ROBUST DEEP LEARNING FRAMEWORKS FOR RECOGNIZING AND LOCALIZING OBJECTS ACCURATELY AND RELIABLY

by
Zhishuai Zhang

A dissertation submitted to Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy

Baltimore, Maryland
August, 2020

Abstract

Detection is an important task in computer vision. It requires to recognize targets inside images, and localize them. The images can be 2D or 3D, and can be represented by dense pixels or sparse point clouds. With recent emergence and development of deep neural networks, many deep learning based detection frameworks have been proposed. They provide promising performance for many targets, *e.g.* natural objects, object parts, pedestrians and faces, thus are widely used in many applications, including surveillance, autonomous driving and medical image analysis. However, robust object detection is still challenging. Ideal detectors should be able to handle objects with unknown occluders, different scales/movements, long-tailed difficult objects, and low-contrast radiology inputs. Recent detectors are not designed with deliberate consideration of those challenges, and may have degraded performance. In this dissertation, we investigate those challenges, and propose novel detection frameworks to mitigate them.

The aforementioned challenges are addressed in different aspects. (i) We address occlusion by proposing end-to-end voting mechanisms for vehicle part detection. It detects targets by accumulating cues relevant to the target. Occlusions eliminate some of the cues, but remaining cues are still able to detect the targets. (ii) We combine semantic segmentation with object detection, to enrich the detection features in multi-layer single-stage detectors. The enriched features capture both low-level details and high-level semantics, thus the quality of detection is significantly improved for both small and large objects due to stronger detection features. (iii) We investigate the issue of long-tailed hard examples and propose a hard image mining strategy. It dynamically identifies hard images and puts more training efforts during the training phase. This leads to models robust to long-tailed hard examples. (iv) For low-contrast multi-slice medical images, we design hybrid detectors to combine 2D and 3D information. Based on a stack of 2D CNNs for each image slice, we design 3D fusion modules to bridge context information from different 2D CNNs. (v) For objects moving in sequences, we design temporal region proposals to model the movements and interactions of them. We model the moving objects with spatial-temporal-interactive features for detecting them through past, current and future.

Thesis Readers

Dr. Alan L. Yuille (Primary Advisor)
Bloomberg Distinguished Professor
Department of Computer Science
Johns Hopkins University

Dr. Vishal M. Patel
Assistant Professor
Department of Electrical and Computer Engineering
Johns Hopkins University

Dr. Wei Shen
Research Assistant Professor
Department of Computer Science
Johns Hopkins University

Acknowledgements

First and foremost, I would like to thank my advisor Prof. Alan L. Yuille at the Johns Hopkins University, whom I first met in Shanghai when I was a senior undergraduate student. He is a mathematician and computer scientist with great devotion, dedication and love to researching, mentoring and teaching. He always gives us insightful guidance and prompt supports to our research, and teaches us a lot about conducting research, *i.e.*, how to build a big picture of research ideas, how to appreciate long-term bold thoughts and how to set up short-term achievable goals which compose the bigger project. He also helps a lot on our personal and career development especially in this special moment with the pandemic. Alan has also been working hard to enrich and organize the research collaboration and computation resources inside our lab, which greatly expedite our projects. As a result, our group is growing prosperously and has become a great place for learning and researching.

Next, I would like to thank Prof. Wei Shen, who is a great researcher in computer vision. He helps a lot in many aspects of my study and research, *e.g.*, proposing problems and ideas, designing experiments and writing papers. His insights and understanding of researches and literatures have always been an inspiration to me. Besides, I would like to thank Prof. Amitabh Basu, Prof. Alexander S. Szalay, Prof. Vishal M. Patel, Prof. Gregory D. Hager, Prof. Xin Jin for participating in my GBO exam and providing invaluable suggestions regarding my research. In addition to those who directly interact with me on research, I also took several high quality courses during my study at the Johns Hopkins, from Prof. Rene Vidal, Prof. Sanjeev

Khudanpur, Prof. Randal Burns and Prof. David Yarowsky. Thanks to their devotion to teaching, I learnt a lot on topics directly or indirectly related to my research, and those topics and knowledge play an important role in my career. I also want to thank Prof. Haider Ali who provided me an invaluable opportunity as a teaching assistant, which enables me to learn how to convey knowledge in a clear way and teach others.

I am lucky enough to work with a group of talented, hard-working and easy-going collaborators throughout my study, including Adam, Bo, Cihang, Jianyu, Jun, Huiyu, Lingxi, Qing, Siyuan, Song, Vittal, Yan, Yingwei, Yuyin and Zhou. I am also grateful to be able to interact and communicate with and learn from many smart lab mates, including Yongyi, Weichao, Chenxi, Zhuotun, Chenxu, Qi, Fengze, Yi, Hongru, Yingda, Jieru, Qihang, Yixiao, Zhuowan, Zihao, Chenglin, Yutong, Angtian and Chen. The interaction and discussion between us sometimes inspire me of new ideas and thoughts for my research.

During my Ph.D. study, I am fortunate enough to have two great internships in Facebook Applied Machine Learning team and Waymo Perception team respectively. I received mentorship from Xianjie Chen, Yan Zhu and Long Jin at Facebook, and Junhua Mao, Jiyang Gao and Yukai Liu at Waymo. During the internships, they provided invaluable helps on my project, and spent time on discussing the projects, answering my questions, reviewing my codes and designs and proofreading my papers and reports. I learnt from them how to conduct research which involves industrial needs and impacts. I also want to thank Liang Xiong and Congcong Li for hosting my internships in Facebook and Waymo respectively, and my internships would not be possible without their supports.

At last, I would like to thank all the administrative stuffs in the computer science and the Johns Hopkins University. They provide lots of helps on administrations like enrolling courses, organizing GBOs and outlining Ph.D. program guidelines. Their helps make my life and study here much easier as an international student. I would

like to thank Zachary Burwell for monitoring my Ph.D. progress, providing assistants for my out-of-campus internships. I also want to thank Kim Franklin for scheduling my GBOs and answering many questions about Ph.D. requirements. I would like to thank Anthony E. Kolasny and Tony Placilla for organizing computers for our lab. They spend a lot of time on purchasing computers, setting up operating systems and carrying out maintenance. Their support ensures my research and experiments smooth and productive.

To my family and parents for their unreserved and unconditional support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Computer vision is about reasoning the world from low-level raw sensor inputs. The inputs could be in 2D or 3D, in static images or sequential videos, and in dense pixels or sparse point clouds. The outputs are the understanding of the scene demonstrated by the inputs. As summarized by David Marr in a famous quote, the computer vision is about knowing "what is where" [1]. One of the most intuitive and important outputs of the reasoning is the detection of objects, which is to recognize and localize objects in the scene, and describes what and where any pre-defined semantically meaningful objects exist in the scene.

An important aspect of parsing and reasoning the scene demonstrated by the inputs is to perceive what types of objects exist in the scene and where are the instances of the objects of interest. It serves as a prerequisites for many higher level vision tasks, like object tracking, face alignment, *etc.*, and it is also important for many vision applications. For example, in an autonomous driving scenario, the self-driving car (SDC) needs to understand whether and where other road users or traffic signs exist. SDC makes driving control decisions based on traffic signs, and reacts to other road users like pedestrian, cars and cyclists. For this goal, detecting them (both traffic signs and other road users) would be a critical step, as illustrated in Figure

[1]What does it mean, to see? The plain man's answer (and Aristotle's, too) would be, to know what is where by looking. In other words, vision is the process of discovering from images what is present in the world, and where it is. – David Marr [1]

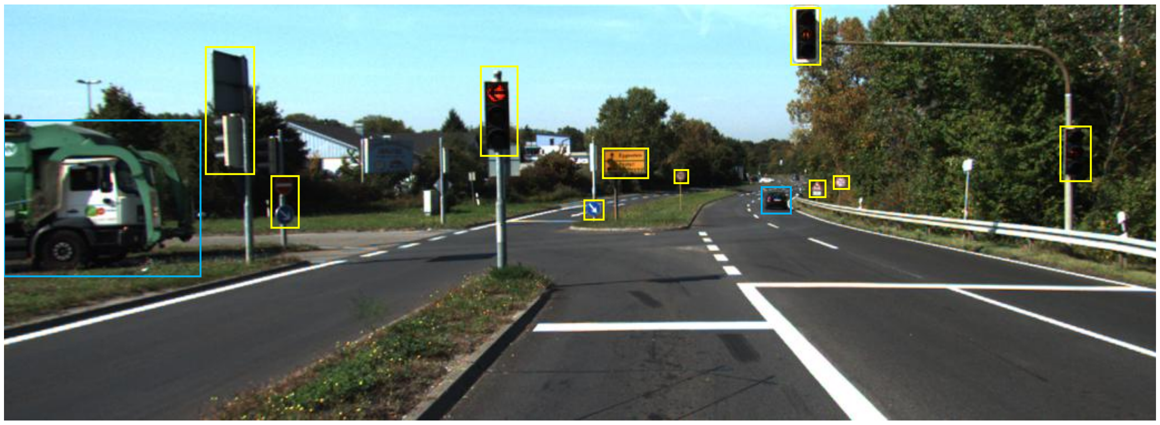

Figure 1.1. An example of a driving scenario provided in Kitti dataset [2]. Self-driving car needs to identify and locate the traffic signs (in yellow boxes) and other road users (in blue boxes) in order to proceed safely. Further processes, including text recognition, vehicle intent prediction, *etc*., take object detection as their backbones.

1.1. Detection provides the SDC information about types and positions of objects around it to make decisions, and also enables further detailed analysis of detected objects (*e.g.* pose, gesture, speed and text recognition/estimation). Another use case of object detection is for taking photos or videos with a camera. As humans are one of the most important subjects in photographies, face-priority auto focus has been developed to build better photos with human faces, by locating faces and providing appropriate focus and exposure. Recently, on powerful cell phones, more complicated face beautifying mechanisms can be applied after detecting the faces.

Detection is about identifying and locating objects of interest in a scene, and it can have different input/output forms for each specific application. The types of objects of interest can be one specific category (*e.g.* face detection or pedestrian detection) or can be a broad range of categories (*e.g.* 20 or 80 pre-defined object categories in Pascal VOC [7] and MS COCO [3] dataset respectively). The location of objects of interest is described by bounding-boxes in most cases due to the simplicity of this representation. Depending on the task, the bounding-boxes could be in 2D or 3D, and they could be without rotation (*i.e.* axis-aligned boxes) or with rotations (*i.e.*, boxes with yaw, roll and pitch). In Figure 1.2, we show some examples of detection tasks based on

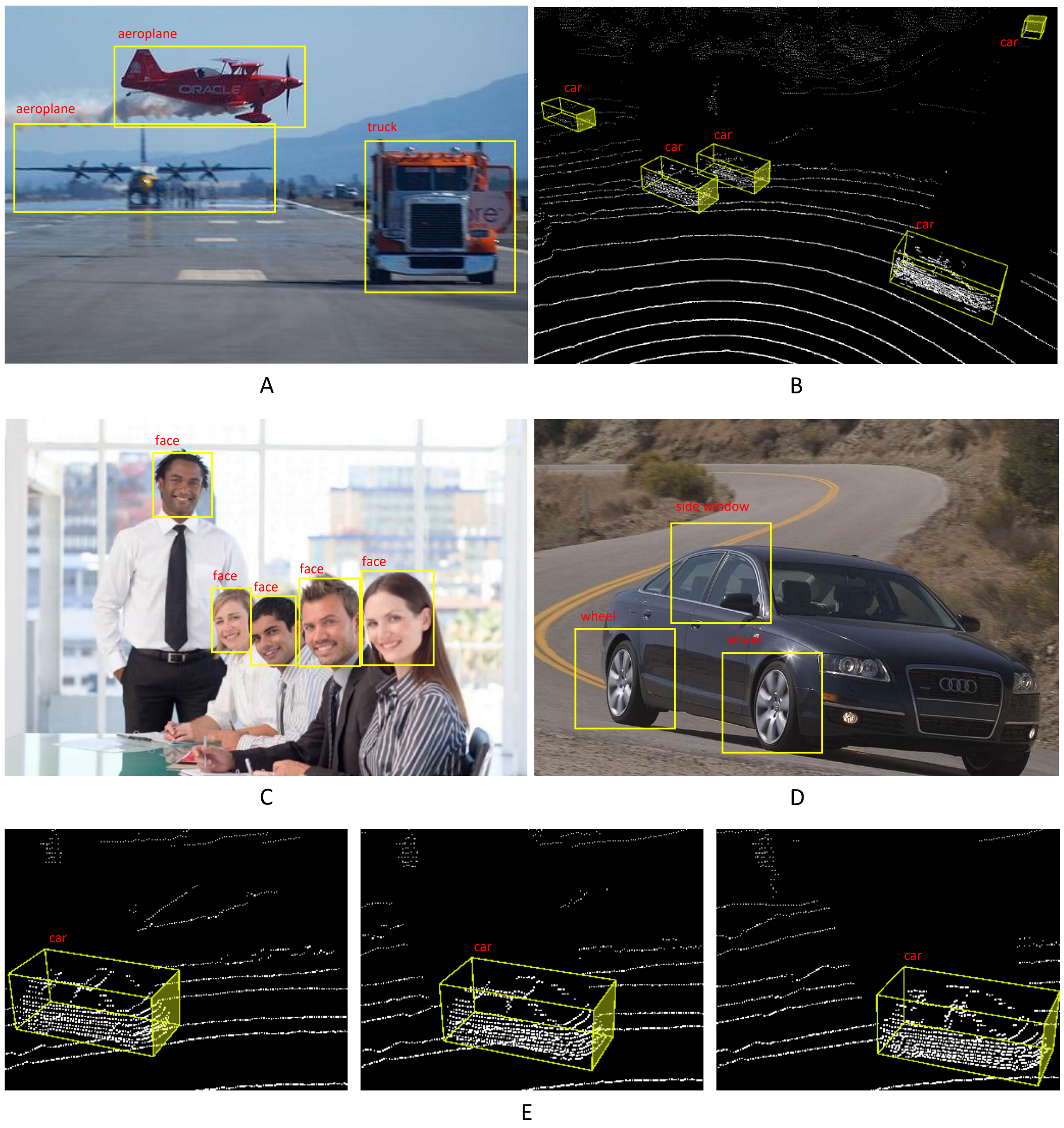

Figure 1.2. Examples of five different detection tasks. From upper-left to lower-right in row-major order: A) natural object detection from a 2D image (MS COCO dataset [3]); B) car detection from 3D point clouds (Lyft dataset [4]); C) face detection from a 2D image (WiderFace dataset [5]); D) car part detection from a 2D image (VehicleSemanticPart dataset [6]); E) car detection from a sequence of 3D point clouds with 3 frames (Lyft dataset [4]). Object types are annotated in red text and object locations are annotated in yellow 2D or 3D bounding-boxes.

different inputs/outputs specifications. Regarding types of inputs, examples A, C and D take static camera image as input, example B takes static point clouds as input and example E takes a sequence of point clouds with 3 frames as input. Regarding types of objects of interest, example A and D consider a broad range of categories (*i.e.* natural objects and car parts respectively) and example B, C and E consider a specific category (*i.e.* car, face and car respectively).

Given the recent emergence and rapid development of deep learning and convolutional neural networks, as well as the accessibility of large scale datasets and powerful GPU accelerators, the ability of parsing and understanding images and videos has been greatly boosted. Many fundamental vision tasks, including image classification [8]–[11] and image segmentation [12]–[14], have been shown able to get tackled by deep learning. Detection is also largely improved by deep neural networks and modern detection frameworks, for numerous applications. We will discuss some important research literatures about detection in Chapter 2.

Although object detection has achieved promising performance recently with rapid development, there is still a gap towards human-level performance. Detection is a challenging task with many difficulties, and can be improved in many aspects, which will be addressed in this dissertation. The difficulties lie in the diversity of object appearances and properties, challenging image patch qualities (*e.g.* small scale, blurred image and poor lighting conditions), as well as complex contexts and occlusions. In this dissertation, we focus on improving the robustness of object detection with those challenges. We explore object detection on a broad range of domains, including detecting object semantic parts [6], general objects [3], [7], faces [5], lesions [15] and pedestrians [4], [16]. These domains are important applications of object detection. They have different types of sensor inputs (*e.g.* camera, computed tomography and lidar) and different dimensions (*e.g.* 2D, 3D and 3D+time), and there are different types of challenges associated with them. We will demonstrate that by carefully designing the architectures and the training mechanisms, these tasks can be addressed with good efficacy and efficiency.

1.1 Challenges and Our Contributions

1.1.1 Occlusion with Unknown Occluders

For real-world applications, objects are frequently occluded, either partially or fully, which make the detection much harder. Detection under occlusion is an important yet under-examined task. Traditional object detectors are shown to be vulnerable to occlusions, as the occluder and occlusion patterns may not be seen during the training and may confuse the detectors if they are simply 'memorizing' what are seen during the training. There are a few research efforts addressing this issue. Wang *et al.* proposed VehicleSemanticPart [6], for the task of semantic part detection. It is one of the few datasets systematically providing objects occluded by irrelevant occluders at different pre-defined occlusion ratios. As illustrated in Figure 1.3, a bus presents in the image, and it is occluded by a few irrelevant occluders (*e.g.* cat). From upper-left to lower-right in row-major order, the bus is occluded in four different configurations (L0 to L3) by 0, 2, 3 and 4 occluders, and the occlusion ratios of the object, computed by pixels are 0.0, 0.2–0.4, 0.4–0.6 and 0.6–0.8, respectively. A traditional detector shows significant performance drop with more occlusions, with an AP of 73.6 dropped to 23.0 (a 69% drop) under occlusions.

An ideal detector should be able to recognize and locate the pre-defined objects of interest regardless of the irrelevant occluders, because the occlusion problem is inevitable and frequent in the real-world applications. A detector is less useful if it cannot handle the occlusion. As we cannot know what and where the occluders will be before actually seeing them during the testing, we believe that the deep learning models trained with images without occlusion should be able to transfer to occlusion cases during the inference. It avoids exhaustively collecting every possible occlusion patterns for the training, which is almost infeasible. In this dissertation, we propose a novel object detection framework which could transfer to occluded testing examples

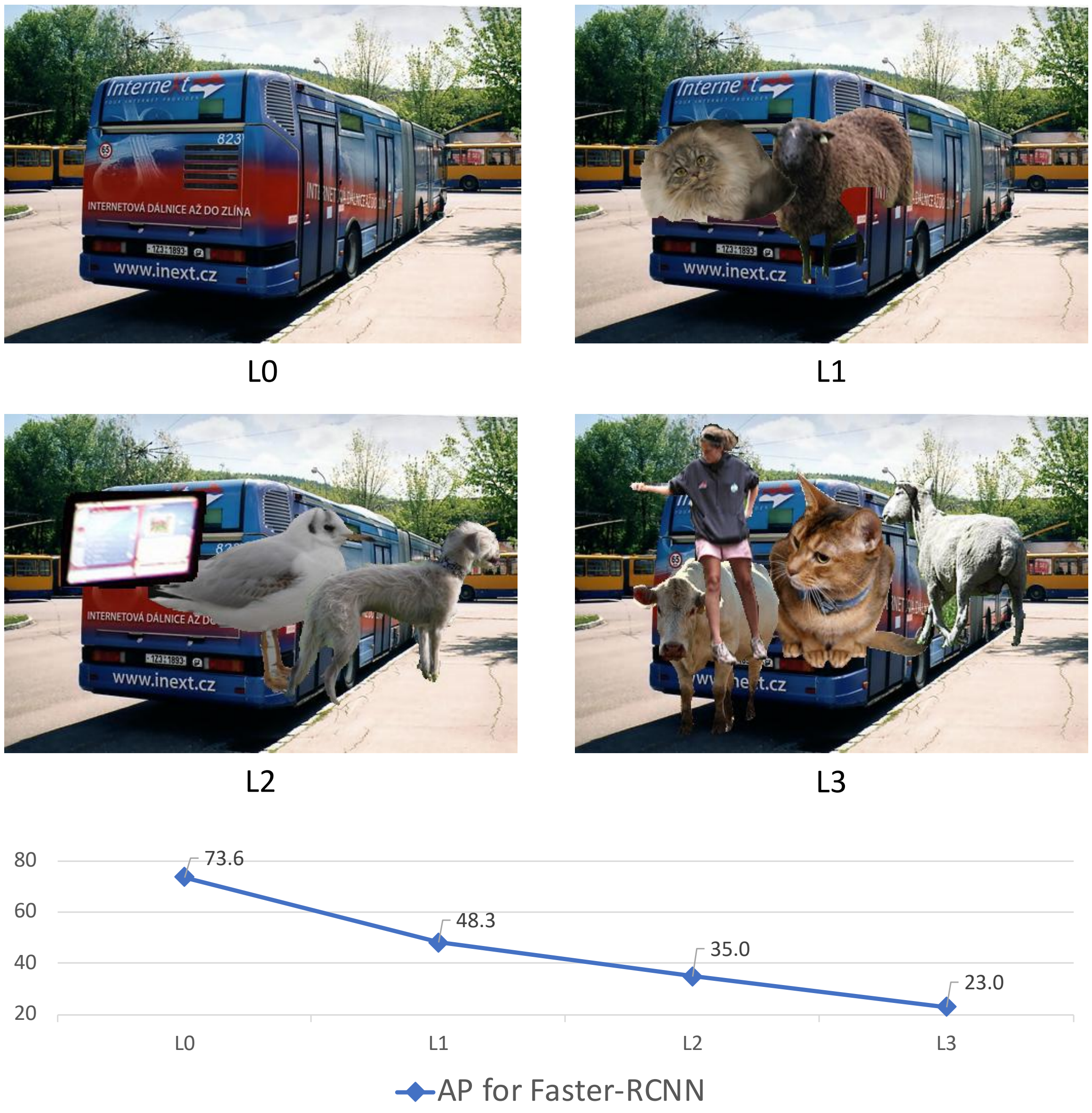

Figure 1.3. Examples of images in the VehicleSemanticPart dataset (first 4 pictures) and performance of a traditional detector (Faster-RCNN) in a line chart. The first image (L0) of a bus is unoccluded. The next three images (L1, L2 and L3) are the same scene with different occlusions. There are 2, 3 and 4 occluders, and the occluded ratios of object, computed by pixels, are 0.2–0.4, 0.4–0.6 and 0.6–0.8, respectively. The line chart shows the detection performance of Faster-RCNN on the whole `test` subset with different occlusion levels, and the performance drops drastically by 69%.

while being trained on unoccluded images, by learning mid-level visual cues as well as their relationships to the detection targets. During the inference, the the visible cues can be accumulated to detect targets under occlusion.

