HRank [Lin 20]	76.15%	69.10%	**92.87%**	89.58%	7.05%	3.29%	8.3M(3.08×)	0.98B(4.17×)
FETS(Ours)	76.15%	**72.06%**	**92.87%**	90.61%	**4.09%**	2.26%	**7.6M(3.36×)**	**0.95B(4.31×)**

Table 6.4: Pruning Results of ResNet-50 on ImageNet

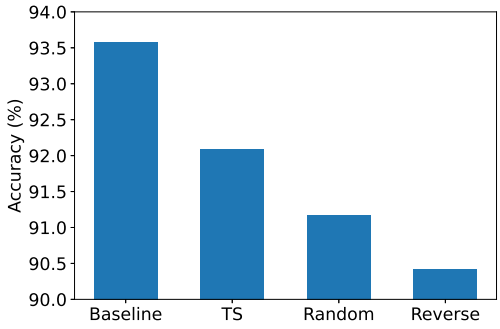


Figure 6.3: Models pruned with different variants of TS on ResNet-56, with a FLOPs CR of 4.

6.4.2.3 Results on ImageNet

We show the results of different pruning methods on ResNet-50 in Table 6.4. Our FETS achieves a FLOPs CR of 4.31 with a small top-1 and top-5 accuracy drop of 4.09% and 2.26%, respectively. It surpasses the performances of GDP [Lin 18], Taylor-FO-BN [Molchanov 19], GAL [Lin 19], ThiNet [Luo 17] and HRank [Lin 20] in almost all aspects. SSS [Huang 18] obtains the smallest top-5 accuracy drop of 2.07%, which is slightly better than FETS. But the FLOPs CR of SSS [Huang 18] is only 1.76, which is significantly poorer than our FETS. In general, FETS performs better than other methods in the large-scale ImageNet dataset.

6.4.3 Ablation Studies

In this section, we perform extensive ablation studies to explore the impacts of different modules in our proposed method. All these studies are conducted on the CIFAR 10 dataset.

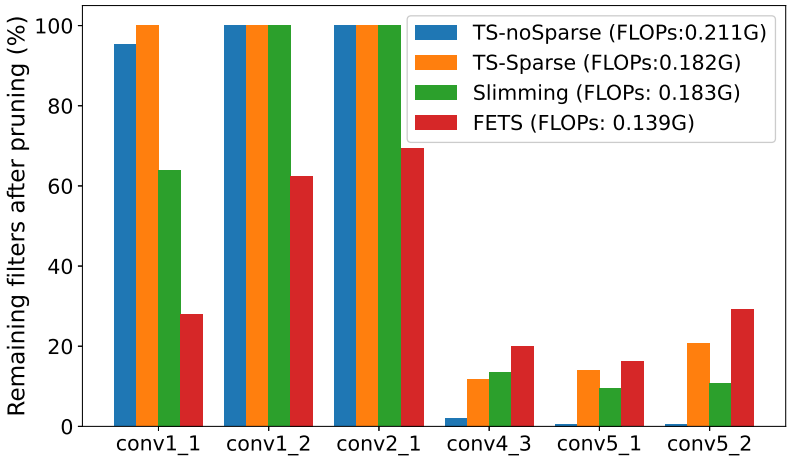


Figure 6.4: The remaining filters in some layers of VGG-16 network (60% filters are pruned). Our FETS prunes more computationally-intensive filters (from shallower layers), which achieves a better FLOPs CR.

6.4.3.1 Effectiveness of TS

In order to demonstrate the effectiveness of TS (Eq. 6.3), we prune models in two different ways to compare with the model pruned on TS: 1) Random: The filters are randomly pruned. 2) Reverse: Filters with the largest TS values are pruned. Other settings are unchanged as in FETS. As is shown in Fig. 6.3, TS consistently outperforms the other two methods on ResNet-56 under the FLOPs CR of 4. "Reverse" has a significantly worse accuracy, which indicates the importance of filters with high TS values.

6.4.3.2 FLOPs-efficient Group Sparsity

In order to explore the effect of our proposed FLOPs-efficient group sparsity (see Section 6.3.3), we evaluate the number of remaining filters after pruning in each layer in a VGG-16 network, which is trained with four different sparsity strategies as is shown in Fig. 6.4: 1) TS-noSparse: Train a model in normal way without using group sparsity during training (equal to setting λ_g as zero in Eq. 6.5), and prune the model according to TS. 2) TS-Sparse: Train a model with group Lasso regulation, and prune the model according to TS. All β_j^i in Eq. 6.5 are set as 1, which gives filters from different layers the same weight during sparsity training. 3) Slimming [Liu 17]: Another method that imposes a channel wise sparsity during training. The trained model is pruned based on the scale factors in BN layers 4) FETS: Our proposed method which trains the

Method	Acc. Drop	FLOPs(CR)	Finetune Epochs
HRank [Lin 20]	2.54%	32.5M(3.86×)	390
Slimming [Liu 17]	2.21%	33.0M(3.80×)	**160**
FETS(Ours)	**1.50%**	**31.6M(4.05×)**	320
FETS*(Ours)	1.84%	**31.6M(4.05×)**	**160**

Table 6.5: Different finetuning epochs on ResNet-56

model with Eq. 6.5 and Eq. 6.9 and prunes the model with TS. We can find from Fig. 6.4 that "TS-noSparse" tends to select almost all filters from deeper layers for pruning. When training with sparsity, "TS-Sparse" and Slimming [Liu 17] prune fewer filters from deeper layers, but the pruning ratio of the shallower layers are still low. Our FETS chooses to prune obviously more filters from the shallower (and also more computationally-intensive) layers, which achieves a much higher FLOPs CR.