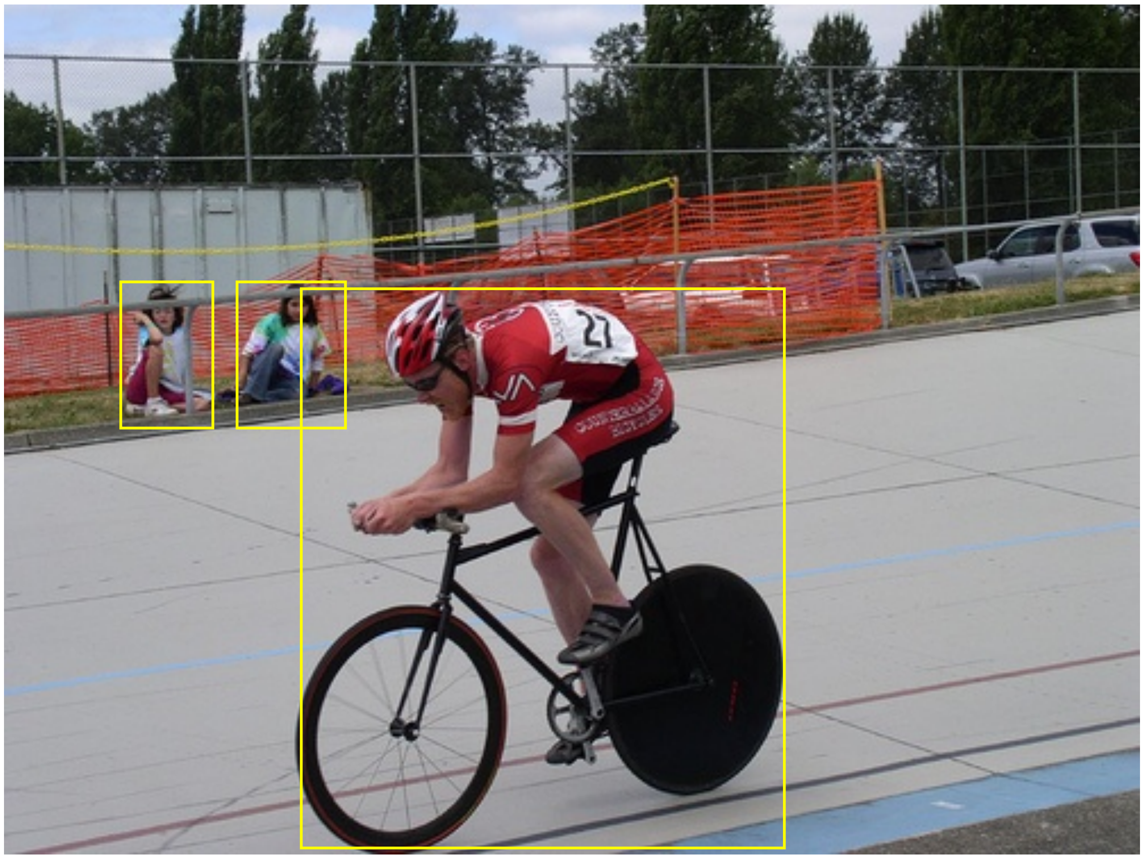

Figure 1.4. An illustration of the scale variance among objects. The cyclist closer to the camera is much larger (around 120 times larger in terms of the number of pixels) than the two pedestrians in the background far away from the camera.

1.1.2 Scale Variance

General object detection requires detectors to be capable of dealing with objects with different scales (*i.e.* sizes), which mainly come from the object size difference and the object-camera distance difference. We illustrate an example in Figure 1.4, where the cyclist closer to the camera is approximately 120 times larger than the pedestrians far away from the camera sitting on the side of the road. The detectors are supposed to detect both small and large objects accurately, and preferably simultaneously. One natural way to achieve that is to resize the input image multiple times with different resizing factors to build an image pyramid and to conduct scale-specific detection on each image of the image pyramid, but this method requires more computation costs and takes longer time.

On the other hand, a multi-layer design has been proposed with a feature pyramid, where low-level features are used for small object detection and high-level features are responsible for large object detection, and it is becoming more and more prevalent [17]. However, this strategy leads to an imbalance of the capability of detecting objects with different scales, as different features have different semantic types. As pointed out by [18], "activation regions (of CNN neurons/units) tended to become more semantically meaningful with increasing depth of layers". The low-level features have smaller receptive field size, and usually only capture basic visual patterns without strong semantic information. This may cause two problems: small objects may not be detected well, and the quality of high-level features is also damaged by the imperfect low-level features. In this dissertation, we follow the multi-layer design and propose a novel mechanism to address the aforementioned issue. We combine object detection with semantic segmentation by exploiting a segmentation branch in parallel, to improve the quality of the backbone features by enriching the segmentation information into the detection features. The enriched detection features contain strong semantics relating to what objects may exist, in addition to low-level knowledge about colors, shapes and textures. Our novel mechanism improves the detection for objects with all scales with a large margin, especially for small scales.

1.1.3 Long-tailed Hard Examples

Comparing with general objects, detections of objects with a specific type (*e.g.* face) are less complex and achieving better performance. But there is still a performance gap between the machine learning algorithms and human's ability, and detectors may still fail on some hard examples, especially when those hard examples are less frequently happening in the training dataset. The failure on those hard examples make the detectors less reliable for practical use. In this dissertation, we focus on the task of face detection, which is very crucial and popular in both academia and industry. Recently deep learning based face detectors can achieve very high performance even on the most challenging dataset WiderFace [5]. However, an evident performance gap still exists especially for those hard images which contain small, blurred and partially occluded faces. We realize that these long-tailed hard images have become the main barriers for face detectors to achieve human-level performance, and are not paid enough attention during the training, as most training images are less challenging. To this end, we propose a novel training mechanism to encourage the deep neural network to focus more on those long-tailed hard images while keeping the general detection performance unaffected. During the training, each training image is assigned a difficulty score which determines how well it is detected by the current model snapshot. During the remaining epochs of the training phase, the images will be re-weighted according to their difficulty scores, and some images will receive more training attentions if the network finds difficult of detecting them image. Thus the training process automatically adjusts itself by learning more on the hard examples, and the long-tailed hard examples can have higher frequency/weight of being trained. Our proposed detector is shown to detect hard testing images well and is faster compared with other state-of-the-arts.

1.1.4 Low Contrast Computed Tomography 3D Scan Images

Unlike natural images, computed tomography scan images have their own challenges of low contrast of soft tissues and similar appearance between foreground and background, as well as their own opportunities with 3D context slices available, as shown in one example of the DeepLesion dataset illustrated in Figure 1.5. Exploiting 3D context information has been shown to improve detection accuracy, and in this dissertation, we investigate and propose a novel and efficient way to fuse 3D context information from neighboring slices with small computation and memory overhead. Directly using

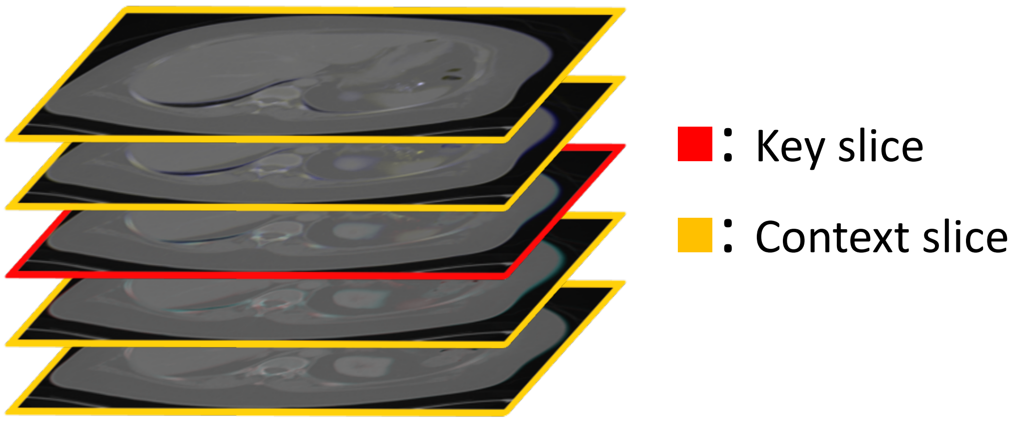

Figure 1.5. Examples of CT scan images for lesion detection in the DeepLesion dataset [15]. The inputs are composed of 2D image slices and the bounding-boxes are labeled on the key slices (marked in red), and the neighboring slices (marked in yellow) are used to provide 3D context.

fully 3D-connected CNNs could not take use of ImageNet pre-trained weights, and also leads to more parameters and computations. On the other hand, we use 2D CNNs initialized with pre-trained weights as our backbones, and design lightweight 3D fusion modules to connect the 2D backbone CNNs and fuse the 3D context knowledge. The 3D fusion modules are inserted at middle and high levels of the backbone CNNs, so that the 3D information is exploited and learnt gradually throughout our backbone CNNs. The proposed method introduces few extra parameters and small computation overhead, while greatly improve the detection accuracy.

1.1.5 Dynamic Objects with Different Moving Patterns

Detecting what are there in static images may not be enough, since everything comes in continuous sequences for the real-world problems. Autonomous driving is such an application with sequence of inputs naturally, as the self-driving car continuously captures its surroundings for analysis. In order to get a richer understanding of the scene and drive safely, it is important to model objects in a sequence (*e.g.* a video), and detect them in the sequence. In addition to detecting object in the current/newest frame, it is crucial to detect them in the future frames, *i.e.*, trajectory prediction, for these safety critical applications. In this dissertation, we aim at tackling the problem of detecting pedestrians throughout the past, the present and the future frames. The inputs are sequences of point clouds collected from self-driving vehicles, and the output is the bounding-boxes of pedestrians at the current frame, as well as their positions for the next few frames in the future, which are predicted based on their history and current locations.

We observed that there are two major challenges that are critical for the proposed task. Firstly, fine-grained and flexible temporal information for each object is necessary to better model objects with different moving patterns (*e.g.* stationary and fast-moving), and a traditional neural network is not capable of modeling the complex dynamics. Secondly, interactions among objects are also useful since the location and trajectory of an object could be influenced by the other objects. To this end, we propose an end-to-end Spatio-Temporal-Interactive network (STINet) to model pedestrians' static, temporal and interactive information jointly, by first predicting temporal proposal with multiple bounding-boxes covering the past and current frames, and then extracting comprehensive proposal features capturing spatio, temporal and interactive knowledge for final prediction. Experiments on two autonomous driving datasets demonstrate the effectiveness of our method.

1.2 Thesis Statement

Ideal detectors should be able to handle different challenges existing in different tasks. By carefully designing network architectures and training strategies, we can build robust and reliable object detectors for a board range of applications.

1.3 Outline

The outline of this dissertation is illustrated as the following:

In Chapter 1 (this chapter), we introduce the topic of this dissertation, including definition and justification. We discuss the challenges and our contributions regarding our dissertation topic.

In Chapter 2, we discuss previous works on object detection for different targets and applications.

In Chapter 3, we propose a novel voting detection mechanism for robust detection under unknown occlusions.

In Chapter 4, we discuss detection with enriched semantic by combining multi-layer scale-invariant detectors with semantic segmentation.

In Chapter 5, we propose an online hard image mining training strategy for detecting long-tailed hard examples.

In Chapter 6, we build a 2D/3D hybrid network for exploiting 3D context in low contrast computed tomography scan images.

In Chapter 7, we design a spatio-temporal-interactive network to model and detect moving pedestrian in point cloud sequences.

1.4 Relevant Publications

The following publications compose the ideas in this dissertation. * indicates equal contribution.

1. **Z. Zhang***, C. Xie*, J. Wang*, L. Xie, A. Yuille, DeepVoting: A Robust and Explainable Deep Network for Semantic Part Detection under Partial Occlusion, *CVPR 2018*

2. **Z. Zhang**, S. Qiao, C. Xie, W. Shen, B. Wang, A. Yuille, Single-Shot Object

Detection with Enriched Semantics, *CVPR 2018*

3. **Z. Zhang**, W. Shen, S. Qiao, Y. Wang, B. Wang, A. Yuille, Robust Face Detection via Learning Small Faces on Hard Images, *WACV 2020*

4. **Z. Zhang**, Y. Zhou, W. Shen, E. Fishman, A. Yuille, Lesion Detection by Efficiently Bridging 3D Context, *MLMI 2019*

5. **Z. Zhang**, J. Gao, J. Mao, Y. Liu, D. Anguelov, C. Li, STINet: Spatio-Temporal-Interactive Network for Pedestrian Detection and Trajectory Prediction, *CVPR 2020*

The following publications indirectly contribute to, or provide contexts and backgrounds for the ideas in this dissertation. * indicates equal contribution.

1. J. Wang*, C. Xie*, **Z. Zhang***, J. Zhu, L. Xie, A. Yuille, Detecting Semantic Parts on Partially Occluded Objects, *BMVC 2017*

2. J. Wang, **Z. Zhang**, C. Xie, Y. Zhou, V. Premachandran, J. Zhu, L. Xie, A. Yuille, Visual concepts and compositional voting, *Annals of Mathematical Sciences and Applications*

3. A. Kortylewski, Q. Liu, H. Wang, **Z. Zhang**, A. Yuille, Localizing Occluders with Compositional Convolutional Networks, *ICCVW 2019*

4. A. Kortylewski, Q. Liu, H. Wang, **Z. Zhang**, A. Yuille, Compositional Convolutional Networks For Robust Object Classification under Occlusion, *WACV 2020*

Below lists other publications and researches I involved in during my Ph.D. study. * indicates equal contribution.

1. C. Xie*, J. Wang*, **Z. Zhang***, J. Zhu, L. Xie, A. Yuille, Adversarial Examples for Semantic Segmentation and Object Detection, *ICCV 2017*

2. C. Xie, J. Wang, **Z. Zhang**, Z. Ren, A. Yuille, Mitigating Adversarial Effects Through Randomization, *ICLR 2018*

3. S. Qiao, **Z. Zhang**, W. Shen, B. Wang, A. Yuille, Gradually Updated Neural Networks for Large-Scale Image Recognition, *ICML 2018*

4. S. Qiao, W. Shen, **Z. Zhang**, B. Wang, A. Yuille, Deep Co-Training for Semi-Supervised Image Recognition, *ECCV 2018*

5. C. Xie, **Z. Zhang**, Y. Zhou, S. Bai, J. Wang, Z. Ren, A. Yuille, Improving Transferability of Adversarial Examples with Input Diversity, *CVPR 2019*

6. Y. Zhou, Y. Li, **Z. Zhang**, Y. Wang, A. Wang, E. Fishman, A. Yuille, S. Park, Hyper-pairing Network for Multi-phase Pancreatic Ductal Adenocarcinoma Segmentation, *MICCAI 2019*

7. Y. Zhou, D. Dreizin, Y. Li, Z. Zhang, Y. Wang, A. Yuille, Multi-Scale Attentional Network for Multi-focal Segmentation of Active Bleed after Pelvic Fractures, *MLMI 2019*

8. Y. Li, S. Bai, Y. Zhou, C. Xie, **Z. Zhang**, A. Yuille, Learning Transferable Adversarial Examples via Ghost Networks, *AAAI 2020*

Chapter 2

Related Work

In this chapter, we discuss research literatures related to this dissertation. We first discuss general approaches for object detection in Section 2.1. Then we will discuss more specific detection tasks for face detection (Section 2.2), lesion detection in medical images (Section 2.3), as well as object detection in point cloud sequences (Section 2.4).

2.1 General Object Detection

General object detection is to detect a broad range of natural objects, and some representative 2D image datasets include Pascal VOC [7] (20 classes) and MS COCO [3] (80 classes). The outputs of detection are 2D axis-aligned bounding-boxes with their labels and confidence scores. To be more specific, an output is a tuple of $(x_{min}, x_{max}, y_{min}, y_{max}, label, score)$, where the first four elements define the location of the output bounding-box, label indicates the category of the object in the bounding-box and score represents the confidence score of the prediction. The evaluation is usually carried out for each category separately by computing the average precision (AP) of each category, and the overall performance is measured by the mean of average precisions on different categories, noted as mAP. Considering a category, given a score threshold, a prediction tuple is regarded as a true positive if all the requirements are met: 1) its confidence score is higher than the score threshold; 2) it can be matched

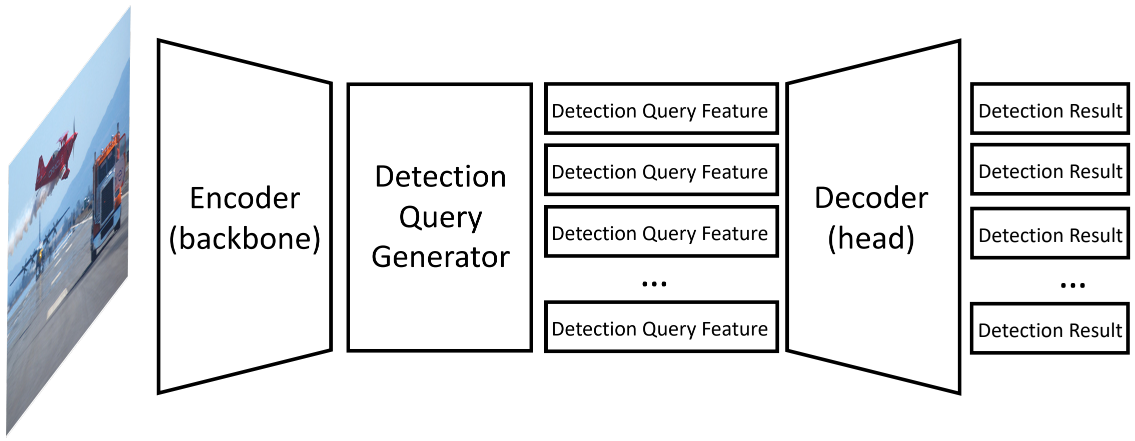

Figure 2.1. General detectors, which take images as input. The feature encoders (also know as backbones) process the input image and generate features for detection queries. The detection decoders (*i.e.* detection heads) compute the final prediction output based on the query and the feature vector. Detection queries are the prior locations to check whether any object exists, and they can be proposals [19], [20], pre-defined anchors [17], grids [21] or corner point pairs [22].

to a ground-truth object, with an Intersection over Union larger than a threshold; 3) the matched ground-truth object is not matched by any prediction tuple with higher confidence scores (otherwise the current prediction tuple is regarded as a duplicated detection). The precision is the ratio of true positive prediction tuples among all prediction tuples with confidence scores above the score threshold. The average precision is defined to be the mean precision on different recall rates, controlled by varying the score threshold.

General object detection is a fundamental task in computer vision and has received extensive research attention [23]–[25]. With the development of deep neural network, almost all the recent object detectors are CNN based. Generally, in an abstract sense, typical object detectors are composed by three components: feature encoders, detection queries and detection decoders as illustrated in Figure 2.1. Feature encoders (*i.e.* detection backbones) take the image as input and generate feature vectors for each of the detection query, and detection decoders (*i.e.* detection heads) generate output tuples by decoding the feature vector for the corresponding detection query.

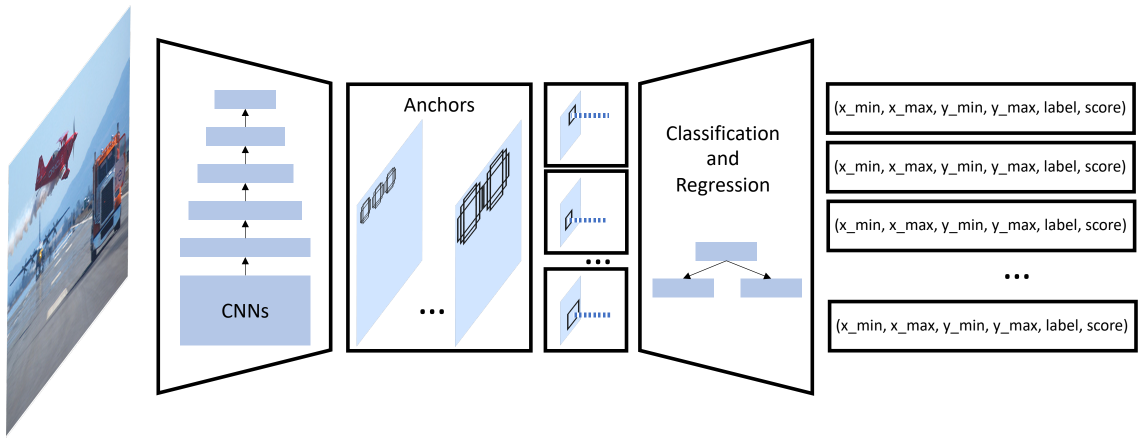
CNNs
Anchors
...
Classification and Regression
(x_min, x_max, y_min, y_max, label, score)
(x_min, x_max, y_min, y_max, label, score)
(x_min, x_max, y_min, y_max, label, score)
...
(x_min, x_max, y_min, y_max, label, score)

Figure 2.2. Overall framework of SSD [17]. The encoder is a multi-layer convolutional neural network, and generates a pyramid of detection feature maps, with different resolutions and receptive field sizes. Detection queries are pre-defined anchors assigned to grid positions of one of the feature maps. The decoder takes the anchor as prior and use the anchor feature vector to conduct classification and regression to generate the final detection tuple results.

Detection queries are a set of queries with prior locations to check whether any object of interest exists around the given locations. They can be formulated by different forms, *e.g.*, ad-hoc rectangular proposals [26], pre-defined anchor boxes [17], or corner point pairs [22].

One prevalent series of detectors is the single-stage detectors, such as OverFeat [27], SSD [17] and YOLO [21]. These detectors detect objects by classifying and regressing pre-defined detection queries in a dense manner. Among them, SSD is a prevalent work which is both fast and scale invariant. As shown in Figure 2.2, it deploys a multi-layer backbone feature encoder, to built a sequence of detection feature maps in a bottom-up manner. Those feature maps have decreasing resolutions, and increasing receptive field sizes. The detection queries are drawn from a set of pre-defined dense anchors with different scales, aspect ratios and locations, and each of them is attached to a point to one of the encoder feature maps. Smaller anchors are attached to the lower-level feature maps while larger anchors are attached to the higher-level feature maps, in order to align with the increasing receptive field size. The query (*i.e.* anchor)

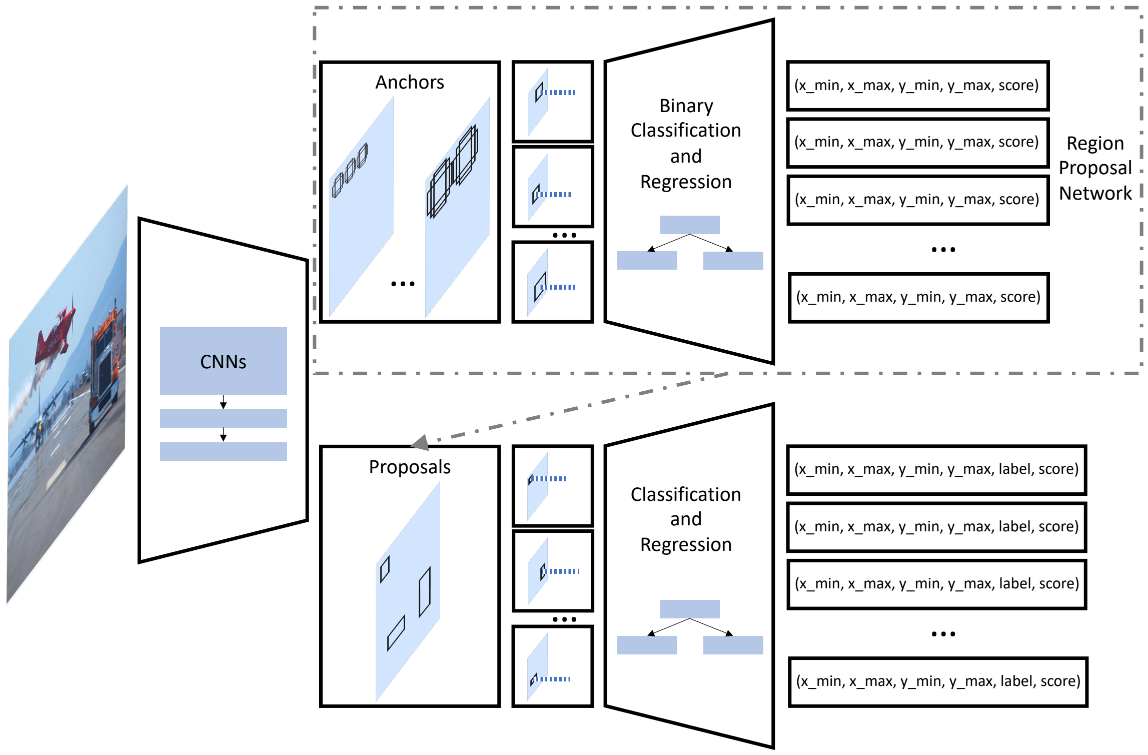
Anchors
Binary Classification and Regression
(x_min, x_max, y_min, y_max, score)
Region Proposal Network
CNNs
Proposals
Classification and Regression
(x_min, x_max, y_min, y_max, label, score)

Figure 2.3. Overall framework of Faster-RCNN [26]. The framework is composed of one feature encoder and two sets of query generators and decoders. The feature encoder is a fully-convolutional neural network and generates two feature map, one for each set of query generator and decoder respectively. The first set of query generator and decoder is also known as region proposal network, which takes pre-defined anchors as input and generate class-agnostic bounding-box candidates as proposals. The second set of query generator and decoder takes the proposals as input, extract proposal features from the backbone feature map and compute final detection outputs.

feature vectors are simply extracted from the corresponding locations at the attached feature map. Finally, decoders take the prior anchors as well as the query feature vectors to conduct classification and regression, generating the final detection outputs.

Another slightly more complicated series of detectors is the two-stage detectors. Some representative works are R-CNN [19], Fast-RCNN [20], Faster-RCNN [26] and R-FCN [28]. These methods first generate a pool of object candidates, named object proposals, by a separate proposal generator such as Selective Search [29], Edge Boxes [30] or Region Proposal Network (RPN), and then they conduct per-proposal classification and bounding box regression. We show the architecture of Faster-RCNN in Figure 2.3 as an example of these works. A convolutional neural network is used as the feature encoder, which outputs two feature maps for region proposal network and final detection network respectively. There are two sets of query generators and decoders. The first set is also known as region proposal network, which takes pre-defined anchors as queries and retrieves feature vectors for every anchor from the first encoder feature map. A set of class-agnostic bounding-box candidates are computed by the first decoder via foreground/background classification and bounding-box regression. The second query generator takes these bounding-box candidates as input, and generates proposals as new queries after thresholding and non-maximum suppression. The feature vectors for these proposals are cropped from the second encoder feature map by a pooling operation. The second decoder carries out label classification and bounding-box regression to generate the final detection outputs.

More recently, a new kind of detectors without pre-defined anchors or proposals are proposed, of which CornerNet [22] is a representative work. CornerNet also deploys a fully-convolutional feature encoder to build a feature map for the input image. However, instead of generating anchors as detection queries, it generates point pairs as detection queries, by regrading each point pair to be the upper-left and lower-right corner point of a bounding-box. The decoder computes the label and confidence score of each point pair by computing the cornerness score of the upper-left and lower-right corner point, as well as comparing the feature vector embedding similarity between the two corner points. As a follow-up, CenterNet [31] is proposed, and it detects objects by point triplets instead of point pairs, by also considering the center points of bounding-boxes.

2.2 Face Detection

With the huge development on object detection for general objects on natural images, the detection for specialized objects with deep neural networks has raised lots of interests. Among them, face detection is a well-defined problem and has important real-world applications. SSH [32] and S3FD [33] are two representative works for face detection, and both of them are adapted from general object detection frameworks (*i.e.* SSD and the region proposal network of Faster-RCNN). SSH builds a feature encoder with three detection feature maps as output, and takes pre-defined anchors as the detection queries. In order to increase the effective receptive field size of the detection feature maps, a simple context module is proposed in the feature encoder. The anchors have aspect ratio of 1:1 and have six different scales. The three feature maps correspond to anchors with scale $\{1, 2\}$, $\{4, 8\}$ and $\{16, 32\}$ respectively where the base anchor size is 16×16 pixels. S3FD [33] has a similar design as SSH. It differs in the feature encoder by having six detection feature maps. It also improves face detection by introducing a scale compensation anchor matching strategy and a max-out background label strategy for the training phase. More recently, PyramidBox [34] utilizes better contextual information by introducing new anchor matching mechanism, feature pyramid for backbone encoders and context-sensitive modules. [35] introduces a two-step classification and regression pipeline by conducting easy negative example filtering and anchor adjustment before generating final predictions.

2.3 Lesion Detection in Medical Images

Applying machine learning and computer vision methods on medical images has been attracting more and more research interests, and lesion detection is an important application in computer-aided medical diagnosis. Similar detection frameworks are applied on computed tomography images, which have some distinct properties compared

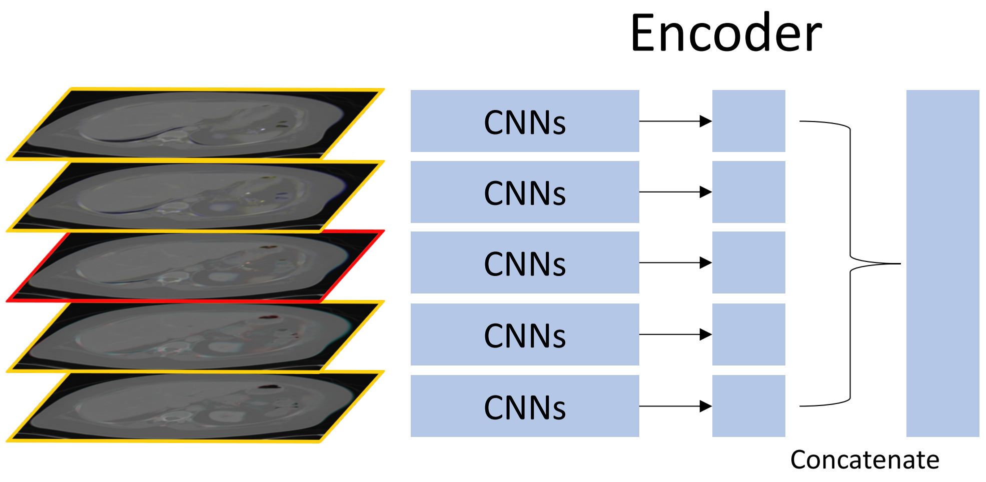

Figure 2.4. Feature encoder of 3DCE. It takes multiple slices as inputs, and separate 2D CNNs are used to extract feature maps, which are later aggregated for final prediction.

with natural 2D camera images, such as low contrast and 3D context. Exploiting 3D information for better detection has been tried in related researches. 3DCE [36], as a follow-up of R-FCN [28], takes a two-stage design. It deploys a Region Proposal Network (RPN) to predict suspicious lesion regions as proposals and a Region Classification Network (RCN) to further classify and regress those suspicious lesion regions. In order to take 3D context information into consideration, [36] proposes 3D context enhanced region-based CNNs. As illustrated in Figure 2.4, the encoder takes multiple neighboring slices as inputs, and 2D backbone CNNs generate feature maps separately for each input slice. Then the feature maps are aggregated by concatenation, for extracting proposal features and making final predictions.

2.4 Detection from Sparse Point Clouds

As a direct application of object detection, autonomous driving is a topic with extensive academic and industrial attentions. In addition to traditional camera images, point clouds obtained from LiDARs have been an important component of the perception systems, with the ability to accurately measure depth. However, point clouds cannot directly be processed by CNNs due to its sparsity and irregularity. PointNet [37] proposes a novel mechanism to handle point clouds without voxelization by adopting symmetric functions (*e.g.* max pooling) over the point features learnt by multi-layer perceptions. Based on this direction, VoteNet [38] is proposed to detect objects from point clouds in a Hough Voting manner, and ImVoteNet [39] proposes to combine image information with point clouds for detection. StarNet [40] uses sparsely sampled locations to extract a small set of points for object detection via a PointNet-style network.

Another direction is based on voxelization. VoxelNet [41] proposes to divide a point cloud scene into equally spaced 3D voxels, and learn the voxel feature with small PointNet-style networks. The point cloud scene is transferred to a 3D pseudo-image in this way, and thus traditional object detection methods can be directly applied. SECOND [42] provides a series of improvements based on VoxelNet, including sparse convolution, angle loss regression and new data augmentation approaches. PointPillars [43] divides the point cloud scene into 2D pillars instead of 3D voxels, and transfers the point clouds into a 2D pseudo-image to achieve a faster speed. MVF [44] synergizes the birds-eye view and the perspective view representation of point clouds together to learn better feature encoder with complementary information from both views.

It is natural to exploit the temporal information for autonomous driving where inputs are captured in a sequence. Temporal context is not only helpful for detecting objects, but also critical for forecasting their trajectories, *i.e.*, detecting objects in the future. StarNet explores to use the outputs of previous frames to improve the detection. FaF [45] performs 3D convolutions on multi-frame inputs for joint detection, tracking and trajectory forecasting. IntentNet [46] further considers high level behavior intents, in addition to future trajectories, as model supervision and outputs.

Chapter 3

DeepVoting for Semantic Part Detection under Partial Occlusion

In this chapter, we study the task of detection under partial occlusion, which is challenging and inevitable for real-world applications. We focus on the VehicleSemanticPart dataset and its occluded counterpart (we call it VehicleOcclusion dataset) proposed in [6]. They are built for systematically analyzing the occlusion issue, and with a goal of detecting semantic parts of vehicle objects, *e.g.*, a wheel of a car, under partial occlusion. We argue that all detection models should be trained without seeing occlusions while still being able to transfer the learned knowledge to deal with testing cases with occlusions. This setting avoids the difficulty of collecting an exponentially large dataset to cover arbitrary occlusion patterns, which is infeasible. In this scenario, the proposal-based deep networks, like RCNN-series, often produce unsatisfactory results, because both the proposal extraction and classification stages may be confused by the irrelevant occluders which are unseen during the training phase. To address this, Wang *et al.* [6] proposed a voting mechanism that combines multiple local visual cues to detect semantic parts. The semantic parts can still be detected even though some visual cues are missing due to occlusions. However, this method is manually-designed, thus is hard to be optimized in an end-to-end manner. In this chapter, we present DeepVoting, which incorporates the robustness shown by Wang *et al.* [6] into a deep network, so that the whole pipeline can be jointly optimized. Specifically, it adds two layers after the intermediate features of a deep network, *e.g.*, the *pool-4* layer of VGGNet. The first layer extracts the evidence of local visual cues, and the second layer performs a voting mechanism by learning and utilizing the spatial relationship between semantic parts and their supporting visual cues. We also propose an improved version named DeepVoting+, by adding visual cues from context outside objects. Shown by the experiments, DeepVoting achieves significantly better performance than several baseline methods, including Faster-RCNN, for semantic part detection under occlusion. In addition, DeepVoting enjoys explainability as the detection results can be diagnosed via looking up the voting cues.

3.1 Motivation and Overview

Deep networks have been successfully applied to a wide range of vision tasks, in particular object detection [17], [20], [21], [26]. Recently, object detection is dominated by a family of proposal-based approaches [20], [26], [28], which first generate a set of object proposals for an input image, followed by a classifier to predict object's score for each proposal. However, semantic part detection, despite its importance, has been much less studied. A *semantic part* is a fraction of an object which has semantic meaning and can be verbally described, such as a *wheel* of a *car* or a *chimney* of a *train*. Detecting semantic parts is a human ability, which enables us to recognize or parse an object at a finer scale.

In the real world, semantic parts of an object are frequently occluded, which make detection much harder. In this chapter, we investigate semantic part detection especially when these semantic parts are partially or fully occluded. We use the same datasets as proposed in [6], *i.e.*, the VehicleSemanticPart dataset and the extended occlusion counterpart VehicleOcclusion dataset. Some typical semantic part examples are shown in Figure 3.1. Note that, the VehicleOcclusion dataset is a synthetic

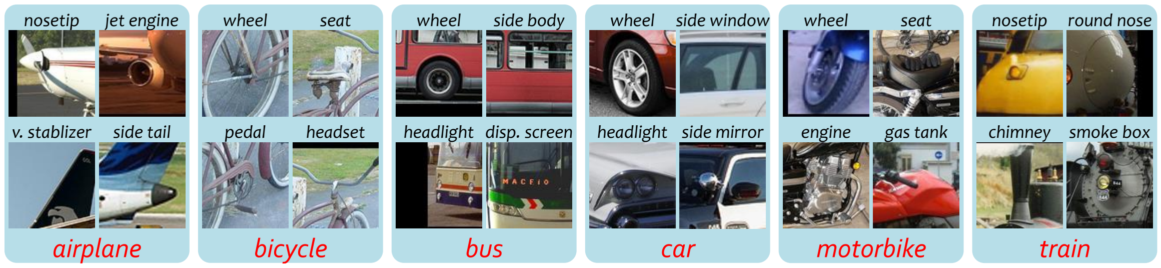

Figure 3.1. Typical semantic parts on six types of rigid objects from the VehicleSemanticPart dataset [6]. Some semantic parts (*e.g.*, *wheel*) can appear in different object classes, while some others (*e.g.*, *chimney*) only appear in one class (*train*).

occlusion dataset, where the target object is randomly superimposed with two, three or four irrelevant objects (named *occluders*) and the occlusion ratios of the target object is constrained. To the best of our knowledge, VehicleOcclusion is the *only* public occlusion dataset that provides accurate occlusion annotations of semantic parts like the occlusion ratio and number of occluders. This allows us to evaluate different methods under different occlusion difficulty levels.

One intuitive solution of dealing with occlusion is to train a model on the dataset that covers different occlusion cases. However, it is extremely difficult yet computationally intractable to collect a dataset that covers occlusion patterns of arbitrarily different numbers, appearances and positions. To overcome this difficulty, we suggest a more essential solution, *i.e.*, training detectors *only* on occlusion-free images, but allowing the learned knowledge (*e.g.*, the appearance template of semantic parts or the spatial relationship between semantic parts, *etc.*) to be transferred from non-occlusion training image to occlusion testing images. This motivates us to design models that are inherently robust to occlusions. A related work is [6], which points out that proposal-based deep networks are less robust to occlusion, and further proposes a voting mechanism that accumulates evidences from multiple local visual cues, and locates the semantic parts with the help of geometric constraints (*i.e.*, the spatial relationship between the visual cues and the target semantic part). However, this manually-designed framework is broken down into several stages, and thus it is difficult to optimize it in an end-to-end manner. This motivates us to see if the robustness shown in [6] can be incorporated into a deep network which enables end-to-end training naturally.

In this chapter, we propose DeepVoting which incorporates the robustness shown by [6] into a deep network. Following [47], the visual concepts are learned when the objects appear at a fixed scale since each neuron on the intermediate layer, *e.g.*, the *pool-4* layer of VGGNet, has a fixed receptive field size [48]. Therefore, we assume that the object scale is approximately the same in both training and testing stages. In the training stage, we use the ground-truth bounding-box to resize the object for the DeepVoting, and compute the object-to-image ratio to train a standalone network, *i.e.* ScaleNet [49], for scale prediction. In the testing stage, the trained ScaleNet is used to predict the resizing ratio, and then we resize the testing image according to the predicted ratio. We further show that visual concepts and spatial heat-maps can also exploit context information by using the whole image to train our model and achieve better performance, and we call this improved version DeepVoting+.

We investigate both DeepVoting and DeepVoting+ in our experiments. The first version, in which contexts are excluded (*i.e.* objects are cropped out for training), significantly outperforms [6] with the same settings, arguably because the end-to-end training manner provides a stronger method for joint optimization. The second version, which allows contextual cues to be incorporated, learns the training data better and consequently produces even higher detection accuracies. In comparison to the state-of-the-art object detectors such as Faster-RCNN [26], DeepVoting enjoys a consistent advantage, and the advantage becomes more significant as the occlusion level goes up. DeepVoting brings two additional benefits apart from being robust to occlusion: (i) DeepVoting enjoys much lower model complexity, *i.e.*, the number of parameters is one order of magnitude smaller, and the average testing speed is 2.5$\times$ faster; and (ii) DeepVoting provides the possibility to interpret the detection results via looking up the voting cues.

3.2 Related Works

Semantic parts or objects are important elements in computer vision. There are some previous works using semantic parts to assist object detection [50], [51]. Graphical model is used to assemble parts into an object. Also, parts can be used for fine-grained object recognition [52], [53], be applied as auxiliary cues to understand classification [54], or be trained for action and attribute classification [55]. Besides, [56] investigated the transferability of semantic parts across a large target set of visually dissimilar classes in image understanding.