6.4.3.3 Number of Finetuning Epochs

One important problem is how many epochs are needed to finetune the pruned model. Although it is not clear how to train the model to "fully convergence", finetuning for more epochs seldomly harms the performance of the model [Liu 18b]. As is mentioned in Section 6.4.1, we finetune the pruned model with a FLOPs CR of k for $\sqrt{k}$ times of the base epochs. To conduct a more equal comparison, we evaluate our FETS on ResNet-56 with a smaller finetuning epoch of 160, which is referred to as "FETS*" in Table 6.5. The results show that when finetuning for fewer epochs, the accuracy of FETS slightly drops, but it still outperforms other methods that have the same or even more finetuning epochs.

6.4.3.4 More Efficient Pruning

Currently we can find the weights of many popular CNN models pre-trained on large datasets (e.g. ImageNet), which can be utilized to speed up the model re-training in new target datasets or tasks. Since the proposed TS (Eq. 6.3) measures the importance of filters in a data-independent manner, our FETS can be used in a more efficient way: Given the pre-trained model, we first directly prune the model according to TS with a smaller (normally half or one third) FLOPs CR, then we apply the normal pruning procedure as explained before to achieve the target FLOPs CR. There are two main advantages of this approach: 1) After the first pruning, the computation cost of the model is reduced, which saves time for the following train-prune-finetune procedure. 2) It helps to prune large-scale networks (e.g. ResNet-164) on resource-constrained devices (e.g. limited GPU memory), because it reduces the memory and storage cost of these models after the first pruning. In our experiments, this efficient version of

Method	Easy	Moderate	Hard	Inference
FRCNN	72.8%	**67.3%**	**61.8%**	0.031s
Pruned	**73.0%**	66.8%	61.2%	**0.011s**

Table 6.6: The detection results of Faster-RCNN before and after pruning on KITTI dataset

FETS achieves similar performance with the normal one, while we save about 30% of the time for the whole pruning procedure.

6.4.3.5 Performance on Detection Tasks

In order to evaluate our method in other computer vision tasks, we apply our FETS to the Faster-RCNN detector [Ren 15] which uses VGG-16 as its backbone feature extractor. The experiments are conducted on the KITTI dataset [Geiger 12], which consists of 7481 training images and 7518 test images, including 80256 labeled objects in total. As the annotations of the test set are not available, we split the training set of 60% and 40% for training and validation. We first train a Faster-RCNN as baseline, then we use FETS to obtain a pruned detector. In Table 6.6, the detection results are evaluated in three difficult levels, which indicate different sizes, occlusion levels and truncation levels of the targets. When pruning 70% of the filters in VGG-16, the pruned Faster-RCNN achieves an inference speedup of 2.8 times, while the accuracies are almost not influenced (even increased for Easy test cases). It demonstrates the good generalization ability of our proposed method.

6.5 Conclusions

In this chapter, we propose a data-independent criterion named TS to prune unimportant filters in CNN. The idea of TS is to estimate the importance of a filter according to its connections to the next layer, i.e., the weights of filters in the next layer. Because a filter that have weak connections with next layer will have little influence on the output of the CNN, therefore it can be safely pruned from the network without obviously sacrificing the accuracy. It is simple to implement, and outperforms other norm-based filter pruning criteria.

Normally, pruning on a trained CNN model according to some criterion has a limited performance. It works well when the pruning ratio is low. But for higher pruning ratios the accuracy of the model can drop dramatically. Thus it is necessary to squeeze the model during training, i.e., forcing the model to use only limited number of filters, to facilitate the following pruning step. In addition, we have observed that filters from computationally-intensive layers, i.e., earlier layers in the network, are more sensitive to pruning. So if we prune more filters from the earlier layers, it reduces the computation cost efficiently, but also results in larger accuracy drop. To solve this problem, we propose a

FLOPs-efficient group Lasso approach for training, which guides the network to use fewer filters in these computationally-intensive layers. Extensive experimental results show that the proposed method achieves significantly better pruning performance than state-of-the-art methods. We also conduct several ablation studies to demonstrate the effectiveness of different modules and the generalization ability of our method on different computer vision tasks.

7

Conclusions and Future Work

7.1 Conclusions

Pedestrian detection has been intensively studied over the past few decades. However, there is still room for improvement especially under some challenging conditions. In some real life situations like the urban streets, a pedestrian has a very high probability of being occluded by vehicles or other neighbouring pedestrians, which significantly reduces the accuracy of the pedestrian detection results. It is also more difficult to recognize a pedestrian when the background is complex or the light condition is poor. Also pedestrian detectors based on deep neural networks normally suffer from serious efficiency problems.

In this thesis, we studied these aforementioned issues and proposed corresponding solutions. Specifically, in Chapter 3, we proposed two novel methods to handle the occlusion problem. We first designed an occlusion-robust feature selection strategy, which utilized the average occlusion distribution of occluded pedestrian samples to guide the training of detectors. Then we proposed a more advanced deep learning based framework, which recognized the occlusion pattern of each pedestrian and then refined the features accordingly. Next, we studied how to better use depth images to enhance the ability of pedestrian detectors in Chapter 4 and Chapter 5. We first explored reusing the idea from Chapter 3, which refined the features from occluded pedestrians for better classification results. But in Chapter 4, we utilized depth images to guide the feature reweighting, which obviously improved the detection performance. In Chapter 5, we generated semantic segmentation masks from depth images, and then proposed an end-to-end joint training structure to handle detection and segmentation tasks simultaneously. Last but not least, in Chapter 6 we focused on the efficiency problem of pedestrian detectors, and proposed a novel filter pruning method for deep neural networks, which largely decreased the cost of deploying such complex detectors on resource limited devices in the real life. The main contributions are summarized as follows.

1. **Partial-occlusion Handling for Pedestrian Detection:** In Chapter 3, we proposed two occlusion-robust pedestrian detection methods, which are based on conventional hand-crafted features and on self-learned deep convolutional features, respectively. First, we observed that most conventional pedestrian detectors intentionally avoid using occluded pedestrian samples for training. This is because conventional machine learning based classifiers are not able to distinguish the occluding objects from the real pedestrian, which will result in learning wrong characteristics when training with occluded pedestrian samples. Therefore, we proposed a new feature selection strategy according to the average occlusion distribution of occluded pedestrian samples. The idea is to guide the classifier to choose more features from the regions with lower probability of being occluded. However, it is a rough strategy to select the feature of each pedestrian sample according to the average occlusion distribution of all samples. So we further improved this method by recognizing the occlusion pattern of each pedestrian sample, and then refined the convolutional features of each sample accordingly to obtain a better classification result. Experimental results demonstrated that our methods achieved significantly lower miss-rate for detecting occluded pedestrian samples, while the detection performance of the non-occluded samples were not obviously affected.

2. **Pedestrian Detection in RGB-D Images:** In Chapter 4, we focused on utilizing depth modality to assist the pedestrian detection task. Depth images can provide clearer spatial information of objects in a scene, which help to better distinguish a pedestrian with the background or the occluding objects. Although RGB-D pedestrian detection has been studied for several years, most of the methods adopt a very similar process, which first extracts features from RGB and depth images independently, and then fuse these features in later stages. In Chapter 3, we propose a new method, which uses the depth images as guidance to refine the convolutional features extracted from RGB images. The refined features that from the real pedestrian region will be assigned higher weights, while the features from the background or occlusion regions will have lower weights. Furthermore, by exploring the projection relationship from depth images, some false candidate region proposals are removed, which obviously improves efficiency of the downstream classification tasks. Experimental results show that our proposed RGBD method outperforms other state-of-the-art methods.

3. **Weak Segmentation Supervised Pedestrian Detection:** Semantic segmentation has been used successfully as a complementary information source in pedestrian detection. However, it requires accurate pixel-level semantic segmentation annotations for training, but it is extremely time consuming to obtain these. In Chapter 5, we solve this problem by using weak segmentation masks automatically generated by depth images.

This enables joint semantic segmentation and pedestrian detection with only ground truth bounding boxes for training. We show that this joint training boosts the performance of the pedestrian detector. Moreover, we have observed that the generated segmentation maps can often indicate the existence of pedestrians, even though they are missed by the classifiers. Thus, we propose to fuse the outputs of the classification network and the generated segmentation masks to further improve the detection performance. The proposed method has achieved state-of-the-art performance on several RGBD pedestrian datasets. Another advantage of this method is we also obtain good quality segmentation results, without training with the expensive pixel-level segmentation annotations.

4. **Filter Pruning for Neural Network Acceleration:** As the novel pedestrian detectors are widely based on deep neural networks, they normally require high computation and storage costs, and are therefore difficult to be deployed in real life applications. In Chapter 6, we are focused on model pruning to accelerate the inference speed of convolutional neural networks. First, we propose a norm-based filter pruning criterion. It measures the importance of filters by exploring the transfer scale (TS) of its feature maps in the next layer. In this way, filters that have weak connections with the next layer will be pruned, which has little influence on the whole network. Second, we propose a new group Lasso approach for training, which guides the network to use fewer filters in the computationally-intensive layers. This is inspired by our observation that filters in the earlier layers contain more FLOPs, and are also more sensitive to pruning. Thus, after pruning, more filters from earlier layers are removed from the network, which results in a better FLOPs compression performance, but with a limited accuracy drop.

7.2 Future Research

Following the work derived from the present thesis, even though some important achievements have been made, several challenges still remain. As with any new work, there are some open avenues for future research that deserve attention and will be explored in the future. In the following, we list the most relevant of these perspectives for our future work:

1. It would be interesting to integrate human pose estimation into our proposed occlusion handling method in Chapter 2. In our proposed method, we classify common occlusion patterns into five categories: fully-visible, left occluded, right occluded, bottom 1/3 occluded and bottom 2/3 occluded. Then the feature of each pedestrian is refined according to its corresponding occlusion pattern. However, a pedestrian can have a variety of different postures, while five patterns are too rough to describe these. As pose estimation can provide a more detailed set of keypoints of the body, it is possible to generate a specialized occlusion pattern

for each pedestrian, and therefore guide the detector to better recognize pedestrians occluded at different parts of the body.

2. Considering some real life applications like autonomous driving in the urban streets, it is more valuable to handle the occlusion problem in video sequences. For instance, when a pedestrian approaches a car, it is possible to predict a following sequence of occlusion patterns in advance. It saves the challenging task of recognizing the occlusion pattern of each pedestrian based on a single frame, and is more efficient and reliable.

3. In Chapter 3 and 4, we utilized depth modality as a complement to RGB images for pedestrian detection. In the future, we will explore the fusion of more complementary data sources, such as thermal, LiDAR and Radar data.

4. In Chapter 5, we proposed a new criterion based on the filter weights to evaluate the importance of each neuron in the neural network. However, it is a heuristic criterion, and we believe a more sophisticated criterion derived from mutual information would be better founded. Mutual information between neurons in multiple layers of neural networks can be helpful to find out how strong these neurons are connected. Moreover, we will explore filter pruning on more efficient CNN structures, such as MobileNet and EfficientNet.

Bibliography

[Angelova 15] Anelia Angelova, Alex Krizhevsky, Vincent Vanhoucke, Abhijit S Ogale & Dave Ferguson. *Real-Time Pedestrian Detection with Deep Network Cascades.* In BMVC, pages 32–1, 2015.