Detecting semantic parts under partial or full occlusion is an important problem but is less studied before. [6] combined multiple visual concepts via the geometric constraints, *i.e.*, the spatial distribution of the visual concepts related to the target semantic parts, to obtain a strong detector. Different from [6], DeepVoting implements visual concept extraction and the geometric relationships as two differentiable layers, and attaches them directly to the intermediate outputs of a deep neural network (*e.g.*, VGGNet) to perform an end-to-end training. This design yields much better performances compared with [6].

3.3 DeepVoting Framework

In this section, we describe the DeepVoting, an end-to-end framework for semantic part detection under partial occlusion. The overall framework is illustrated in Figure 3.2. Specifically, we add two convolutional layers after the intermediate features of a deep neural network, *e.g.*, the neural responses at the *pool-4* layer of VGGNet. The first convolutional layer performs local template matching and outputs local visual cues named *visual concepts*, which are verified to be capable of corresponding to or weakly

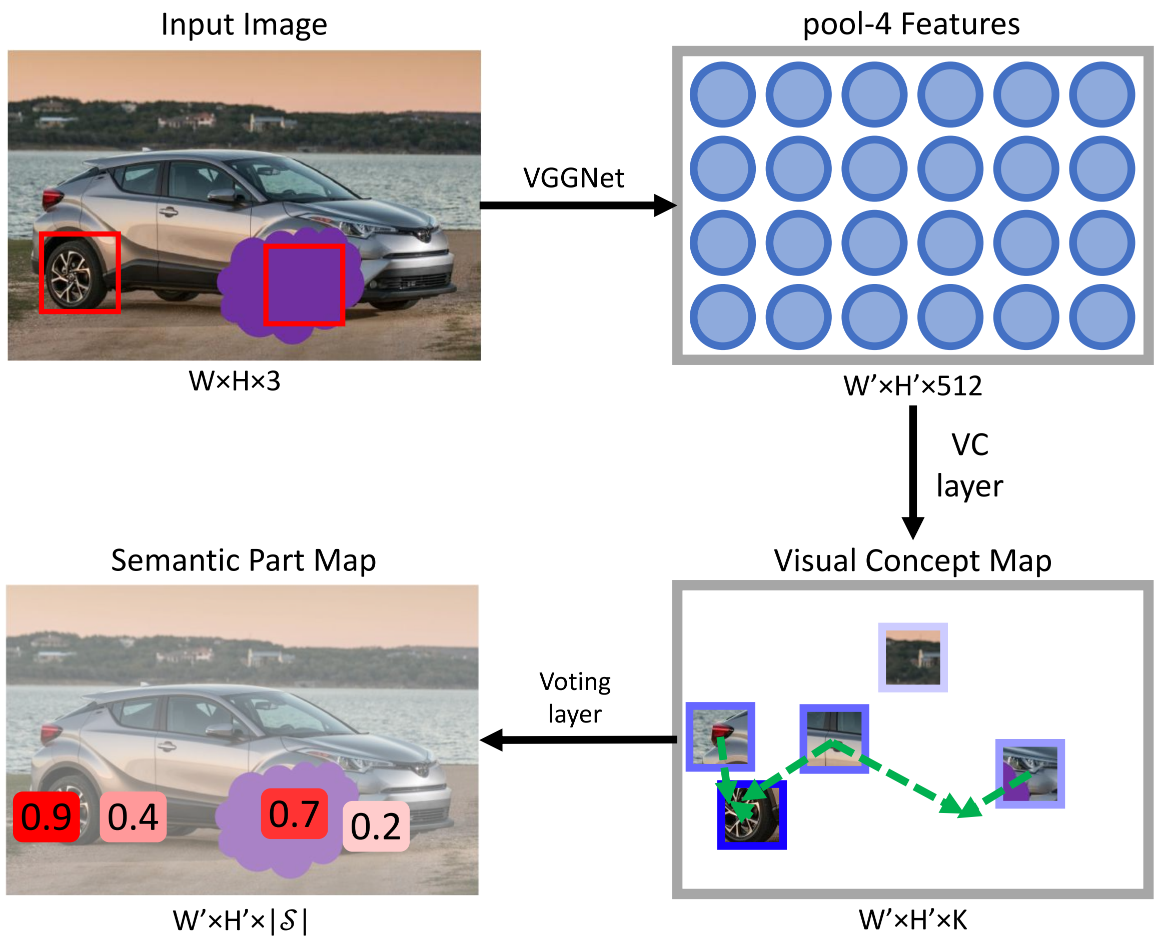

Figure 3.2. The overall framework of DeepVoting (best viewed in color). A *car* image with two *wheels* (marked by red bounding-boxes, and one of them is occluded) is fed into VGGNet [48], and the intermediate (*i.e. pool-4* layer) outputs are passed through a visual concept extraction layer and a voting layer. We aggregate local visual cues from the visual concept map (darker blue indicates more significant cues, in the lower right subfigure), and consider their spatial relationship to the target semantic parts via voting, and obtain a low-resolution confidence heat-map for semantic parts (darker red or a larger number indicates higher confidence). Based on this map, we perform bounding-box regression followed by non-maximum suppression to obtain the final detection results.

detecting semantic parts as in [47]. This layer is equipped with a ReLU activation [57], which sets a threshold of zero for filtering the matched patterns, as well as a dropout layer [58] as Faster-RCNN does. After that, the second convolution layer is added to perform a voting mechanism by learning and utilizing the spatial relationship between the local visual cues and the semantic parts. The spatial/geometric relations are stored as convolutional weights and visualized as *spatial heat-maps*. The visual concepts and spatial heat-maps can be learned either on cropped foreground objects only or on the whole image with background context. We first follow [6] to train our model on foreground objects only by cropping the object bounding boxes. We further show that visual concepts and spatial heat-maps can also exploit context information by using the whole image to train our model, and we call this improved version DeepVoting+.

3.3.1 Formulation

Let $\mathbf{I}$ denote an input image with a shape of $W \times H \times 3$. Following [6], we feed this image into a 16-layer VGGNet [48], and extract the *pool-4* features as a set of intermediate neural outputs. Denote the output of the *pool-4* layer as $\mathbf{X}$, which is a $W' \times H' \times D$ tensor, where W' and H' are the down-sampled scales of W and H, and D is 512 for VGGNet. These features can be considered as $W' \times H'$ high-dimensional vectors, and each of them captures the appearance of a local region determined by its index. Denote each D-dimensional feature vector as $\mathbf{x}_i$ where i is an index at the $W' \times H'$ lattice. These feature vectors are ℓ_2-normalized so that $\|\mathbf{x}_i\|_2 = 1$.

3.3.2 Visual Concept Extraction via the VC Layer

In [6], a set of visual concepts (VCs) $\mathcal{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_K\}$ are obtained via k-means clustering over the *pool-4* features, and each visual concept is considered intuitively as a template to capture the mid-level visual cues from these intermediate features. Specifically, the response of the visual concept $\mathbf{v}_k$ at the *pool-4* feature vector $\mathbf{x}_i$ is measured by the ℓ_2-distance, *i.e.*, $\|\mathbf{v}_k - \mathbf{x}_i\|_2^2$.

We note that the vector $\mathbf{x}_i$ has unit ℓ_2 length, and so $\|\mathbf{v}_k\|_2 \approx 1$ as it is averaged over a set of neighboring $\mathbf{x}_i$'s, thus we have $\|\mathbf{v}_k - \mathbf{x}_i\|_2^2 \approx 2 - 2\langle \mathbf{v}_k, \mathbf{x}_i \rangle$ where $\langle \cdot, \cdot \rangle$ is the dot product operator. Then the log-likelihood ratio tests are applied to eliminate negative responses. This is driven by the idea that the presence of a visual concept can provide positive cues for the existence of a semantic part, but the absence of a visual concept shall not give the opposite information.

Different from [6], DeepVoting implements this module as a convolutional layer, namely visual concept layer, and attaches it directly after the normalized intermediate outputs of a deep neural network. The kernel size of this convolutional layer is set to be 1×1, *i.e.*, each $\mathbf{x}_i$ is considered individually. The ReLU activation [57] follows to set the negative responses as 0's and thus avoids them from providing negative cues. We append a dropout layer [58] with a drop ratio 0.5, so that a random subset of the visual concept responses are discarded in the training process. This strategy facilitates the model to perform detection robustly using incomplete information and, consequently, improves the testing accuracy when occlusion is present.

The output of the visual concept layer is a map $\mathbf{Y}$ of size $W' \times H' \times |\mathcal{V}|$, where $\mathcal{V}$ is the set of visual concepts. We set $|\mathcal{V}| = 256$, though a larger set may lead to slightly better performance. Although these visual concepts are trained from scratch rather than obtained from k-means clustering [47], we find that they are also capable of capturing repeatable visual patterns and semantically meaningful. Some example outputs of the visual concept layer are visualized in Figure 3.4.

3.3.3 Semantic Part Detection via the Voting Layer

After the previous stage, we can find some fired visual concepts, *i.e.*, those local visual cues with positive response values. In [6], the fired visual concepts are determined via log-likelihood ratio tests. These *fired* visual concepts are then accumulated together for semantic part detection by incorporating the spatial constraints between each pair of visual concept and semantic part. It is motivated by the nature that a visual concept can, at least weakly, suggest the existence of a semantic part, at specific locations. For example, as shown in Figure 3.2, in a *car* image, finding a *headlight* implies that there is a *wheel* nearby, and the distance and direction from the *headlight* to the *wheel* are approximately the same under a fixed scale.

Different from [6], DeepVoting implements the spatial constraints between visual concepts and the semantic parts as another convolutional layer, named the voting layer, in which we set the receptive field of each convolutional kernel to be large, *e.g.*, 15×15, so that a visual concept can vote for the presence of a semantic part at a relatively long distance. This strategy helps particularly when the object is partially occluded, as effective visual cues often emerge outside the occluder and may be far from the target.

Though the spatial constraints are learned from scratch and only semantic part level supervision is imposed during training, they can still represent the frequency that visual concepts appear at different relative positions. We refer to each learned convolutional kernel at this layer as a spatial *heat-map*, and some of them are visualized in Figure 3.4.

Denote the output of the voting layer, *i.e.*, the semantic part map, as $\mathbf{Z}$. It is a $W' \times H' \times |\mathcal{S}|$ tensor where $\mathcal{S}$ is the set of semantic parts. Each local maximum at the semantic part map corresponds to a region on the image lattice according to their receptive filed. To generate a bounding-box for semantic part detection, we first set an anchor box, of size 100×100 and centered at this region, and then learn the spatial rescaling and translation to regress the anchor box (following the same regression procedure in [20]) from the training data. The anchor size 100×100 is the average semantic part scale over the entire training dataset [6].

3.4 Training and Inference

We train the network on an occlusion-free image corpus. This helps us to obtain a clear relationship between the visual concepts and the semantic parts. We discard the background region by cropping the object according to the ground-truth bounding box, to be consistent with [6]. Then, we rescale the cropped image so that the object short edge has 224 pixels, which is motivated by [47] to capture the visual concepts at a fixed scale. The image is fed into the 16-layer VGGNet, and we obtain the feature vectors at the *pool-4* layer.

These feature vectors are ℓ_2-normalized and passed through two layers for visual concept extraction and semantic part voting. We compare the output semantic part map $\mathbf{Z}$ with the ground-truth annotation tensor $\mathbf{L}$ by computing dice coefficient between prediction and ground-truth [59]. To generate the ground-truth annotation tensor in the output heat-map lattice, we find the nearest grid point at the $W' \times H'$ grid (down-sampled from the original image by a factor of 16) based on the center pixel of each annotated semantic part bounding-box, and set the labels of these positions as 1 and others as 0. Then we apply Gaussian smoothing on the binary ground-truth annotation, to generate the smoothed ground-truth annotation $\mathbf{L}$. The ground-truth annotation tensor $\mathbf{L}$ is also of shape $W' \times H' \times |\mathcal{S}|$. The similarity between $\mathbf{Z}$ and $\mathbf{L}$ is defined as:

$$\mathcal{D}(\mathbf{Z}, \mathbf{L}) = \frac{1}{|\mathcal{S}|} \sum_{s=1}^{|\mathcal{S}|} \frac{2 \times \sum_{w=1,h=1}^{W',H'} z_{w,h,s} \times l_{w,h,s}}{\sum_{w=1,h=1}^{W',H'} \left(z_{w,h,s}^2 + l_{w,h,s}^2 \right)},$$

It is straightforward to compute the gradients based on the loss function $\mathcal{L}(\mathbf{X}, \mathbf{L}) = 1 - \mathcal{D}(\mathbf{Z}, \mathbf{L})$.

During the testing phase, we first use an independently trained ScaleNet [49] to obtain the object scale. Then, we rescale the image so that the short edge of the object roughly contains 224 pixels. We do not crop the object because we do not know its location during the testing. Then, the image is passed through the VGGNet followed by both visual concept extraction layer and semantic part voting layer, and finally we apply the spatial rescaling and translation (*i.e.* bound-box regression) to the anchor box (100×100) towards more accurate localization. A standard non-maximum suppression is performed to finalize the detection results by pruning redundant detections.

DeepVoting is trained on the images cropped with respect to the object bounding-boxes to be consistent with [6]. However, visual concepts and spatial heat-maps can also exploit context outside object bounding boxes. To verify this, we train an improved version, named DeepVoting+, without cropping the bounding boxes. We also resize the image so that the object short edge contains 224 pixels in the training stage, and the testing stage is the same as DeepVoting. Experiments show that DeepVoting+ achieves better performance compared with DeepVoting.

3.4.1 The Scale Prediction Network

The above framework is based on an important assumption, that the objects appear in approximately the same scale. This is due to two reasons. First, as shown in [47], the visual concepts emerge when the object is rescaled to the same scale, *i.e.*, the short edge of the object bounding box contains 224 pixels. Second, we expect the voting layer to learn fixed spatial offsets which relate semantic parts to their supporting visual concepts. As an example, the heat-map delivers the knowledge that in the side view of a *car*, the *headlight* often appears at the upper-left direction of a *wheel*, and the spatial offset on x and y axes are about 64 and 48 pixels (4 and 3 at the *pool-4* grid), respectively. Such information is not scale-invariant.

To deal with the aforementioned issues, we introduce an individual network, namely the ScaleNet [49], to predict the object scale in each image. The main idea is to feed an input image to a 16-layer VGGNet for a regression task (the *fc-8* layer is replaced by a 1-dimensional output), and the regression target is the ground-truth object size. In order to learn the scale of objects on a uniform canvas, each input image is rescaled, so that the long edge contains 224 pixels. Then the rescaled image is placed at the center of a canvas with a shape of 224×224 pixels and the remaining pixels on the canvas are filled up with the averaged intensity. During the training, we consider the short edge of the object as the target, and ask the deep network to predict the ratio of the object short edge to the image long edge (224 pixels). In the testing phase, an image is prepared and fed into the network in the same flowchart, and the predicted rescaling ratio is used to normalize the object to the desired size, *i.e.*, its short edge contains 224 pixels. We conduct experiments to verify the effectiveness of the proposed ScaleNet (in Subsection 3.5.4), which show that this method works very well on our dataset and task.

3.5 Experiments Results for Part Detection

3.5.1 Dataset and Baseline

We use the VehicleSemanticPart dataset and the VehicleOcclusion dataset proposed in [6] for experiments. The VehicleSemanticPart dataset contains 4549 training images and 4507 testing images covering six types of vehicles, *i.e.*, *airplane*, *bicycle*, *bus*, *car*, *motorbike* and *train*. In total, 133 types of semantic parts are annotated. The VehicleOcclusion dataset is an extension of the VehicleSemanticPart dataset. For each testing image in VehicleOcclusion dataset, some randomly-positioned occluders (irrelevant to the target object) are placed onto the target object, and make sure that the occlusion ratio of the target object is constrained at different levels.

We train six models, one for each vehicle class. All the models are trained on an occlusion-free dataset, but evaluated on either non-occluded images, or the images with different levels of occlusions added. In the later case, we vary the difficulty level by using images with different occlusion fractions of the vehicle object. We evaluate all the competitors following a popular criterion [7], which computes the mean average precision (mAP) based on the list of detected semantic parts. A detected box may be considered to be true-positive if its IoU rate with a ground-truth box is not lower than 0.5. Each semantic part is evaluated individually, and the mAP of each object class is the average mAP over all the semantic parts.

DeepVoting and DeepVoting+ are denoted as **DV** and **DV+** respectively, and they are compared with the following four baselines:

- **KVC**: These visual concepts are obtained by running k-means clustering on a set of *pool-4* features[47]. The ScaleNet is used to tackle scale issue and the extracted visual concepts are directly used to detect the semantic parts.

- **DVC**: These visual concepts are obtained from DeepVoting internally, *i.e.*, the weights of the 1×1 visual concept extraction layer. The ScaleNet is used to tackle scale issue and the extracted visual concepts are directly used to detect the semantic parts.

- **VT**: The voting method first finds fired visual concepts via log-likelihood ratio tests, and then utilizes spatial constraints to combine these local visual cues, as described in [6].

- **FR**: We train Faster-RCNN [26] models for each vehicle category independently. Each semantic part of a vehicle category is considered as a separate class during the training, *i.e.*, for each category, we train a model with $|\mathcal{S}| + 1$ classes, corresponding to $|\mathcal{S}|$ semantic parts and the background. Different from other baselines, Faster-RCNN here is trained on full images, *i.e.*, object cropping is not performed. This enables Faster-RCNN to use context for semantic parts detection and handle scale issue naturally since semantic parts with various scales are used in training.

3.5.2 Semantic Part Detection without Occlusion

As a simplified task, we evaluate our algorithm in detecting semantic parts on non-occluded objects. This is also a baseline for later comparison. In Table 3.1, we list the detection accuracy produced by different methods. The average detection accuracies by both voting and DeepVoting are significantly higher than using single visual concept for

Category	**KVC**	**DVC**	**VT**	**FR**	**DV**	**DV+**
airplane	15.8	26.6	30.6	56.9	59.0	**60.2**
bicycle	58.0	52.3	77.8	90.6	89.8	**90.8**
bus	23.8	25.1	58.1	**86.3**	78.4	81.3
car	25.2	36.5	63.4	83.9	80.4	**80.6**
motorbike	32.7	29.2	53.4	63.7	65.2	**69.7**
train	12.3	12.8	35.5	59.9	59.4	**61.2**
mean	28.0	30.4	53.1	73.6	72.0	**74.0**

Table 3.1. Comparison of detection accuracy (mean AP, %) of **KVC**, **DVC**, **VT**, **FR**, **DV** and **DV+** on non-occluded testing images. Note that **DV+** is DeepVoting trained with context outside object bounding boxes.

detection, regardless whether the visual concepts are obtained from k-means clustering or the DeepVoting visual concept layer. This indicates the advantage of the approach which aggregates multiple visual cues for detection, compared with the approaches which use single visual concept for semantic part detection. Meanwhile, DeepVoting is much better than voting due to the better scale prediction and the end-to-end training manner. Even the right scale is provided for voting (oracle scale results in [6]), DeepVoting still outperforms it by more than 20% in terms of averaged mAP over 6 objects, which indicates the benefit brought by the joint optimization of both weights for visual concept extraction layer and voting layer in an end-to-end manner.

On the other hand, DeepVoting produces slightly lower detection accuracy compared to Faster-RCNN. We argue that Faster-RCNN benefits from the context outside object bounding-boxes, as we can see, if we improve DeepVoting by adding context during the training (*i.e.* DeepVoting+), Faster-RCNN will be less competitive compared with our method. Meanwhile, DeepVoting enjoys lower computational overheads, *i.e.*, it runs 2.5× faster.

3.5.3 Semantic Part Detection under Occlusion

We further detect semantic parts when the objects are occluded at three different occlusion levels. Since the baselines **KVC** and **DVC** perform much worse than

Occlusion	Category	**VT**	**FR**	**DV**	**DV+**
L1	*airplane*	23.2	35.4	**40.6**	**40.6**
	bicycle	71.7	77.0	83.5	**85.2**
	bus	31.3	55.5	56.9	**65.8**
	car	35.9	48.8	56.1	**57.3**
	motorbike	44.1	42.2	51.7	**55.5**
	train	21.7	30.6	33.6	**43.7**
	mean	38.0	48.3	53.7	**58.0**
L2	*airplane*	19.3	27.0	31.4	**32.3**
	bicycle	66.3	62.0	78.7	**79.6**
	bus	19.3	40.1	44.1	**54.6**
	car	23.6	30.9	40.0	**41.7**
	motorbike	34.7	32.4	41.4	**43.4**
	train	8.4	17.7	19.8	**29.8**
	mean	28.6	35.0	42.6	**46.9**
L3	*airplane*	15.1	20.1	**25.9**	25.4
	bicycle	54.3	41.1	**63.0**	62.5
	bus	9.5	25.8	30.8	**40.5**
	car	13.8	19.8	27.3	**29.4**
	motorbike	24.1	20.1	29.4	**31.2**
	train	3.7	10.9	13.3	**22.2**
	mean	20.1	23.0	31.6	**35.2**

Table 3.2. Comparison of detection accuracy (mean AP, %) of **VT**, **FR**, **DV** and **DV+** when the object is occluded at three different levels. Note that **DV+** is DeepVoting trained with context outside object bounding boxes.

other methods even when occlusion is not present, we ignore these two baselines when performing semantic part detection under occlusion. In the first level (*i.e.* **L1**), we place 2 occluders on each vehicle object, and the occluded ratio r of the object, computed by pixels, satisfying $0.2 \leqslant r < 0.4$. For **L2** and **L3**, we have 3 and 4 occluders, and $0.4 \leqslant r < 0.6$ and $0.6 \leqslant r < 0.8$, respectively. The original occlusion-free testing set is denoted as **L0**. The detection results are summarized in Table 3.2. One can see that DeepVoting outperforms the voting and the Faster-RCNN significantly in the occlusion cases. Compared with the Faster-RCNN, the accuracy gain increases as the occlusion level goes up, suggesting the advantage of DeepVoting in detecting occluded semantic parts. As a side evidence and ablation study, we investigate the impact of

Category	**Recall**				**mAP** + Prop.			**mAP** by **DV+**		
	L0	L1	L2	L3	L1	L2	L3	L1	L2	L3
airplane	99.3	98.1	97.4	96.7	36.2	27.7	20.7	**40.6**	**32.3**	**25.4**
bicycle	99.5	99.0	98.0	96.5	77.9	64.0	44.7	**85.2**	**79.6**	**62.5**
bus	99.8	96.3	93.8	91.5	57.1	42.4	28.3	**65.8**	**54.6**	**40.5**
car	99.8	96.0	94.4	92.7	48.2	30.2	19.4	**57.3**	**41.7**	**29.4**
motorbike	99.0	96.5	95.7	93.3	43.6	33.1	21.3	**55.5**	**43.4**	**31.2**
train	98.3	93.5	90.6	85.6	32.0	19.4	11.3	**43.7**	**29.8**	**22.2**
mean	99.3	96.6	95.0	92.7	49.2	36.1	24.2	**58.0**	**46.9**	**35.2**

Table 3.3. Left 4 columns: the recall rates (%) of the proposal network at different occlusion levels. Middle 3 and right 3 columns: detection mAPs (%) of Faster-RCNN with ground-truth bounding boxes added as additional proposals and DeepVoting+ at different occlusion levels.

the size of spatial heat-map (the kernel of the voting layer). At the heaviest occlusion level (*i.e.* **L3**), when we reduce the shape of the heat-map from the default 15×15 to a smaller 11×11, the mean detection accuracy drops from 31.6% to 30.6%, suggesting the usefulness of long-distance voting in detecting occluded semantic parts. When the heat-map size is increased to 19×19, the accuracy is slightly improved to 31.8%. Therefore, we keep the kernel size to be 15×15 for a better computation tradeoff.

To verify our motivation that Faster-RCNN suffers downgraded performance in both the proposal network and the classifier, we investigate both the recall of the proposals and the accuracy of the classifier. Results are summarized in Table 3.3. First, we can see that the recall of the proposals goes down significantly as the occlusion level goes up, since the objectiveness of the semantic part region may become weaker due to the randomly placed occluders. Thus the second stage, *i.e.*, classification, has to start with a relatively low-quality set of candidates. In the second part, we add the ground-truth bounding-boxes to the existing proposals so that the recall is 100%, feed these candidates to the classifier, and evaluate the detection performance on the occluded images. Even with such benefits, Faster-RCNN still produces unsatisfying detection accuracy. For example, in detecting the semantic parts of a *bicycle* at the highest occlusion level (**L3**), making use of the additional proposals from ground-truth

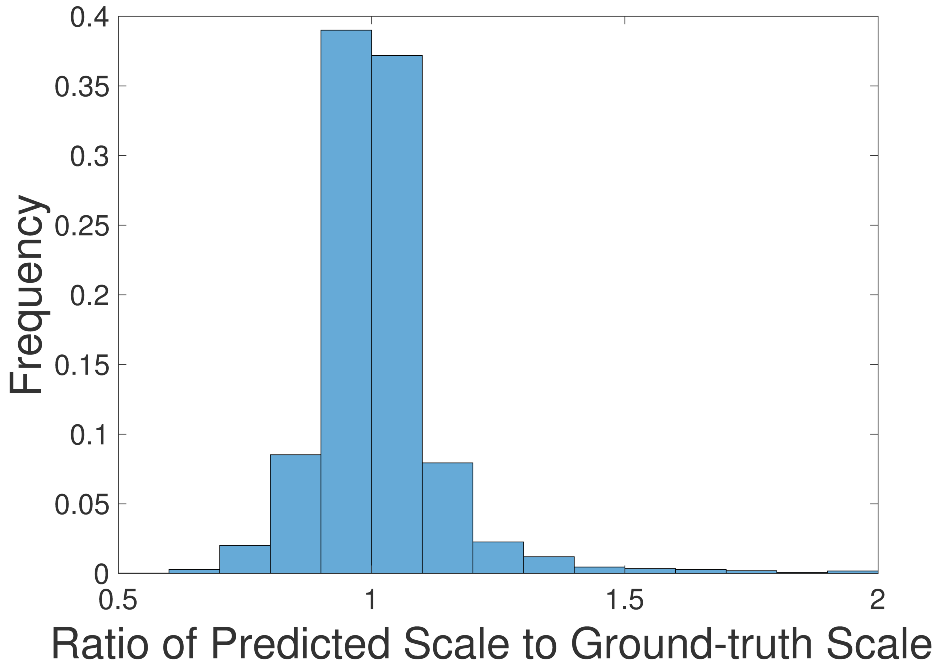

Figure 3.3. The accuracy of scale prediction. The x-axis is the ratio of predicted scale to the ground-truth scale, and a ratio of 1 indicates perfect scale prediction while ratios differ from 1 with a large gap indicate incorrect scale estimations. The y-axis is the frequency of the corresponding prediction/ground-truth ratio over the testing set.

bounding boxes merely improves the detection accuracy from 41.1% to 44.7%, which is still much lower than the number 62.5% achieved by DeepVoting+. This implies that the classifier may be confused since the occluder changes the appearance of the proposals.

3.5.4 Scale Prediction Accuracy

We investigate the accuracy of ScaleNet, which is essential for scale normalization. For each testing image, we compute the ratio of the predicted object scale to the actual scale, and plot the distribution of this ratio over the entire testing set in Figure 3.3. One can see that in more than 75% cases, the relative error of the predicted scale does not exceed 10%. Actually, these prediction results are accurate enough for DeepVoting. We also conduct an ablation experiment to test DeepVoting on the images without

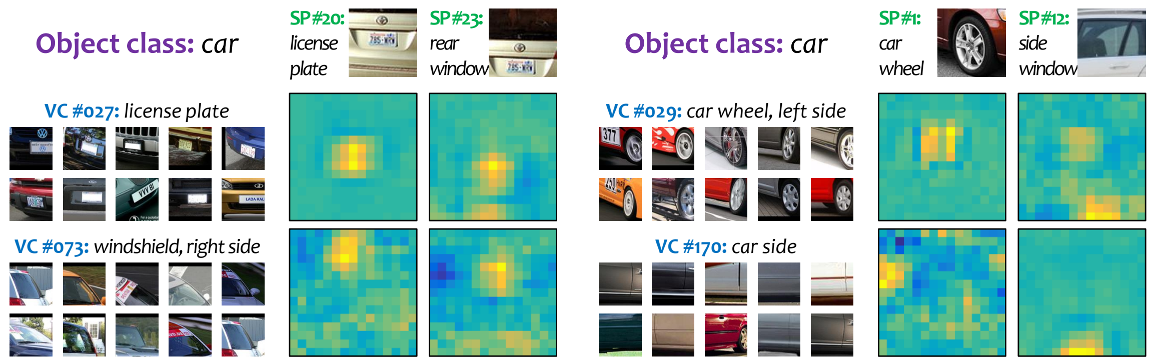

Figure 3.4. Visualization of visual concepts and spatial heat-maps (best viewed in color). For each visual concept, we show 10 patches with the highest responses. Each spatial heat-map illustrates the cues to support detecting a semantic part at different relative spatial locations, in which yellow, cyan and dark blue indicate positive, neutral and negative cues, respectively. For example, `VC #073` (*windshield*) often appears above `SP #20` (*license plate*), and `VC #170` (*car side bottom*) often appears below `SP #12` (*side window*).

occlusion, while providing the ground-truth scales of the objects instead of using ScaleNet to predict the scales. Even in this oracle setting, the detection accuracy is only slightly improved from 72.0% to 74.5%. It shows that our ScaleNet provides good enough estimations of object scales for semantic part detection.

3.6 Visualization and Explanation

3.6.1 Visualizing Visual Concepts and Heat-maps

In Figure 3.4, we show some typical examples of the learned visual concepts and spatial heat-maps. The visualization of visual concepts follows the approach used in [47], which finds 10 most significant responses on each convolutional filter, *i.e.*, the matching template, traces back to the original image lattice, and crops out the region corresponding to the receptive field of the neuron at the *pool-4* layer. We can see the visual concepts can learn local visual patterns with meaningful semantics, for example, `VC #027`, `VC #029`, `VC #073` and `VC #170` correspond to license plates, left side of car wheels, right side of windshields and car side respectively. We also visualize spatial heat-maps for several relevant pairs of visual concept and semantic part, and plot the

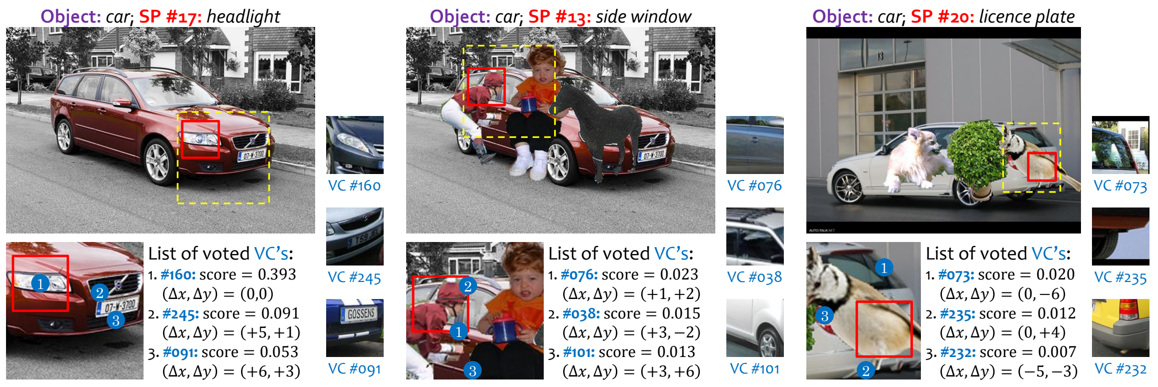
Object: car; SP #17: headlight
Object: car; SP #13: side window
Object: car; SP #20: licence plate
List of voted VC's:
1. #160: score = 0.393 (Δx, Δy) = (0,0)
2. #245: score = 0.091 (Δx, Δy) = (+5,+1)
3. #091: score = 0.053 (Δx, Δy) = (+6,+3)
VC #160
VC #245
VC #091
List of voted VC's:
1. #076: score = 0.023 (Δx, Δy) = (+1,+2)
2. #038: score = 0.015 (Δx, Δy) = (+3,−2)
3. #101: score = 0.013 (Δx, Δy) = (+3,+6)
VC #076
VC #038
VC #101
List of voted VC's:
1. #073: score = 0.020 (Δx, Δy) = (0,−6)
2. #235: score = 0.012 (Δx, Δy) = (0,+4)
3. #232: score = 0.007 (Δx, Δy) = (−5,−3)
VC #073
VC #235
VC #232

Figure 3.5. DeepVoting allows us to explain the detection results. In the example of heavy occlusion (the third column), the target semantic part, *i.e.*, the *license plate* on a *car*, is fully occluded by a *bird*. With the help of several supporting visual concepts (illustrated in blue dots), especially the 73-rd VC (also visualized in Figure 3.4), we can infer the position of the occluded semantic part (marked in the red bounding-box). Note that we only plot the top 3 VC's with the highest scores, regardless the number of supporting VC's can be much larger.

convolutional weights of the voting layer. As the visualization shows, the heat-maps can capture the correspondence between visual concepts and semantic parts, *i.e.*, whether a visual concept can support a semantic part, as well as the relative spatial location between the visual concept and the semantic part. For example, `VC #073` (*windshield*) can suggest the existence of `SP #20` (*license plate*), which may appear in a region below the `VC #073`.

3.6.2 Explaining the Detection Results

Finally, we show an intriguing benefit of our approach, which allows us to explain the detection results. In Figure 3.5, we visualize three examples, in which the target semantic parts are not occluded, partially occluded and fully occluded, respectively. DeepVoting can detect the occluded semantic parts, and is also capable of looking up the voting (supporting) visual concepts for explaining and diagnosing the detection, to dig into errors and understand the working mechanism of our approach.

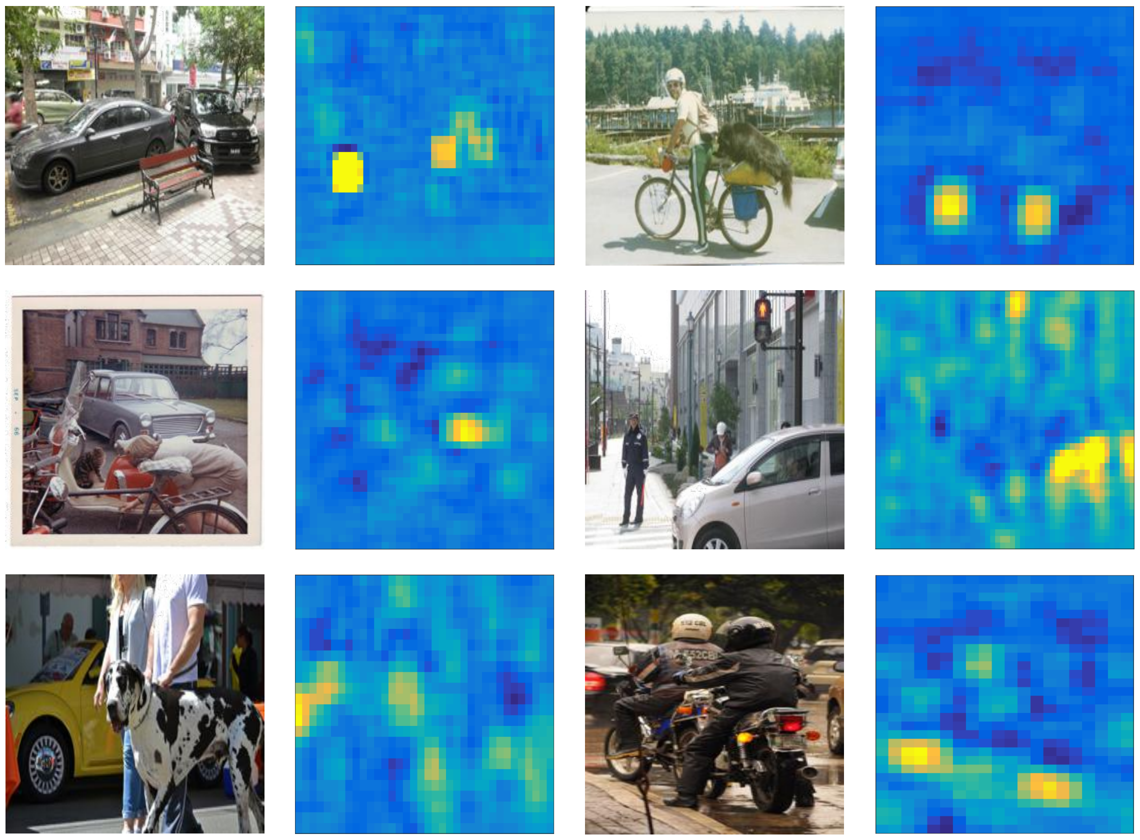

Figure 3.6. Input image from MS COCO `2014val` and the corresponding semantic part detection heat-map. From top to bottom and left to right, in row major order: detecting *wheels* of *cars*; detecting *wheels* of a *bike*; detecting *license plates* of a *car*; detecting *side windows* of a *car*; detecting *headlights* of a *car*; detecting *wheels* of a motorbike.

3.6.3 Detection under Natural Occlusions

Our proposed methods are evaluated on the VehicleOcclusion dataset, in which the occluders are superimposed on the natural images. In this section we demonstrate the effectiveness of our method on real-world occlusions. Since there are no real-world semantic part datasets with different occlusion levels, we randomly choose several images with partial occlusion and/or truncation from MS COCO [3] `2014val`. We run DeepVoting on these images and plot the detection heat-map (yellow for high confidence scores and blue for low confidence scores) in Figure 3.6. We can see that although some parts are occluded (*e.g.* The left rear *wheel* of the left *car* is partially occluded by a chair in the top left subfigure), DeepVoting can still detect the semantic parts in good quality. Note that no ground-truth semantic part annotations are available for these images.

3.7 Conclusion and Future Works

In this chapter, we address an important yet less-studied task, *i.e.*, detection under arbitrary and unknown occlusions, which could not be covered in the training dataset. To this end, we propose a robust and explainable deep learning detection framework, named DeepVoting, for semantic part detection with possible occlusions. In the proposed network, the intermediate visual representations, named *visual concepts*, are extracted and used to vote for semantic parts via two convolutional layers. The spatial relationship between visual concepts and semantic parts is learned from an occlusion-free dataset and then transferred to the occluded testing images. DeepVoting is evaluated on both the VehicleSemanticPart dataset and the VehicleOcclusion dataset, and shows comparable performance as Faster-RCNN in the non-occlusion scenario, and superior performance in the occlusion scenario. If context outside objects is utilized during the training, this framework (*i.e.* DeepVoting+) outperforms both DeepVoting and Faster-RCNN significantly under all scenarios. Moreover, our approach enjoys the advantage of being explainable, which allows us to diagnose the semantic parts detection results by checking the contribution of each voting visual concepts.

As future works, DeepVoting can be further improved in several aspects. First, the current framework is designed for rigid objects (*i.e.* vehicles), as the spatial relationship between visual concepts and semantic parts are modeled in convolution kernels. Better representation of this information with more flexible models can help to extend the current method to detect semantic parts of non-rigid and articulated objects like animals. Second, the proposed method can be extend to perform object-level detection under occlusion, by either directly modeling the correspondence between the visual concepts and object bounding-boxes, or designing a two-level hierarchy which models the correspondence between visual concepts and semantic parts, and between the semantic parts and the object bounding-boxes in a bottom-up manner. Also, viewpoints are important for object-level detection with our approach since the semantic parts may appear at totally different locations for different viewpoints. It would be necessary to learn a mixture of templates which models the spatial relationship between visual cues and object bounding-boxes for various viewpoints, as designed in [60].

Chapter 4

Enriching Semantics for Multi-layer Scale-invariant Detectors

In this chapter, we move our focus from semantic part detection to a more popular and fundamental vision task, *i.e.* general object detection. Unlike the previous task, in which there is usually only one vehicle object in an image and all parts of vehicle objects are learned on objects with similar sizes (*i.e.* with a short edge of 224 pixels for the vehicle bound-boxes), detecting general objects in the wild involves a more complicated issue, *i.e.*, different objects in a single image may have largely different scales/sizes. This requires us to build detectors with feature maps which 1) have strong object-level semantics and 2) cover objects with different sizes. To this end, we propose a novel single-stage object detection network named **Detection with Enriched Semantics** (**DES**). DES is a multi-layer detection network with increasing receptive field sizes from lower to higher layers, to detect objects with different sizes. The key innovation of our method is to enrich the semantics of object detection feature maps at different layers, by a semantic segmentation branch and a global activation module. Thus the detection feature maps, especially those in the lower layers, contain stronger semantics of object-level knowledge compared with traditional multi-layer detectors, of which the lower layer feature maps are learning low-level visual patterns like color, texture, *etc.* The segmentation branch is supervised by weak (*i.e.* pseudo) segmentation ground-truth, with no expensive pixel-wise annotation required. In conjunction with that, we employ a global activation module which learns relationship between channels and object classes in a self-supervised manner. Comprehensive experiment results on both PASCAL VOC [7] and MS COCO [3] detection datasets demonstrate the effectiveness of the proposed method. In particular, with a 16-layer VGGNet (VGG16) based DES, we achieve an mAP of 81.7 on VOC2007 `test` and an mAP of 32.8 on MS COCO `test-dev` with an inference speed of 31.5 milliseconds per image on a single Titan Xp GPU. With a faster version which takes lower resolution input images, we achieve an mAP of 79.7 on VOC2007 with an inference speed of 13.0 milliseconds per image.

4.1 Motivation and Overview

With the emergence and rapid development of deep neural networks, computer vision has been improved significantly in many aspects such as image classification [8]–[11], [61], object detection [17], [21], [26], [28], [62], and segmentation [12]–[14]. Among them, object detection is a fundamental task which has already been extensively studied, especially for detecting general objects in the wild [3], [7]. Currently, popular object detection frameworks lie in two directions: the two-stage frameworks such as Faster-RCNN [26] and R-FCN [28] which extract proposals, followed by per-proposal classification and regression; and the one-stage frameworks such as YOLO [21] and SSD [17], which apply object classifiers and regressors in a dense manner directly on the backbone feature maps without objectiveness-based pruning. Both of them do classification and regression on a set of pre-computed anchors.