[Bagautdinov 15a] T. Bagautdinov, F. Fleuret & P. Fua. *Probability occupancy maps for occluded depth images.* In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2829–2837, 2015.

[Bagautdinov 15b] Timur Bagautdinov, Francois Fleuret & Pascal Fua. *Probability Occupancy Maps for Occluded Depth Images.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015.

[Benenson 12] Rodrigo Benenson, Markus Mathias, Radu Timofte & Luc Van Gool. *Pedestrian detection at 100 frames per second.* In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pages 2903–2910. IEEE, 2012.

[Benenson 13] Rodrigo Benenson, Markus Mathias, Tinne Tuytelaars & Luc Van Gool. *Seeking the strongest rigid detector.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3666–3673, 2013.

[Bottou 07] Léon Bottou & Chih-Jen Lin. *Support vector machine solvers.* Large scale kernel machines, vol. 3, no. 1, pages 301–320, 2007.

[Braun 19] Markus Braun, Sebastian Krebs, Fabian B. Flohr & Dariu M. Gavrila. *EuroCity Persons: A Novel Benchmark for Person Detection in Traffic Scenes.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 41, pages 1844–1861, 2019.

[Brazil 17] Garrick Brazil, Xi Yin & Xiaoming Liu. *Illuminating pedestrians via simultaneous detection & segmentation.*

In Proceedings of the IEEE International Conference on Computer Vision, pages 4950–4959, 2017.

[Bucilua 06] Cristian Bucilua, Rich Caruana & Alexandru Niculescu-Mizil. *Model compression*. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 535–541, 2006.

[Cai 16] Zhaowei Cai, Quanfu Fan, Rogerio S Feris & Nuno Vasconcelos. *A unified multi-scale deep convolutional neural network for fast object detection*. In European conference on computer vision, pages 354–370. Springer, 2016.

[Camplani 16] Massimo Camplani, Adeline Paiement, Majid Mirmehdi, Dima Damen, Sion Hannuna, Tilo Burghardt & Lili Tao. *Multiple human tracking in RGB-depth data: a survey*. IET computer vision, vol. 11, no. 4, pages 265–285, 2016.

[Cao 19a] Jiale Cao, Yanwei Pang, Shengjie Zhao & Xuelong Li. *High-level semantic networks for multi-scale object detection*. IEEE Transactions on Circuits and Systems for Video Technology, vol. 30, pages 3372–3386, 2019.

[Cao 19b] Yanpeng Cao, Dayan Guan, Yulun Wu, Jiangxin Yang, Yanlong Cao & Michael Ying Yang. *Box-level segmentation supervised deep neural networks for accurate and real-time multispectral pedestrian detection*. ISPRS journal of photogrammetry and remote sensing, vol. 150, pages 70–79, 2019.

[Carreira-Perpinan 18] Miguel A Carreira-Perpinan & Yerlan Idelbayev. *Learning-Compression Algorithms for Neural Net Pruning*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 8532–8541, 2018.

[Choi 11] Wongun Choi, Caroline Pantofaru & Silvio Savarese. *Detecting and tracking people using an rgb-d camera via multiple detector fusion*. In 2011 IEEE international conference on computer vision workshops (ICCV workshops), pages 1076–1083. IEEE, 2011.

[Choudhary 20] Tejalal Choudhary, Vipul Mishra, Anurag Goswami & Jagannathan Sarangapani. *A comprehensive survey on model compression and acceleration*. Artificial Intelligence Review, pages 1–43, 2020.

[Cortes 95] Corinna Cortes & Vladimir Vapnik. *Support-vector networks*. Machine learning, vol. 20, no. 3, pages 273–297, 1995.

[Dalal 05] Navneet Dalal & Bill Triggs. *Histograms of oriented gradients for human detection*. In 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05), volume 1, pages 886–893. Ieee, 2005.

[Dimitrievski 18] Martin Dimitrievski, Peter Veelaert & Wilfried Philips. *Learning morphological operators for depth completion*. In International Conference on Advanced Concepts for Intelligent Vision Systems, pages 450–461. Springer, 2018.

[Dollár 09a] Piotr Dollár, Zhuowen Tu, Pietro Perona & Serge Belongie. *Integral channel features*. 2009.

[Dollár 09b] Piotr Dollár, Christian Wojek, Bernt Schiele & Pietro Perona. *Pedestrian detection: A benchmark*. In 2009 IEEE Conference on Computer Vision and Pattern Recognition, pages 304–311. IEEE, 2009.

[Dollár 10] Piotr Dollár, Serge J Belongie & Pietro Perona. *The Fastest Pedestrian Detector in the West.* In BMVC, volume 2, page 7, 2010.

[Dollar 12] Piotr Dollar, Christian Wojek, Bernt Schiele & Pietro Perona. *Pedestrian detection: An evaluation of the state of the art*. IEEE transactions on pattern analysis and machine intelligence, vol. 34, no. 4, pages 743–761, 2012.

[Dollár 14] Piotr Dollár, Ron Appel, Serge Belongie & Pietro Perona. *Fast feature pyramids for object detection*. IEEE transactions on pattern analysis and machine intelligence, vol. 36, no. 8, pages 1532–1545, 2014.

[Du 17] Xianzhi Du, Mostafa El-Khamy, Jungwon Lee & Larry Davis. *Fused DNN: A deep neural network fusion approach to fast and robust pedestrian detection*. In 2017 IEEE winter conference on applications of computer vision (WACV), pages 953–961. IEEE, 2017.

[Eitel 15] Andreas Eitel, Jost Tobias Springenberg, Luciano Spinello, Martin Riedmiller & Wolfram Burgard. *Multimodal deep learning for robust RGB-D object recognition*. In 2015 IEEE/RSJ International Conference on

Intelligent Robots and Systems (IROS), pages 681–687. IEEE, 2015.

[Enzweiler 10] Markus Enzweiler, Angela Eigenstetter, Bernt Schiele & Dariu M Gavrila. *Multi-cue pedestrian classification with partial occlusion handling.* In Computer vision and pattern recognition (CVPR), 2010 IEEE Conference on, pages 990–997. IEEE, 2010.

[Ertler 17] Christian Ertler, H Posseger, M Optiz & Horst Bischof. *Pedestrian detection in rgb-d images from an elevated viewpoint.* In 22nd Computer Vision Winter Workshop, 2017.

[Felzenszwalb 04] Pedro F Felzenszwalb & Daniel P Huttenlocher. *Efficient graph-based image segmentation.* International journal of computer vision, vol. 59, no. 2, pages 167–181, 2004.

[Felzenszwalb 10] Pedro F Felzenszwalb, Ross B Girshick, David McAllester & Deva Ramanan. *Object detection with discriminatively trained part-based models.* IEEE transactions on pattern analysis and machine intelligence, vol. 32, no. 9, pages 1627–1645, 2010.

[Fidler 13] Sanja Fidler, Roozbeh Mottaghi, Alan Yuille & Raquel Urtasun. *Bottom-up segmentation for top-down detection.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3294–3301, 2013.

[Freund 97] Yoav Freund & Robert E Schapire. *A decision-theoretic generalization of on-line learning and an application to boosting.* Journal of computer and system sciences, vol. 55, no. 1, pages 119–139, 1997.

[Friedman 00] Jerome Friedman, Trevor Hastie, Robert Tibshirani*et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors).* The annals of statistics, vol. 28, no. 2, pages 337–407, 2000.

[Gavrila 07] Dariu M Gavrila & Stefan Munder. *Multi-cue pedestrian detection and tracking from a moving vehicle.* International journal of computer vision, vol. 73, no. 1, pages 41–59, 2007.

[Geiger 12] Andreas Geiger, Philip Lenz & Raquel Urtasun. *Are we ready for autonomous driving? the kitti vision benchmark suite*. In 2012 IEEE Conference on Computer Vision and Pattern Recognition, pages 3354–3361. IEEE, 2012.

[Girshick 14] Ross Girshick, Jeff Donahue, Trevor Darrell & Jitendra Malik. *Rich feature hierarchies for accurate object detection and semantic segmentation*. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 580–587, 2014.

[Girshick 15] Ross Girshick. *Fast r-cnn*. In Proceedings of the IEEE international conference on computer vision, pages 1440–1448, 2015.

[González 13] Domingo Iván Rodríguez González & Jean-Bernard Hayet. *Fast Human Detection in RGB-D Images with Progressive SVM-Classification*. In Pacific-Rim Symposium on Image and Video Technology, pages 337–348. Springer, 2013.

[Guo 18a] Zhixin Guo, Wenzhi Liao, Peter Veelaert & Wilfried Philips. *Occlusion-robust detector trained with occluded pedestrians*. In 7th International Conference on Pattern Recognition Applications and Methods (ICPRAM), pages 86–94. SCITEPRESS-Science and Technology Publications, 2018.

[Guo 18b] Zhixin Guo, Wenzhi Liao, Yifan Xiao, Peter Veelaert & Wilfried Philips. *An occlusion-robust feature selection framework in pedestrian detection*. Sensors, vol. 18, no. 7, page 2272, 2018.

[Guo 19] Zhixin Guo, Wenzhi Liao, Yifan Xiao, Peter Veelaert & Wilfried Philips. *Deep learning fusion of RGB and depth images for pedestrian detection*. In 30th British Machine Vision Conference, pages 1–13, 2019.

[Guo 21a] Zhixin Guo, Wenzhi Liao, Yifan Xiao, Peter Veelaert & Wilfried Philips. *Weak Segmentation Supervised Deep Neural Networks for Pedestrian Detection*. Pattern Recognition, 2021.

[Guo 21b] Zhixin Guo, Yifan Xiao, Wenzhi Liao, Peter Veelaert & Wilfried Philips. *FLOPs-efficient filter pruning via transfer scale for neural network acceleration*. Journal of Computational Science, 2021.

[Gupta 14] Saurabh Gupta, Ross Girshick, Pablo Arbeláez & Jitendra Malik. *Learning rich features from RGB-D images for object detection and segmentation.* In European Conference on Computer Vision, pages 345–360. Springer, 2014.

[Gupta 16] Saurabh Gupta, Judy Hoffman & Jitendra Malik. *Cross modal distillation for supervision transfer.* In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2827–2836, 2016.

[Han 15a] Song Han, Huizi Mao & William J Dally. *Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding.* arXiv preprint arXiv:1510.00149, 2015.

[Han 15b] Song Han, Jeff Pool, John Tran & William Dally. *Learning both weights and connections for efficient neural network.* In Advances in neural information processing systems, pages 1135–1143, 2015.

[He 16] Kaiming He, Xiangyu Zhang, Shaoqing Ren & Jian Sun. *Deep residual learning for image recognition.* In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[He 17a] Kaiming He, Georgia Gkioxari, Piotr Dollár & Ross Girshick. *Mask r-cnn.* In Proceedings of the IEEE international conference on computer vision, pages 2961–2969, 2017.

[He 17b] Yihui He, Xiangyu Zhang & Jian Sun. *Channel pruning for accelerating very deep neural networks.* In Proceedings of the IEEE International Conference on Computer Vision, pages 1389–1397, 2017.