A fundamental challenge for general object detection is the various scales for different objects in a scene, which are inevitable since different types of objects can have largely different 3D shape in the world coordinate, and objects can have different

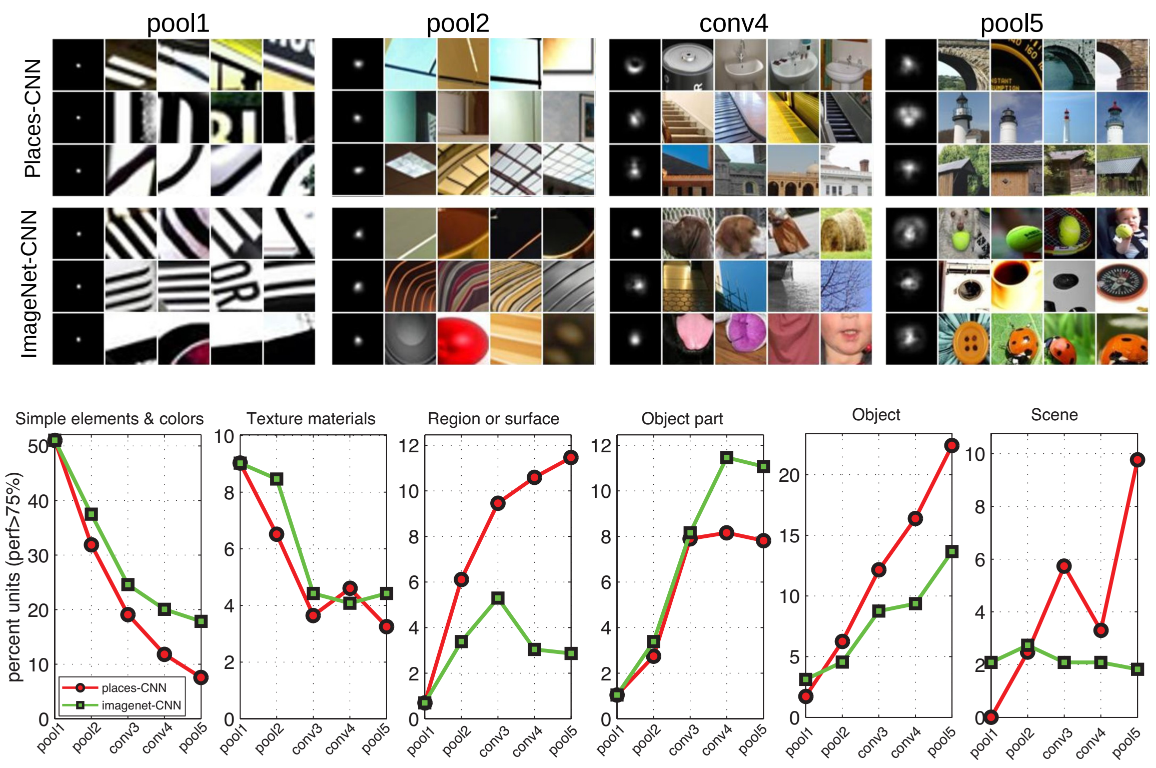

Figure 4.1. This figure is a compilation of two figures drawn in [18]. The upper subfigure shows the receptive fields and the top activating patches for three neurons at different layers of the two CNNs proposed in [18]. We can see the *conv4* layer (which is the penultimate convolution layer) only corresponds to basic low-level elements. Even for the *pool5* layer, the neurons sometimes fire at different objects with similar appearance. The lower subfigure provides quantitative analysis about the semantic types captures by all the neurons in each layer. For a ImageNet-trained CNN, there are still around 20% neurons captures simple elements and colors in *conv4* and *pool5*.

magnifications controlled by their relative distances to the camera lens. To detect general objects in natural images with different scales/sizes (*i.e.* scale-invariant), conventional single-stage object detectors, such as SSD [17], use multiple convolutional layers (in the form of a feature pyramid) to detect objects with different sizes and aspect ratios. SSD uses a backbone network (*e.g.*, VGG16) to generate a low-level detection feature map. Based on that, several layers of object detection feature maps are built, learning semantic information in a hierarchal manner. Smaller objects are detected by lower layers while larger objects are detected by higher layers. However, due to the hierarchy in the feature pyramid, the semantics are learned in a bottom-up

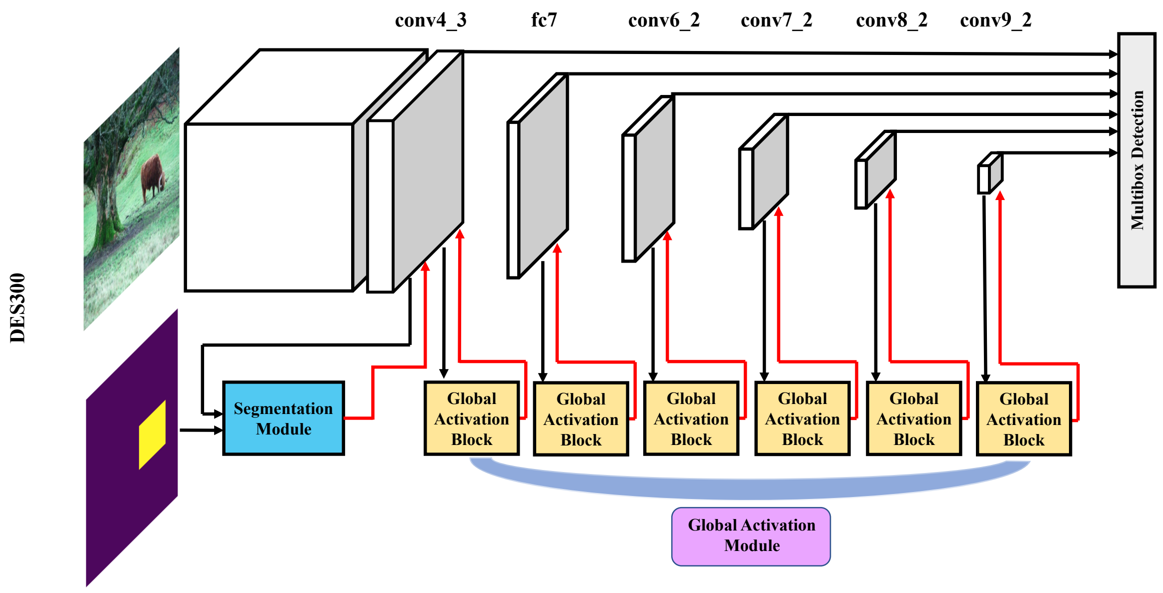

Figure 4.2. Pipeline for DES: the upper half is the object detection branch for DES which has six detection feature maps from *conv4_3* up to *conv9_2*; the lower half illustrates the segmentation branch and the global activation module. The segmentation branch is employed at the first detection feature map *i.e.* *conv4_3*. The global activation module consists of six global activation blocks. Those global activation blocks are added at each of the detection feature maps. The black arrows pointed to those modules are the input flow, and the red arrows pointed out from those modules are the output flow to replace the original feature map.

manner gradually. The low-level features usually only capture basic visual patterns without strong semantic information. This phenomenon has been pointed out in [18], and demonstrated in Figure 4.1 (a compilation of two figures borrowed from [18]). This may cause two problems: small objects may not be detected well by low-level feature maps, and the quality of high-level feature maps is also damaged by the imperfect low-level features.

In this chapter, we aim to address the problem discussed above, by designing a novel single-stage detection network, named **Detection with Enriched Semantics** (**DES**), which consists of two branches – a detection branch and a segmentation branch. The overall pipeline is illustrated in Figure 4.2. The detection branch is a typical single-stage detector, which takes the *conv4_3* layer from VGG16 as its backbone

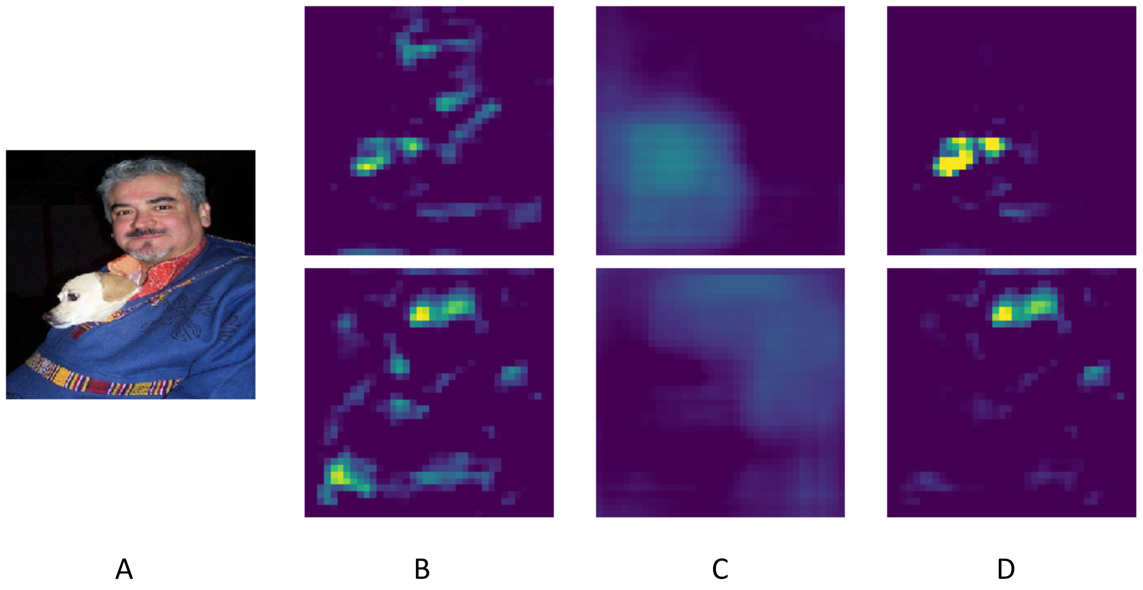

Figure 4.3. low-level features augmented with semantic meaningful features from the segmentation branch. A: original image fed into our detection network. B: original low-level detection features (X) for the input image. C: semantic meaningful features (Z) from the segmentation branch. D: augmented low-level features $(X' = X \odot Z)$ which is then used in the later stages for our detection network. We can see that X' can capture both basic visual pattern and high level semantic information.

feature map. Based on that, it builds a series of feature maps (*i.e.*, from *conv4_3* to *conv9_2*) to detect objects of small to large sizes, by attaching anchors with different sizes and aspect ratios on these feature maps. This is shown in the upper part of Figure 4.2.

An auxiliary segmentation branch is used to enrich the low-level detection feature map with strong semantic information. It takes the low-level detection feature map as input, and learns semantic segmentation supervised by bounding-box level segmentation ground-truth, as a side branch. Then it augments the low-level detection feature map with its semantically meaningful feature map, as shown in the left lower part of Figure 4.2.

Figure 4.3 gives an illustration of this semantic enrichment process. The original low-level detection feature map (in column B) is activated by the segmentation feature map (in column C), to generate an augmented low-level feature map (in column D), which can capture both the basic low-level visual pattern as well as the semantic information of the object. This enrichment can also be considered as an attention process, in which each channel of the original low-level feature map is activated by a semantically meaningful attention map, to magnify the firing neurons at the semantically related locations.

In addition to the segmentation branch attached to the low-level detection feature map, we also employ a global activation module for higher level detection feature maps. It consists of several global activation blocks, as shown in the right lower corner of Figure 4.2. The global activation block can prune out the location information, and learn the correspondence between feature channels and object classes in a self-supervised manner, which increases the semantic information of the object detection feature maps at higher layers.

We summarize our contributions of this chapter as the follows:

- We improve the typical deep single-stage detectors by enriching semantics, with a semantic segmentation branch to enhance low-level detection features, and a global activation module to learn the semantic correspondence between detection feature map channels and object classes for higher-level object detection features.
- We significantly improve the performance compared with popular single-stage detectors. DES achieves an mAP of 81.7 on VOC2007 `test` and mAP of 32.8 on COCO `test-dev`.
- DES is time efficient. With a single Titan Xp GPU, it achieves 31.7 FPS, and is much faster than competitors like R-FCN and ResNet-based SSD.

4.2 Related Work

In addition to object detection, semantic segmentation is another important computer vision task, which requires each pixel of the input image to be classified/assigned to one of the class labels. Traditional semantic segmentation methods such as DeepLab [12] and fully convolutional network (FCN) [14] need pixel-wise annotations for the training. On the other hand, it has been shown that weakly annotated training data such as bounding-boxes or image-level classification labels can also be utilized for semantic segmentation in [63].

We are not the first one to show segmentation information can be leveraged to help object detection [13], [64], [65]. Gidaris and Komodakis [64] use semantic segmentation-aware CNN features to augment detection features by concatenation at the highest level, but our work differs in a way that we put the segmentation information at the lowest detection feature map, and we use activation instead of concatenation to combine object detection features and segmentation features. He *et al.* [13] shows that multi-task training of object detection and instance segmentation can help to improve the object detection task with extra instance segmentation annotation, however, we do not consider extra annotation in our work, which could take much more time to label. Another difference is how the segmentation branch in used. He *et al.* [13] train detection and segmentation in parallel, but our method uses segmentation features to activate the detection features.

Other work such as [66] has been done to improve object detectors by using top-down architecture to increase the semantic information. Our work achieves this in a simpler way, which does not involve reverse connections.

4.3 Methodology

In this section we present the detailed design of our framework. We will first go over our framework, and then discuss three key components in Subsections 4.3.1, 4.3.2 and 4.3.3 respectively.

Detection with Enriched Semantics (DES) is a single-stage object detection network with three components: a single-stage detection branch, a segmentation branch to enrich semantics at low-level detection layer, and a global activation module to enrich semantics at higher-level detection layers. We use SSD [17] as our single-stage detection branch. SSD is built on top of a CNN backbone which generates a low-level detection feature map for object detection (*conv4_3* for VGG16). Based on that, SSD builds a series of feature maps (*i.e.*, from *conv4_3* to *conv9_2*) to detect objects of small to large sizes, in a hierarchical manner, by applying anchors with different sizes and aspect ratios on these feature maps.

In order to deal with the problems discussed in the previous sections, we employ a segmentation branch to enrich the low-level detection feature maps with semantic information. This segmentation branch is added at the first detection feature map layer *conv4_3*. Traditional segmentation algorithms require pixel-wise label annotation, but this is not feasible in the object detection task. Instead, we use bounding-box level weak segmentation labels to perform supervision for segmentation task. As shown in the left lower part in Figure 4.2, our segmentation branch takes *conv4_3* as input, represented by the black arrow pointed from *conv4_3* to segmentation branch. Then it generates a semantically augmented low-level feature map *conv4_3'*, which will be used for detection, represented by the red arrow pointed from segmentation module to *conv4_3*. By employing segmentation branch, our network becomes a multi-task learning problem with both detection and segmentation losses. The feature map generated by the segmentation branch captures object-level semantic information for each local area since the segmentation supervision pushes each local area to be assigned to one of the classes. The detailed design of the segmentation branch is illustrated in Subsection 4.3.1.

At higher-level detection layers, the semantic information is already learned from previous layers; so it is not necessary to employ the segmentation branch for them. Further, since the resolution is smaller in higher levels, it will become harder to do the segmentation task based on them. Due to these reasons, we employ simple global activation blocks, on *conv4_3* through *conv9_2*, to enrich their semantic information in a self-supervised manner. The details are illustrated in Subsection 4.3.2.

4.3.1 Semantic Enrichment at Low-level Layer

Semantic enrichment at low-level detection feature maps is achieved by a segmentation branch, which performs weakly supervised semantic segmentation. It takes the low-level detection layer from the detection branch (*e.g.* *conv4_3* for VGG16) as inputs, and generates a semantically meaningful feature map with the same dimension. Then this feature map is used to activate the input low-level detection layer from the detection branch by element-wise multiplication.

Mathematically, let $X \in \mathbb{R}^{C\times H\times W}$ be the low-level detection feature map retrieved from the detection branch (*e.g.* *conv4_3*), and $G \in \{0, 1, 2, \cdots, N\}^{H\times W}$ be the segmentation ground-truth where N is the number of classes (20 for Pascal VOC and 80 for MS COCO). The segmentation branch computes $Y \in \mathbb{R}^{(N+1)\times H\times W}$ as the prediction of pixel-wise segmentation where

$$Y = \mathcal{F}(\mathcal{G}(X))$$

satisfying

$$Y_{c,h,w} \in [0, 1], \text{ and } \sum_{c=0}^{N} Y_{c,h,w} = 1.$$

$\mathcal{G}(X) \in \mathbb{R}^{C'\times H\times W}$ is the intermediate result which will be further used to generate

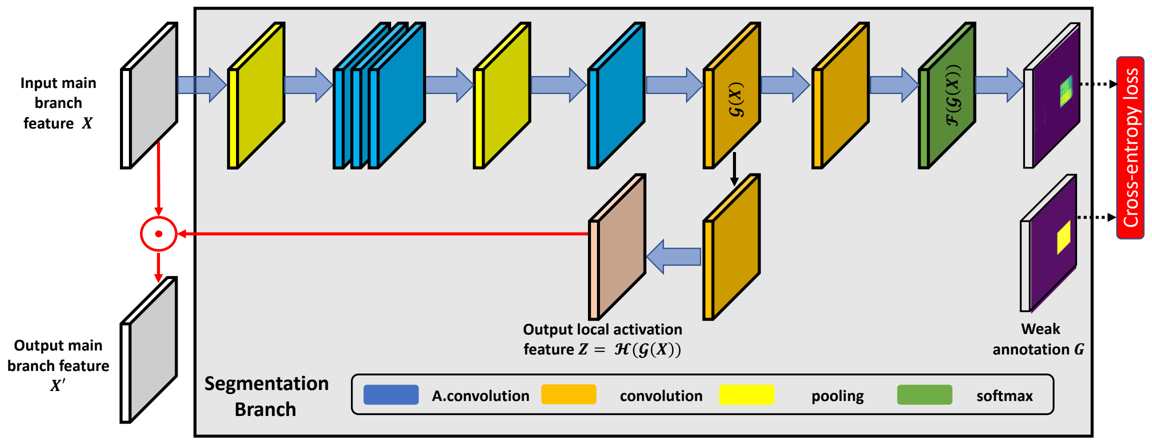
Input main branch feature X
$\mathcal{G}(X)$
$\mathcal{F}(\mathcal{G}(X))$
Cross-entropy loss
Output main branch feature X'
Output local activation feature $Z = \mathcal{H}(\mathcal{G}(X))$
Weak annotation G
Segmentation Branch
A.convolution
convolution
pooling
softmax

Figure 4.4. The architecture of the semantic segmentation branch. It takes an intermediate feature map from object detection branch (*e.g.* *conv4_3* for SSD300) as input (as shown in the grey cube on the upper left corner), and generates a semantically meaningful feature map Z (as shown in the tan cube in the center) to enrich the input X to X' (as shown in the grey cube on the lower left corner). X' is then used in the detection branch, to replace the original feature map X.

semantic meaningful feature map:

$$Z = \mathcal{H}(\mathcal{G}(X)) \in \mathbb{R}^{C \times H \times W}.$$

The semantic meaningful feature map Z is then used to activate the original low-level detection feature map X by element-wise multiplication: $X' = X \odot Z$, where X' is the semantically activated low-level detection feature map which conveys both basic visual patterns and high level semantic information. X' will replace the original X in the detection branch for object detection, as represented by the left-most red arrow in Figure 4.2. Figure 4.4 gives an detailed illustration of the architecture of the segmentation branch.

For the segmentation branch, we design a simple network architecture mainly composed of atrous convolutional layers [12]. We add four atrous convolutional layers (noted as 'A. convolution' in Figure 4.4) with 3×3 kernel size after the input feature map X. The first three atrous convolutional layers have a dilation rate of 2 and the last atrous convolutional layer has a dilation rate of 4. After that we deploy another

1×1 convolutional layer to generate the aforementioned feature map $\mathcal{G}(X)$. This intermediate feature map is used in two ways: to generate semantic segmentation prediction $Y = \mathcal{F}(\mathcal{G}(X))$ and to provide strong semantic information to activate the input feature map $X' = X \odot \mathcal{H}(\mathcal{G}(X))$. Towards this end, there are two paths attached to $\mathcal{G}(X)$. The first path ($\mathcal{F}$ path) takes a 1×1 convolution layer with $N + 1$ output channels and a softmax layer to generate the segmentation prediction Y. The second path ($\mathcal{H}$ path) takes another 1×1 convolution layer whose output channel number equals to the channel number of X, to generate a semantically meaningful feature map Z in order to activate the feature map in the detection branch by element-wise multiplication. We show an example of this activation process in Figure 4.3. Column A is the input image and column B shows two slices of the original low-level object detection feature map X. We can notice that the semantic meaningful feature map Z generated by our segmentation branch can capture very high level semantic information (the dog or human information). The final activated feature map X' conveys both basic visual pattern and high level semantic information. All these layers keep the size of feature maps unchanged.

The final issue is how to generate pseudo semantic segmentation ground-truth annotations given only the object bounding-boxes. The segmentation ground-truth G should have the same resolution as the input layer of segmentation branch (*conv4_3* for SSD300). We take a simple strategy to generate it: if a pixel G_{hw} locates within a bounding-box on the image lattice I, we assign the label of that bounding-box to G_{hw}; if it locates within more than one bounding-boxes, we choose the label of the bounding-box with the smallest size; and if it does not locate inside any bounding-box, we assign it to the background class. This simple strategy guarantees that there is only one class to be assigned to each pixel in G. We show an example of such weak segmentation ground-truth labels in Figure 4.5.

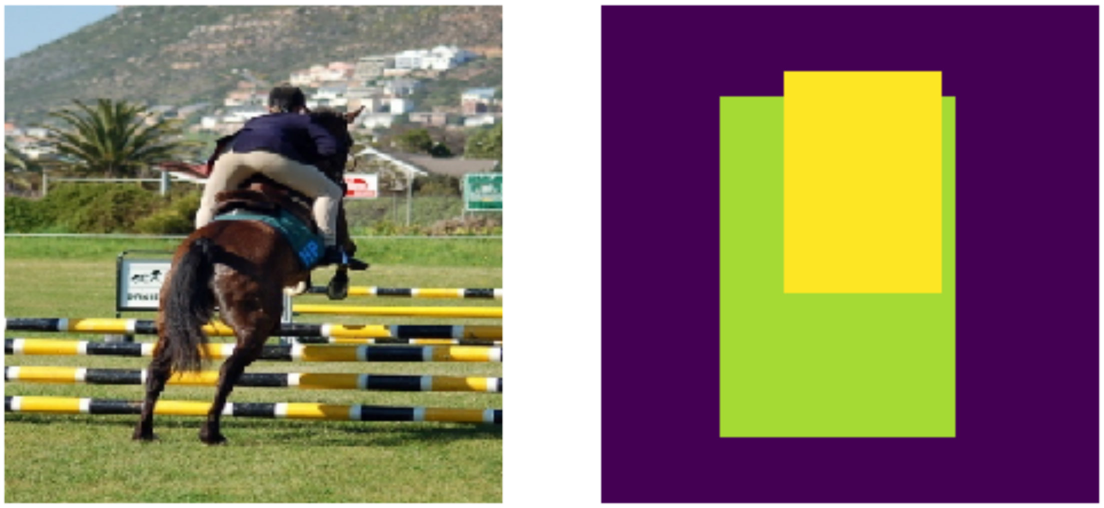

Figure 4.5. Example of weak (pseudo) segmentation ground-truth. Left: Input image with a size of 300×300 pixels, with a person and a horse on the image. Right: Weak segmentation ground-truth for the left image, with a size of 38×38; the pixels locate inside both the person and the horse bounding-boxes will be assigned to person class since its bounding-box is smaller.

4.3.2 Semantic Enrichment at Higher-level Layers

In conjunction with our segmentation branch, we propose another module named global activation module at higher layers. It contains several global activation blocks, attached at each object detection feature map in the detection branch. Global activation blocks can learn the correspondence between feature channels and object classes, after eliminating the spatial information, in a self-supervised manner.

Each global activation block consists of three stages: spatial average pooling, channel-wise learning and broadcasted multiplying. Mathematically, given the input feature map $X \in \mathbb{R}^{C \times H \times W}$, the spatial average pooling stage will produce $Z \in \mathbb{R}^{C}$ by

$$Z_i = \frac{1}{HW} \sum_{h,w} X_{ihw}$$

and the channel-wise learning stage will generate the activation feature

$$S = \text{Sigmoid}(W_2 \cdot \text{ReLU}(W_1 Z)) \in \mathbb{R}^{C \times 1 \times 1}$$

where $W_2 \in \mathbb{R}^{C \times C'}, W_1 \in \mathbb{R}^{C' \times C}$, used to magnify the channels which deemed useful for detection. The W_1 and W_2 are learnable weights of two fully-connected layers. Finally, in the broadcasted multiplying stage, S is used to activate X to get $X' \in \mathbb{R}^{C \times H \times W}$ by

$$X'_{ihw} = X_{ihw} \cdot S_i$$

The activated feature map X' will replace the original X in the detection branch. In our experiments, we keep $C' = \frac{1}{4}C$ for all global activation blocks. This architecture was originally designed for image classification in [67], and here we extend it for object detection.

4.3.3 Multi-task Training

In the training phase, two losses are combined for computing gradients. The first loss is the original detection loss, $L_{\text{det}}(I, B)$ where I is the input image and B is the bounding-box annotations. As defined in [17], it is computed on all matched anchor boxes, and considers both the classification and regression of anchor boxes. The second loss is an extra cross-entropy loss function for the semantic segmentation task. It is formulated as:

$$L_{\text{seg}}(I, G) = -\frac{1}{HW} \sum_{h,w} \log(Y_{G_{h,w},h,w})$$

where $Y \in [0, 1]^{(N+1) \times H \times W}$ is the semantic segmentation predictions, and $G \in \{0, 1, 2, \cdots, N\}^{H \times W}$ is the semantic segmentation ground-truth generated by bounding-box annotation. N is the number of classes excluding background class (20 for Pascal VOC and 80 for MS COCO). The detailed definition of Y and G could be found in the Subsection 4.3.1.

By summing up the new segmentation loss function and the original detection loss function, the final objective function we are optimizing is:

$$L(I, B, G) = L_{\text{det}}(I, B) + \alpha L_{\text{seg}}(I, G)$$

where α is a parameter to balance those two tasks.

4.4 Experiments

We present comprehensive experiment results on two major object detection datasets: Pascal VOC [7] and MS COCO [3]. For the Pascal VOC, we follow the conventional split, which uses the union of VOC2007 `trainval` and VOC2012 `trainval` as the training data, and uses VOC2007 `test` as the testing data. We also show the experiment result on VOC2012 `test` with the model trained on the union of VOC2007 `trainvaltest` and VOC2012 `trainval`. For MS COCO, we use a popular split which takes `trainval35k` [68] for training, `minival` for validation, and we show results on `test-dev2017` which is evaluated on the official evaluation server.

For the backbone object detection framework, we choose VGG16-based SSD300 [17] and SSD512 as our single-stage detection branch. Note that SSD has been updated with new data augmentation strategies which boost the performance by a huge margin, and we follow the latest version of SSD with all those improvements. The semantic segmentation branch is inserted at the first detection feature map layer, *i.e.* *conv4_3* for both SSD300 and SSD512. The global activation module consists of several global activation blocks, 6 for SSD300 and 7 for SSD512, and all of those blocks are added at each detection feature map. For the first detection feature map, the segmentation branch is inserted before the global activation block. We follow the original SSD training strategy throughout our experiments, and set the trade-off hyperparameter α (introduced in Subsection 4.3.3) to be 0.1.

We will use the terminologies 'DES300' and 'DES512' to represent our Detection with Enriched Semantics network built on VGG16-based SSD300 and SSD512 respectively in the rest of this chapter for simplicity.

4.4.1 Experiment on VOC

		mAP	aero	bike	bird	boat	bottle	bus
method	backbone	car	cat	chair	cow	table	dog	horse
		mbike	persn	plant	sheep	sofa	train	tv
		70.0	77.0	78.1	69.3	59.4	38.3	81.6
Fast-RCNN [20]	VGG16	78.6	86.7	42.8	78.8	68.9	84.7	82.0
		76.6	69.9	31.8	70.1	74.8	80.4	70.4
		73.2	76.5	79.0	70.9	65.5	52.1	83.1
Faster-RCNN [26]	VGG16	84.7	86.4	52.0	81.9	65.7	84.8	84.6
		77.5	76.7	38.8	73.6	73.9	83.0	72.6
		76.4	79.8	80.7	76.2	68.3	55.9	85.1
Faster-RCNN [8]	ResNet101	85.3	**89.8**	56.7	87.8	69.4	88.3	88.9
		80.9	78.4	41.7	78.6	79.8	85.3	72.0
		80.5	79.9	87.2	81.5	72.0	**69.8**	86.8
R-FCN [28]	ResNet101	88.5	**89.8**	**67.0**	**88.1**	74.5	**89.8**	**90.6**
		79.9	81.2	53.7	81.8	81.5	85.9	79.9
		77.6	86.0	82.5	76.9	69.1	59.2	86.2
RON384++ [66]	VGG16	85.5	87.2	59.9	81.4	73.3	85.9	86.8
		82.2	79.6	52.4	78.2	76.0	86.2	78.0
		78.2	80.3	84.1	78.5	70.8	68.5	88.0
Gidaris[64]	VGG16	85.9	87.8	60.3	85.2	73.7	87.2	86.5
		85.0	76.4	48.5	76.3	75.5	85.0	81.0
		76.4	79.3	80.5	76.8	72.0	58.2	85.1
Shrivastava[65]	VGG16	86.5	89.3	60.6	82.2	69.2	87.0	87.2
		81.6	78.2	44.6	77.9	76.7	82.4	71.9
		77.5	79.5	83.9	76.0	69.6	50.5	87.0
SSD300 [17]	VGG16	85.7	88.1	60.3	81.5	77.0	86.1	87.5
		84.0	79.4	52.3	77.9	79.5	87.6	76.8
		77.1	76.3	84.6	79.3	64.6	47.2	85.4
SSD321 [17]	ResNet101	84.0	88.8	60.1	82.6	76.9	86.7	87.2
		85.4	79.1	50.8	77.2	**82.6**	87.3	76.6
		79.7	83.5	86.0	78.1	74.8	53.4	87.9
DES300 (Ours)	VGG16	87.3	88.6	64.0	83.8	77.2	85.9	88.6
		87.5	80.8	57.3	80.2	80.4	**88.5**	79.5
		79.5	84.8	85.1	81.5	73.0	57.8	87.8
SSD512 [17]	VGG16	88.3	87.4	63.5	85.4	73.2	86.2	86.7
		83.9	82.5	55.6	81.7	79.0	86.6	80.0
		80.6	84.3	**87.6**	82.6	71.6	59.0	88.2
SSD513 [17]	ResNet101	88.1	89.3	64.4	85.6	76.2	88.5	88.9
		87.5	83.0	53.6	83.9	82.2	87.2	**81.3**
		81.7	**87.7**	86.7	**85.2**	**76.3**	60.6	**88.7**
DES512 (Ours)	VGG16	**89.0**	88.0	**67.0**	86.9	**78.0**	87.2	87.9
		87.4	**84.4**	**59.2**	**86.1**	79.2	88.1	80.5

Table 4.1. Results on VOC2007 `test`. The first section contains some representative baselines [8], [20], [26], [28], [66], the second section contains other detectors exploiting segmentation information [64], [65], the third section contains low resolution SSD and DES, and the last section contains high resolution SSD and DES. Note that all these methods are trained on VOC2007 `trainval` and VOC2012 `trainval`.

For the Pascal VOC dataset, we run the training on a machine with 2 Titan Xp GPUs. In order to focus on the effectiveness of our DES network, we keep the training settings unchanged as used in SSD. We first train the model with a learning rate of 10^{-3} for 80k iterations, and then continue the training with a learning rate of 10^{-4} for 20k iterations and 10^{-5} for another 20k iterations. The momentum is fixed to be 0.9 and the weight decay is set to be 0.0005. Those hyperparameters are aligned with the original SSD experiments. We use pre-trained SSD model on Pascal VOC to initialize our model, and initialize the parameters in the first five layers of segmentation branch with the parameters of *conv5_1*, *conv5_2*, *conv5_3*, *fc_6* and *fc_7* in the detection branch. The rest two convolutional layers of the segmentation branch are initialized by Xavier initialization [69]. We also do another experiment by resetting all the parameters after *conv6_1* layer in the detection branch with Xavier initialization. This will lead to similar results compared with the current setting.

The results on VOC2007 `test` are shown in Table 4.1. We can see DES outperforms original SSD on both resolution settings, and it improves the mAP from 77.5 to 79.7 and from 79.5 to 81.7 for the low (300) and the high (512) resolution inputs respectively. Our VGG16-based detectors can even significantly outperform ResNet101-based SSD detectors, which are much deeper and more complex compared with VGG16 backbones, and this comparison highlights the effectiveness of our method. Compared with other baselines such as popular two-stage methods and other detector combined with segmentation, our DES also shows a significant performance improvement. We show the results on VOC2012 `test` in Table 4.2, and the same tendency remains. DES outperforms all the competitors with a large gap.

method	backbone	mAP	aero	bike	bird	boat	bottle	bus
		car	cat	chair	cow	table	dog	horse
		mbike	persn	plant	sheep	sofa	train	tv
		73.8	86.5	81.6	77.2	58.0	51.0	78.6
Faster-RCNN [8]	ResNet101	76.6	93.2	48.6	80.4	59.0	92.1	85.3
		84.8	80.7	48.1	77.3	66.5	84.7	65.6
		77.6	86.9	83.4	**81.5**	63.8	62.4	81.6
R-FCN [28]	ResNet101	81.1	93.1	58.0	83.8	60.8	92.7	86.0
		84.6	84.4	**59.0**	80.8	68.6	86.1	72.9
		75.4	86.5	82.9	76.6	60.9	55.8	81.7
RON384++ [66]	VGG16	80.2	91.1	57.3	81.1	60.4	87.2	84.8
		84.9	81.7	51.9	79.1	68.6	84.1	70.3
		73.9	85.5	82.9	76.6	57.8	**62.7**	79.4
Gidaris [64]	VGG16	77.2	86.6	55.0	79.1	62.2	87.0	83.4
		84.7	78.9	45.3	73.4	65.8	80.3	74.0
		72.6	84.0	81.2	75.9	60.4	51.8	81.2
Shrivastava [65]	VGG16	77.4	90.9	50.2	77.6	58.7	88.4	83.6
		82.0	80.4	41.5	75.0	64.2	82.9	65.1
		75.8	88.1	82.9	74.4	61.9	47.6	82.7
SSD300 [17]	VGG16	78.8	91.5	58.1	80.0	64.1	89.4	85.7
		85.5	82.6	50.2	79.8	73.6	86.6	72.1
		75.4	87.9	82.9	73.7	61.5	45.3	81.4
SSD321 [17]	ResNet101	75.6	92.6	57.4	78.3	65.0	90.8	86.8
		85.8	81.5	50.3	78.1	75.3	85.2	72.5
		77.1	88.5	84.4	76.0	65.0	50.1	83.1
DES300 (Ours)[1]	VGG16	79.7	92.1	61.3	81.4	65.8	89.6	85.9
		86.2	83.2	51.2	81.4	**76.0**	88.4	73.3
		78.5	90.0	85.3	77.7	64.3	58.5	85.1
SSD512 [17]	VGG16	84.3	92.6	61.3	83.4	65.1	89.9	88.5
		88.2	85.5	54.4	82.4	70.7	87.1	75.6
		79.4	90.7	87.3	78.3	66.3	56.5	84.1
SSD513 [17]	ResNet101	83.7	**94.2**	62.9	**84.5**	66.3	**92.9**	88.6
		87.9	85.7	55.1	83.6	74.3	88.2	76.8
		80.3	**91.1**	**87.7**	81.3	**66.5**	58.9	**84.8**
DES512 (Ours)[2]	VGG16	**85.8**	92.3	**64.7**	84.3	**67.8**	91.6	**89.6**
		88.7	**86.4**	57.7	**85.5**	74.4	**89.2**	**77.6**

Table 4.2. Results on VOC2012 `test`. Note that all methods in this table are trained on VOC2007 `trainvaltest` and VOC2012 `trainval`, except Gidaris [64] is trained on VOC2007 `trainval` and VOC2012 `trainval`.

method	backbone	07 `test`	12 `test`
SSD300 [17]	VGG16	79.8	78.5
DES300	VGG16	82.7	81.0[3]
SSD512 [17]	VGG16	83.2	82.2
DES512	VGG16	84.3	83.7[4]

Table 4.3. Results on VOC2007 `test` and VOC2012 `test` when detectors are fine-tuned from models pre-trained on COCO.

Table 4.3 summarizes the results when SSD and DES are fine-tuned from models trained on MS COCO, which is a much larger dataset. DES outperforms SSD on all experiment settings with a large margin. It suggests our method can also get benefit from extra training data like the MS COCO dataset.

4.4.2 Experiment on COCO

We use the similar strategy for the MS COCO dataset. The DES is implemented based on the original SSD detectors which have slightly different anchor box settings to fit the MS COCO dataset. The training is conducted on the `trainvel35k` subset generated from MS COCO `trainval2014` dataset. We first train the network with a learning rate of 10^{-3} for 280k iterations, followed by training it with a learning rate of 10^{-4} for 80k iteration and 10^{-5} for another 40k iteration. The momentum is set to be 0.9 and the weight decay is set to be 0.0005, which are consistent with the original SSD settings.

Similar to our methods used for Pascal VOC, we use the pre-trained SSD model on MS COCO to initialize our parameters, and use weights in $conv5\_1$, $conv5\_2$, $conv5\_3$, fc_6 and fc_7 to initialize the first five layers in the segmentation branch.

[1]http://host.robots.ox.ac.uk:8080/anonymous/RCMS6B.html
[2]http://host.robots.ox.ac.uk:8080/anonymous/OBE3UF.html
[3]http://host.robots.ox.ac.uk:8080/anonymous/IRJJ5L.html
[4]http://host.robots.ox.ac.uk:8080/anonymous/MURP2C.html

method	backbone	data	mAP	AP50	AP75	APs	APm	APl
Faster [26]	VGG16	trainval	21.9	42.7	-	-	-	-
Faster+++ [8]	ResNet101	trainval	34.9	55.7	-	-	-	-
R-FCN [28]	ResNet101	trainval	29.9	51.9	-	10.8	32.8	45.0
RON384++ [66]	VGG16	trainval	27.4	49.5	27.1	-	-	-
Shrivastava [65]	VGG16	trainval35k	27.5	49.2	27.8	8.9	29.5	41.5
SSD300 [17]	VGG16	trainval35k	25.1	43.1	25.8	6.6	25.9	41.4
SSD321 [17]	ResNet101	trainval35k	28.0	45.4	29.3	6.2	28.3	49.3
DES300 (Ours)	VGG16	trainval35k	28.3	47.3	29.4	8.5	29.9	45.2
SSD512 [17]	VGG16	trainval35k	28.8	48.5	30.3	10.9	31.8	43.5
SSD513 [17]	ResNet101	trainval35k	31.2	50.4	33.3	10.2	34.5	49.8
DES512 (Ours)[5]	VGG16	trainval35k	32.8	53.2	34.6	13.9	36.0	47.6

Table 4.4. Results of average precision (AP) on COCO `test-dev`. 'APs', 'APm' and 'APr' stand for the AP for small, medium and large objects respectively, and 'mAP', 'AP50' and 'AP75' mean average precision of IOU=0.5:0.95, IOU=0.5 and IOU=0.75 respectively. `trainval35k` is obtained by removing the 5k `minival` set from `trainval`.

However, different from VOC, we find that resetting weights after *conv*6_1 is crucial for good performance, and we can only get a small improvement around 0.2 for AP@0.5 if we keep those weights after *conv*6_1 same as the SSD pre-trained model.

method	backbone	data	AR1	AR10	AR100	ARs	ARm	ARl
Faster [26]	VGG16	trainval	-	-	-	-	-	-
Faster+++ [8]	ResNet101	trainval	-	-	-	-	-	-
R-FCN [28]	ResNet101	trainval	-	-	-	-	-	-
RON384++ [66]	VGG16	trainval	-	-	-	-	-	-
Shrivastava [65]	VGG16	trainval35k	25.5	37.4	38.3	14.6	42.5	57.4
SSD300 [17]	VGG16	trainval35k	23.7	35.1	37.2	11.2	40.4	58.4
SSD321 [17]	ResNet101	trainval35k	25.9	37.8	39.9	11.5	43.3	64.9
DES300 (Ours)	VGG16	trainval35k	25.6	38.3	40.7	14.1	44.7	62.0
SSD512 [17]	VGG16	trainval35k	26.1	39.5	42.0	16.5	46.6	60.8
SSD513 [17]	ResNet101	trainval35k	28.3	42.1	44.4	17.6	49.2	65.8
DES512 (Ours)	VGG16	trainval35k	28.4	43.5	46.2	21.6	50.7	64.6

Table 4.5. Results of average recall (AR) on COCO `test-dev`. 'ARs', 'ARm' and 'ARl' stand for the AR for small, medium and large objects respectively

We report results on COCO `test-dev2017` which contains 20288 images and evaluated on the official evaluation server deployed on CodaLab in Table 4.4 and Table 4.5.

[5]https://competitions.codalab.org/competitions/5181#results

Compared with baseline SSD detectors, our DES can provide huge improvement on all of the evaluation metrics. For the low (300) resolution version (the third block in the table), we can achieve a relative improvement of 12.7% for mAP compared with baseline SSD300, from 25.1 to 28.3, and a significant relative improvement of 28.8% for small objects. For the high resolution version (the fourth block in the table), DES can improve the baseline from 28.8 to 32.8. Our DES can also outperform SSD based on ResNet101, which is deeper and much slower.