[He 18] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu & Yi Yang. *Soft filter pruning for accelerating deep convolutional neural networks.* arXiv preprint arXiv:1808.06866, 2018.

[He 19] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu & Yi Yang. *Filter pruning via geometric median for deep convolutional neural networks acceleration.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4340–4349, 2019.

[Hoffman 16] Judy Hoffman, Saurabh Gupta & Trevor Darrell. *Learning with side information through modality hallucination.* In Proceedings of the IEEE Conference on

Computer Vision and Pattern Recognition, pages 826–834, 2016.

[Hosang 15] Jan Hosang, Mohamed Omran, Rodrigo Benenson & Bernt Schiele. *Taking a deeper look at pedestrians*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4073–4082, 2015.

[Hu 16] Hengyuan Hu, Rui Peng, Yu-Wing Tai & Chi-Keung Tang. *Network trimming: A data-driven neuron pruning approach towards efficient deep architectures*. arXiv preprint arXiv:1607.03250, 2016.

[Huang 17] Gao Huang, Zhuang Liu, Laurens Van Der Maaten & Kilian Q Weinberger. *Densely connected convolutional networks*. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4700–4708, 2017.

[Huang 18] Zehao Huang & Naiyan Wang. *Data-driven sparse structure selection for deep neural networks*. In Proceedings of the European conference on computer vision (ECCV), pages 304–320, 2018.

[Hubara 17] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv & Yoshua Bengio. *Quantized neural networks: Training neural networks with low precision weights and activations*. The Journal of Machine Learning Research, vol. 18, no. 1, pages 6869–6898, 2017.

[Jafari 14] Omid Hosseini Jafari, Dennis Mitzel & Bastian Leibe. *Real-time RGB-D based people detection and tracking for mobile robots and head-worn cameras*. In 2014 IEEE International Conference on Robotics and Automation (ICRA), pages 5636–5643. IEEE, 2014.

[Konig 17] Daniel Konig, Michael Adam, Christian Jarvers, Georg Layher, Heiko Neumann & Michael Teutsch. *Fully convolutional region proposal networks for multispectral person detection*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 49–56, 2017.

[Krizhevsky 09] Alex Krizhevsky, Geoffrey Hinton*et al. Learning multiple layers of features from tiny images*. 2009.

[Krizhevsky 12] Alex Krizhevsky, Ilya Sutskever & Geoffrey E Hinton. *Imagenet classification with deep convolutional neural networks*. Advances in neural information processing systems, vol. 25, pages 1097–1105, 2012.

[LeCun 89] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard & Lawrence D Jackel. *Backpropagation applied to handwritten zip code recognition.* Neural computation, vol. 1, no. 4, pages 541–551, 1989.

[LeCun 90] Yann LeCun, John S Denker & Sara A Solla. *Optimal brain damage.* In Advances in neural information processing systems, pages 598–605, 1990.

[Li 16] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet & Hans Peter Graf. *Pruning filters for efficient convnets.* arXiv preprint arXiv:1608.08710, 2016.

[Li 18] Chong Li & CJ Richard Shi. *Constrained optimization based low-rank approximation of deep neural networks.* In Proceedings of the European Conference on Computer Vision (ECCV), pages 732–747, 2018.

[Lin 14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár & C Lawrence Zitnick. *Microsoft coco: Common objects in context.* In European conference on computer vision, pages 740–755. Springer, 2014.

[Lin 18] Shaohui Lin, Rongrong Ji, Yuchao Li, Yongjian Wu, Feiyue Huang & Baochang Zhang. *Accelerating Convolutional Networks via Global & Dynamic Filter Pruning.* In IJCAI, pages 2425–2432, 2018.

[Lin 19] Shaohui Lin, Rongrong Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang & David Doermann. *Towards optimal structured cnn pruning via generative adversarial learning.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2790–2799, 2019.

[Lin 20] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian & Ling Shao. *HRank: Filter Pruning using High-Rank Feature Map.* In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 1529–1538, 2020.

[Liu 17] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan & Changshui Zhang. *Learning efficient convolutional networks through network slimming.* In Proceedings of the IEEE International Conference on Computer Vision, pages 2736–2744, 2017.

[Liu 18a] Tianrui Liu & Tania Stathaki. *Faster R-CNN for robust pedestrian detection using semantic segmentation network.* Frontiers in neurorobotics, vol. 12, page 64, 2018.

[Liu 18b] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang & Trevor Darrell. *Rethinking the value of network pruning.* arXiv preprint arXiv:1810.05270, 2018.

[Louizos 17] Christos Louizos, Max Welling & Diederik P Kingma. *Learning Sparse Neural Networks through L_0 Regularization.* arXiv preprint arXiv:1712.01312, 2017.

[Lowe 04] David G Lowe. *Distinctive image features from scale-invariant keypoints.* International journal of computer vision, vol. 60, no. 2, pages 91–110, 2004.

[Luo 17] Jian-Hao Luo, Jianxin Wu & Weiyao Lin. *Thinet: A filter level pruning method for deep neural network compression.* In Proceedings of the IEEE international conference on computer vision, pages 5058–5066, 2017.

[Luo 20] Jian-Hao Luo & Jianxin Wu. *Neural Network Pruning with Residual-Connections and Limited-Data.* In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 1458–1467, 2020.

[Mathias 13] Markus Mathias, Rodrigo Benenson, Radu Timofte & Luc Van Gool. *Handling occlusions with franken-classifiers.* In Proceedings of the IEEE International Conference on Computer Vision, pages 1505–1512, 2013.

[Molchanov 19] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio & Jan Kautz. *Importance estimation for neural network pruning.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 11264–11272, 2019.

[Nam 14] Woonhyun Nam, Piotr Dollár & Joon Hee Han. *Local decorrelation for improved detection.* Eprint Arxiv, 2014.

[Ohn-Bar 16] Eshed Ohn-Bar & Mohan M Trivedi. *To boost or not to boost? On the limits of boosted trees for object detection.* In Pattern Recognition (ICPR), 2016 23rd International Conference on, pages 3350–3355. IEEE, 2016.

[Ojala 02] Timo Ojala, Matti Pietikainen & Topi Maenpaa. *Multiresolution gray-scale and rotation invariant texture classification with local binary patterns.* IEEE Transactions on pattern analysis and machine intelligence, vol. 24, no. 7, pages 971–987, 2002.

[Ophoff 19] Tanguy Ophoff, Kristof Van Beeck & Toon Goedemé. *Exploring RGB+ Depth Fusion for Real-Time Object Detection.* Sensors, vol. 19, no. 4, page 866, 2019.

[Ouyang 13a] Wanli Ouyang & Xiaogang Wang. *Joint deep learning for pedestrian detection.* In Proceedings of the IEEE International Conference on Computer Vision, pages 2056–2063, 2013.

[Ouyang 13b] Wanli Ouyang & Xiaogang Wang. *Single-pedestrian detection aided by multi-pedestrian detection.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3198–3205, 2013.

[Ouyang 16] Wanli Ouyang, Xingyu Zeng & Xiaogang Wang. *Partial occlusion handling in pedestrian detection with a deep model.* IEEE Transactions on Circuits and Systems for Video Technology, vol. 26, no. 11, pages 2123–2137, 2016.

[Paisitkriangkrai 16] Sakrapee Paisitkriangkrai, Chunhua Shen & Anton van den Hengel. *Pedestrian detection with spatially pooled features and structured ensemble learning.* IEEE transactions on pattern analysis and machine intelligence, vol. 38, no. 6, pages 1243–1257, 2016.

[Paszke 17] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga & Adam Lerer. *Automatic differentiation in pytorch.* 2017.

[Premebida 14] Cristiano Premebida, Joao Carreira, Jorge Batista & Urbano Nunes. *Pedestrian detection combining rgb and dense lidar data.* In 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 4112–4117. IEEE, 2014.

[Qi 18] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su & Leonidas J Guibas. *Frustum pointnets for 3d object detection from rgb-d data.* In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 918–927, 2018.

[Redmon 17] Joseph Redmon & Ali Farhadi. *YOLO9000: better, faster, stronger*. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 7263–7271, 2017.

[Ren 15] Shaoqing Ren, Kaiming He, Ross Girshick & Jian Sun. *Faster r-cnn: Towards real-time object detection with region proposal networks*. In Advances in neural information processing systems, pages 91–99, 2015.

[Ren 17] Jimmy Ren, Xiaohao Chen, Jianbo Liu, Wenxiu Sun, Jiahao Pang, Qiong Yan, Yu-Wing Tai & Li Xu. *Accurate single stage detector using recurrent rolling convolution*. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 5420–5428, 2017.

[Russakovsky 15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein*et al. Imagenet large scale visual recognition challenge*. International journal of computer vision, vol. 115, no. 3, pages 211–252, 2015.

[Sainath 13] Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy & Bhuvana Ramabhadran. *Low-rank matrix factorization for deep neural network training with high-dimensional output targets*. In 2013 IEEE international conference on acoustics, speech and signal processing, pages 6655–6659. IEEE, 2013.

[Sermanet 13] Pierre Sermanet, Koray Kavukcuoglu, Soumith Chintala & Yann Lecun. *Pedestrian Detection with Unsupervised Multi-Stage Feature Learning*. In Computer Vision and Pattern Recognition, 2013.

[Simonyan 14] Karen Simonyan & Andrew Zisserman. *Very deep convolutional networks for large-scale image recognition*. arXiv preprint arXiv:1409.1556, 2014.

[Spinello 11] Luciano Spinello & Kai O Arras. *People detection in RGB-D data*. In 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 3838–3843. IEEE, 2011.

[Tang 14] Siyu Tang, Mykhaylo Andriluka & Bernt Schiele. *Detection and tracking of occluded people*. International Journal of Computer Vision, vol. 110, no. 1, pages 58–69, 2014.

[Tian 15a] Yonglong Tian, Ping Luo, Xiaogang Wang & Xiaoou Tang. *Deep learning strong parts for pedestrian detection.* In Proceedings of the IEEE international conference on computer vision, pages 1904–1912, 2015.

[Tian 15b] Yonglong Tian, Ping Luo, Xiaogang Wang & Xiaoou Tang. *Pedestrian detection aided by deep learning semantic tasks.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5079–5087, 2015.

[Uijlings 13] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers & Arnold WM Smeulders. *Selective search for object recognition.* International journal of computer vision, vol. 104, no. 2, pages 154–171, 2013.

[Viola 04] Paul Viola & Michael J Jones. *Robust real-time face detection.* International journal of computer vision, vol. 57, no. 2, pages 137–154, 2004.

[Walk 10] Stefan Walk, Nikodem Majer, Konrad Schindler & Bernt Schiele. *New features and insights for pedestrian detection.* In Computer vision and pattern recognition (CVPR), 2010 IEEE conference on, pages 1030–1037. IEEE, 2010.

[Wang 09] Xiaoyu Wang, Tony X Han & Shuicheng Yan. *An HOG-LBP human detector with partial occlusion handling.* In Computer Vision, 2009 IEEE 12th International Conference on, pages 32–39. IEEE, 2009.

[Wang 12] Ningbo Wang, Xiaojin Gong & Jilin Liu. *A new depth descriptor for pedestrian detection in RGB-D images.* In Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012), pages 3688–3691. IEEE, 2012.

[Wang 18] Dong Wang, Lei Zhou, Xueni Zhang, Xiao Bai & Jun Zhou. *Exploring linear relationship in feature map subspace for convnets compression.* arXiv preprint arXiv:1803.05729, 2018.

[Wang 19] Zhixin Wang & Kui Jia. *Frustum ConvNet: Sliding Frustums to Aggregate Local Point-Wise Features for Amodal 3D Object Detection.* In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1742–1749, 2019.

[Wei 19] Jian Wei, Jianhua He, Yi Zhou, Kai Chen, Zuoyin Tang & Zhiliang Xiong. *Enhanced object detection with deep convolutional neural networks for advanced driving assistance*. IEEE Transactions on Intelligent Transportation Systems, vol. 21, no. 4, pages 1572–1583, 2019.

[Wen 16] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen & Hai Li. *Learning structured sparsity in deep neural networks*. In Advances in neural information processing systems, pages 2074–2082, 2016.

[Wojek 11] Christian Wojek, Stefan Walk, Stefan Roth & Bernt Schiele. *Monocular 3D scene understanding with explicit occlusion reasoning*. In Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, pages 1993–2000. IEEE, 2011.

[Wu 11] Shengyin Wu, Shiqi Yu & Wensheng Chen. *An attempt to pedestrian detection in depth images*. In 2011 Third Chinese Conference on Intelligent Visual Surveillance, pages 97–100. IEEE, 2011.

[Xie 18] Xiaolu Xie & Zengfu Wang. *Multi-scale Semantic Segmentation Enriched Features for Pedestrian Detection*. In 2018 24th International Conference on Pattern Recognition (ICPR), pages 2196–2201. IEEE, 2018.

[Yang 15] Bin Yang, Junjie Yan, Zhen Lei & Stan Z Li. *Convolutional channel features*. In Proceedings of the IEEE international conference on computer vision, pages 82–90, 2015.

[Ye 19] Xucheng Ye, J Yang, P Dai*et al.* *Accelerating CNN training by pruning activation gradients*. 2019.

[Yu 18] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin & Larry S Davis. *Nisp: Pruning networks using neuron importance score propagation*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 9194–9203, 2018.

[Yuan 06] Ming Yuan & Yi Lin. *Model selection and estimation in regression with grouped variables*. Journal of the Royal Statistical Society: Series B (Statistical Methodology), vol. 68, no. 1, pages 49–67, 2006.

[Zhang 13] Hao Zhang, Christopher Reardon & Lynne E Parker. *Real-time multiple human perception with color-depth*

cameras on a mobile robot. IEEE Transactions on Cybernetics, vol. 43, no. 5, pages 1429–1441, 2013.

[Zhang 15] Shanshan Zhang, Rodrigo Benenson & Bernt Schiele. *Filtered channel features for pedestrian detection*. In Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on, pages 1751–1760. IEEE, 2015.

[Zhang 16] Liliang Zhang, Liang Lin, Xiaodan Liang & Kaiming He. *Is faster r-cnn doing well for pedestrian detection?* In European conference on computer vision, pages 443–457. Springer, 2016.

[Zhang 17] Shanshan Zhang, Rodrigo Benenson & Bernt Schiele. *Citypersons: A diverse dataset for pedestrian detection*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3213–3221, 2017.

[Zhang 18] Shiquan Zhang, Xu Zhao, Liangji Fang, Haiping Fei & Haitao Song. *Led: Localization-quality estimation embedded detector*. In 2018 25th IEEE International Conference on Image Processing (ICIP), pages 584–588. IEEE, 2018.

[Zhang 19] Shifeng Zhang, Yiliang Xie, Jun Wan, Hansheng Xia, Stan Z Li & Guodong Guo. *Widerperson: A diverse dataset for dense pedestrian detection in the wild*. IEEE Transactions on Multimedia, vol. 22, no. 2, pages 380–393, 2019.

[Zhang 20] Wenli Zhang, Jiaqi Wang, Xiang Guo, Kaizhen Chen & Ning Wang. *Two-Stream RGB-D Human Detection Algorithm Based on RFB Network*. IEEE Access, vol. 8, pages 123175–123181, 2020.

[Zhao 19] Chenglong Zhao, Bingbing Ni, Jian Zhang, Qiwei Zhao, Wenjun Zhang & Qi Tian. *Variational convolutional neural network pruning*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2780–2789, 2019.

[Zhao 20] Chenchen Zhao, Yeqiang Qian & Ming Yang. *Monocular pedestrian orientation estimation based on deep 2D-3D feedforward*. Pattern Recognition, vol. 100, page 107182, 2020.

[Zhou 17] Kaiyang Zhou, Adeline Paiement & Majid Mirmehdi. *Detecting humans in RGB-D data with CNNs.* In 2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA), pages 306–309. IEEE, 2017.

[Zhu 18] Xiatian Zhu, Shaogang Gong*et al. Knowledge distillation by on-the-fly native ensemble.* In Advances in neural information processing systems, pages 7517–7527, 2018.

[Zhu 20] Kuan Zhu, Haiyun Guo, Zhiwei Liu, Ming Tang & Jinqiao Wang. *Identity-Guided Human Semantic Parsing for Person Re-Identification.* In European Conference on Computer Vision. Springer, 2020.

[Zitnick 14] C Lawrence Zitnick & Piotr Dollár. *Edge boxes: Locating object proposals from edges.* In European conference on computer vision, pages 391–405. Springer, 2014.

Occlusion Handling and Multi-modality Fusion for Efficient Pedestrian Detection.