We can find that DES performs much better on small objects, outperforms at least 27.5% relatively compared with all other competitors which report performance on small objects. Although DES512 outperforms SSD512 based on VGG16 for detecting large objects, it is slightly worse than SSD513 based on ResNet101 for detecting large objects. We argue that SSD513 can benefit from a better backbone ResNet101 which is much deeper, to detect large objects.

4.5 Discussion and Ablation

4.5.1 Architecture Ablation and Diagnosis

To further understand the effectiveness of our two extra modules (*i.e.*, semantic segmentation branch and the global activation module), we conduct ablation experiments with different settings and report the results in Table 4.6 on the VOC2007 `test` dataset based on DES300.

As can be seen from Table 4.6, the global activation module (G) can improve the performance by 0.6, which confirms the effectiveness of the global activation with global activation features. With the segmentation branch (S) introduces, the performance can be further improved by a large margin, which confirms our intuition that segmentation can be used to help object detection, and introducing high-level semantic knowledge to the early stage of the detection network can contribute to a

method	mAP
SSD300	77.5
SSD300+G	78.1
SSD300+G+S ($\alpha = 0.0$)	79.4
SSD300+G+S ($\alpha = 0.1$)	**79.7**
SSD300+G+S ($\alpha = 1.0$)	78.6
SSD300+G+S (in parallel)	78.2
SSD300+G+DeeperVGG16	77.6

Table 4.6. Ablation result evaluated on VOC2007 `test` dataset. G stands for the global activation module and S stands for the segmentation branch. α is the hyper-parameter controlling the tradeoff between segmentation loss and detection loss discussed in Subsection 4.3.3.

stronger object detector.

Another ablation study conducted is the weight of the segmentation loss. To do this, we train our DES network for VOC2007 `test` task with different α's, *i.e.*, 0, 0.1 and 1. This hyperparameter plays an important role for balancing the object detection loss and the semantic segmentation loss. Experiments shows that $\alpha = 0.1$ yields the best performance, 0.3 better than $\alpha = 0$ (eliminating segmentation loss) and 1.1 better than $\alpha = 1$ (taking the tasks of object detection and segmentation equally important). This means the supervision over the segmentation task is useful in our segmentation branch. But it should take less weight than the detection loss since the main task is object detection instead of segmentation, otherwise the backbone features would lean towards the segmentation task too much and hurt the detection performance.

To further justify the effectiveness of our segmentation branch architecture, we conduct another two experiments. In the first experiment, we mimic Mask-RCNN [13] by training segmentation and detection branches in parallel, in stead of using segmentation features to activate low-level detection features. The improvement is very small (mAP of 78.2 as shown in the 6th row in Table 4.6) and we believe the low-level feature activation process is very important to improve detection features, and since our weak segmentation ground-truth does not contain extra information, it will not improve the backbone feature quality and the detection accuracy significantly if trained in parallel. As a side evidence, we train a modified version of Mask-RCNN, with only pseudo (weak) segmentation supervision, and the performance only goes up by a small amount of 0.6 on COCO `minival`. This result indicates that Mask-RCNN cannot get a huge benefit from the weak segmentation supervision trained in parallel, and confirms our observations on DES.

The second justification experiment we do is removing the segmentation loss and the low-level feature activation process, while keeping the extra weights and layers involved in the semantic segmentation branch. Then the feature map $Z = \mathcal{H}(\mathcal{G}(X))$ is directly used by the object detection branch. This modification keeps the number of parameters introduced by our segmentation branch, and can be regraded as a 'deeper VGG16' with more parameters as the backbone. This architecture achieves an mAP of 77.6, which is much lower compared with our proposed method. This confirms the architecture design and the feature enrichment process of our segmentation branch is crucial, and the performance cannot be improved by naively adding more layers and parameters.

4.5.2 Inference Speed

To quantitatively evaluate the inference speed, we compile and run DES, SSD, as well as R-FCN, on our machine with an nVIDIA Titan Xp GPU to compare the speed fairly.

All results are summarized in Table 4.7. Note that to make comparison fair, we keep the batch-size to be the same in each comparison group (*i.e.* low resolution group based on SSD300 and the high resolution group based on SSD512). For ResNet101 based SSD321 and SSD513, we remove the batch normalization layer at the test time to reduce the run time and memory consumption following [70].

Our method is a bit slower compared with original VGG16-based SSD due to our

method	backbone	mAP	time (ms/img)	FPS	batch-size
R-FCN [28]	ResNet101	80.5	89.6	11.2	1
SSD300 [17]	VGG16	77.5	9.2	109.3	8
SSD321 [17]	ResNet101	77.1	33.2	30.2	8
DES300	VGG16	79.7	13.0	76.8	8
SSD512 [17]	VGG16	79.5	18.6	53.8	8
SSD513 [17]	ResNet101	80.6	61.6	16.2	8
DES512	VGG16	81.7	31.5	31.7	8

Table 4.7. Inference Speed of two-shot baseline R-FCN and single-stage SSD and DES under different input resolutions. Here we report the mAP on VOC2007 `test` dataset in the mAP column, the time spent for inferring a single image in milliseconds in the time column, as well as the number images processed within one second in the FPS column.

extra modules, however, DES is faster than ResNet101-based SSD by a large margin, and outperforms it in terms of accuracy at the same time. DES300 has an FPS of 76.8 with an mAP of 79.7, while DES512 achieves higher mAP with a lower FPS (*i.e.*, 81.7 and 31.7 respectively).

4.5.3 Qualitative Examples

We show some detection examples in Figure 4.6. The left column shows the results of original SSD300, and the right column shows the results of our DES300. We show 'aeroplane' in the first two rows, and 'pottedplant' in the last row, for all detection results with a score higher than or equal to 0.3. The detected objects are highlighted with yellow bounding-boxes and the corresponding confidence scores are shown on the upper-right side of the bounding-boxes. From these examples, we can see that our method is good at detecting small objects like small aeroplanes and pottedplants, and it can also prune out some false positives which are incorrectly detected as aeroplane shown in the first row.

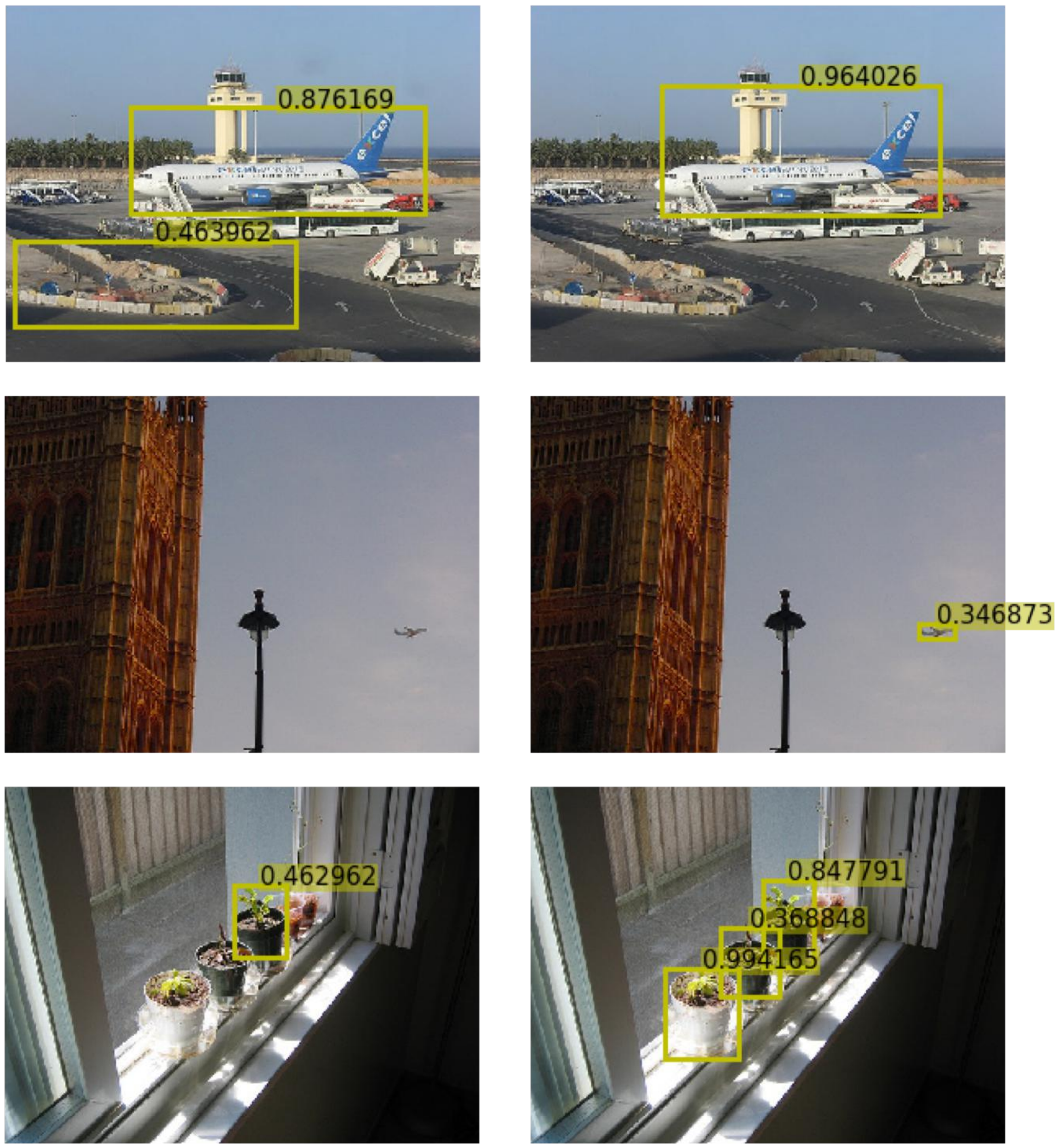

Figure 4.6. Examples of detection results. Left: Results of SSD300. Right: Results of DES300. All detected objects with confidence scores higher than or equal to 0.3 are highlighted in yellow bounding-boxes, and the corresponding confidence scores are printed at the upper-right side of the bounding-boxes.

4.6 Conclusion and Future Works

In this chapter, we propose a novel single-stage object detector named Detection with Enriched Semantics (DES). Following previous works, it builds a detection feature pyramid for detection objects with different scales in an image. To address the problem that low-level detection feature map does not contain high-level semantic information, we introduce a semantic segmentation side branch, which utilizes the idea of weakly supervised semantic segmentation, to provide high semantically meaningful and class-aware features to activate and enrich feature map used for the object detection. We also utilize a global activation module to provide global information and learn channel-wise attention. Our method is flexible and simple, and does not require a huge redesign of the original popular single-stage detection frameworks. Quantitative evaluation on both Pascal VOC and MS COCO shows our method excels in both accuracy and speed, compared with several baseline detectors. Our method can also be applied to other two-stage or single-stage object detectors, with stronger backbone, and we remain this as future work.

The current design introduces some computation overheads (although small) during the inference. Thus one direction of future improvement is to learn semantically strong detection feature maps without extra neural network layers and computation overheads. There are some later works (*e.g.* [71]) exploit the semantics enriching for training only and keep the inference unchanged without compromising the speed. Another direction of future works is to learn detection and segmentation simultaneously, rather than making use of pseudo segmentation. Object detection and semantic segmentation are highly related, and they are potentially mutually beneficial. This direction is partially addressed by [72].

Chapter 5

Online Hard Image Mining for Long-tailed Hard Cases

Among all the detection tasks, face detection is one of the most important task. Analyzing faces is an important application and detecting the them is the first step towards this direction, and it is also a backbone for many down-streaming face-related tasks. In this chapter, we move our focus from general object detection to face detection, which could have different accuracy requirements and different challenges. Recently anchor-based deep face detectors have achieved promising performance, but they are still struggling to detect hard faces, such as small, blurred and partially occluded faces. One reason is that their training treats all images and faces equally, and ignores the imbalance between easy images and hard images; however large amounts of training images only contain easy faces, which are less helpful to learn robust detectors for hard faces. Thus, we propose that the robustness of a face detector against hard faces can be improved by learning small faces on hard images. Our intuitions are (1) hard images are the images which contain at least one hard face, thus they facilitate training robust face detectors; (2) most hard faces are small faces and other types of hard faces can be easily shrunk to small faces. To this end, we build an anchor-based deep face detector, which only outputs a single high-resolution feature map with small anchors, to specifically learn small faces and train it by a novel hard image mining strategy which automatically adjusts training weights on images according to their difficulties. Extensive experiments have been conducted on WIDER FACE [5], FDDB [73], Pascal Faces [74], and AFW [75] datasets and our method achieves APs of 95.7, 94.9 and 89.7 on `easy`, `medium` and `hard` WIDER FACE `val` subsets respectively, which verify the effectiveness of our methods, especially on detecting hard faces. Our detector is also lightweight and enjoys a fast inference speed. Code and model are available at https://github.com/bairdzhang/smallhardface.

5.1 Motivation and Overview

Face detection is a fundamental and important computer vision problem, which is critical for many face-related tasks, such as face alignment [76], [77], tracking [78] and recognition [79], [80]. Stem from the recent success of deep neural networks, massive CNN-based face detection approaches [32], [81]–[85] have been proposed and achieved state-of-the-art performance. However, face detection remains a challenging task due to occlusion, illumination, makeup, as well as pose and scale variance, as shown in the benchmark dataset WIDER FACE [5].

Current state-of-the-art CNN-based face detectors attempt to address these challenges by employing more powerful backbone models [86], exploiting feature pyramid-style architectures to combine features from multiple detection feature maps [82], designing denser anchors [85] and utilizing larger contextual information [82]. These methods and techniques have been shown to be successful to build a robust face detector, and improve the performance towards human-level for most images.

In spite of their success for most images, an evident performance gap still exists especially for those hard images which contain small, blurred and partially occluded faces. We realize that these hard images have become the main barriers for face detectors to achieve human-level performance. In Figure 5.1, we show that, even on

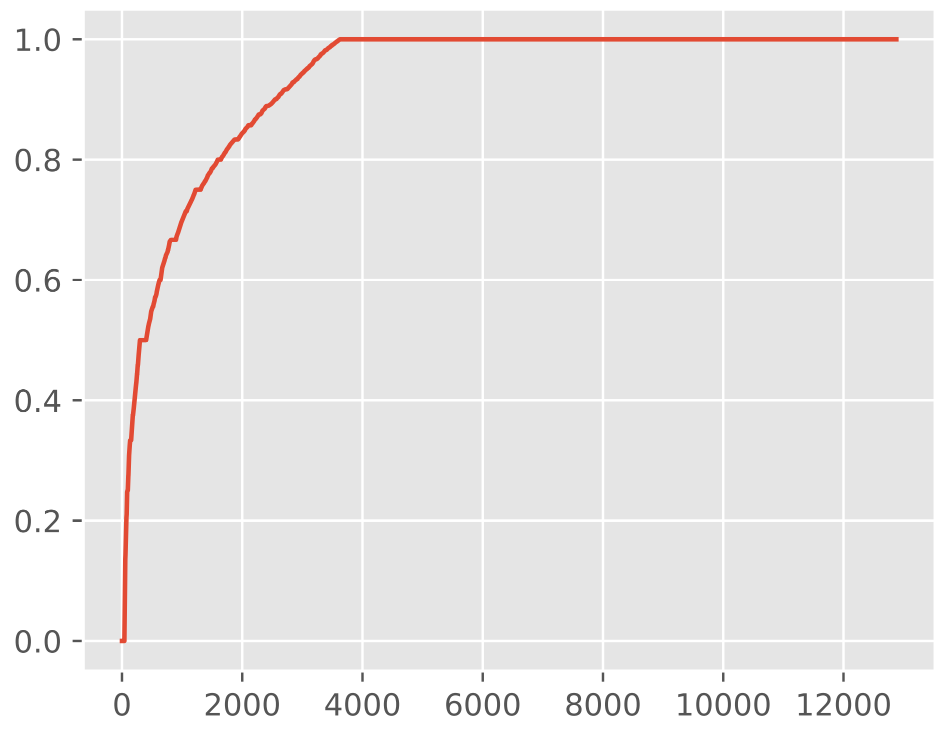

Figure 5.1. AP of each training image computed based on official SSH model, the x-axis is index of the training image, the y-axis is the AP for the corresponding image. The images are sorted in descending order of APs.

the `train` set of WIDER FACE, the official SSH model[1] still fails on some of the images with extremely hard faces. We show two such hard training images in the upper side in Figure 5.2.

On the contrary, most training images with easy faces can be almost perfectly detected (see the illustration in the lower right corner of Figure 5.2). As shown in Figure 5.1, over two-thirds of the training images already obtained perfect detection accuracy, and we believe that those easy images are less useful towards training a robust face detector. To address this issue, in this paper, we propose a robust face detector by putting more training efforts on those hard images.

This problem is mostly related to anchor-level hard example mining discussed in OHEM [87]. OHEM aims at mining hard anchors/proposals in each image during the training phase. However, when the majority of the training set are easy images

[1]https://github.com/mahyarnajibi/SSH

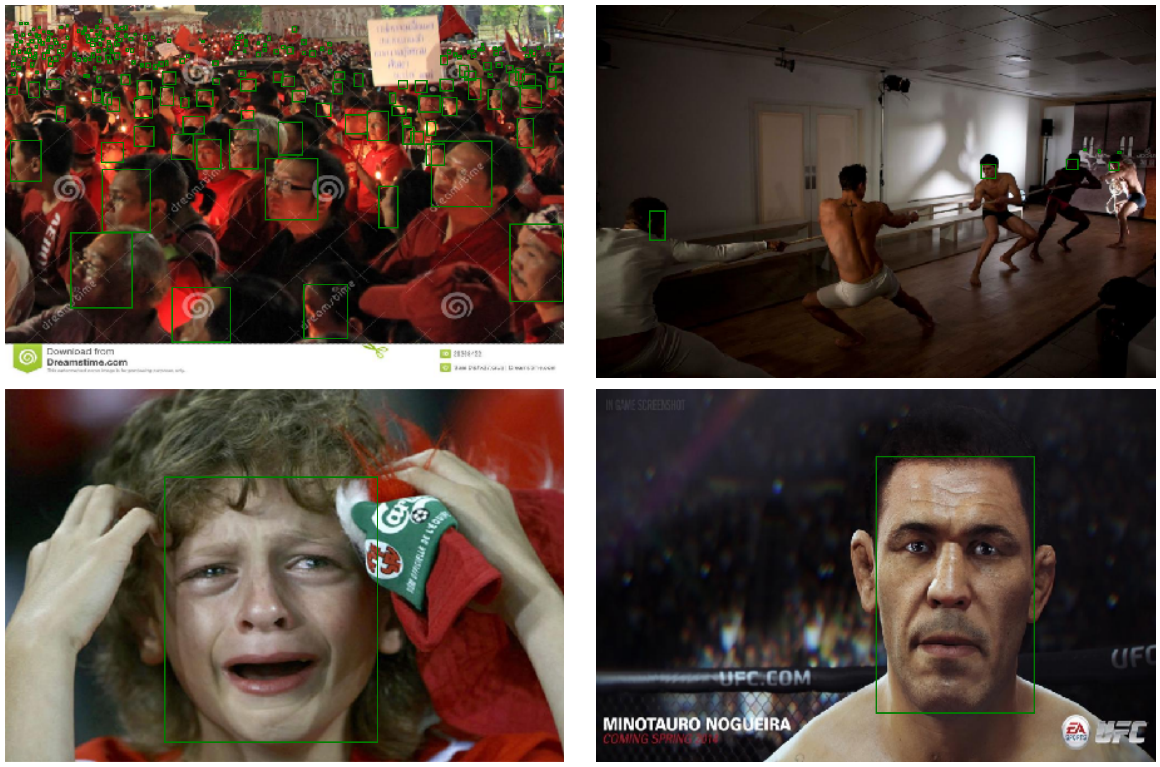

Figure 5.2. Upper half: two hard training images. Lower half: two easy training images.

with no hard anchors/proposals (see Figure 5.1), OHEM will exhibit less effectiveness since less useful information can be exploited towards a more robust model. Also, due to the sparsity of faces, OHEM mainly focuses on mining hard negative anchors and takes all faces equally into training. To this end, we propose to mine hard positive examples at image level (*i.e.* hard image mining, HIM) in conjunction with anchor level OHEM. More specifically, we propose to dynamically assign difficulty scores to training images during the learning process, which can determine whether an image is already well-detected or still useful for further training. This allows us to fully utilize the images which were not perfectly detected to better facilitate the subsequent learning process. We show that this strategy can make our detector more robust towards hard faces, without involving more complex network architecture and computation overhead for the inference. The main difference between our HIM and OHEM is that ours takes an image as a basic unit for mining while OHEM takes an anchor/proposal as a basic unit for mining. Our proposed HIM is complementary to OHEM and our face detector uses both OHEM and HIM since they aim to solve different issues.

Apart from mining the hard images, we also propose to improve the detection quality and efficiency by exclusively exploiting small faces. Small faces are typically hard and have attracted extensive research attention [81], [85], [86]. Existing methods aim at building a scale-invariant face detector to learn and infer on both small and big faces, with multiple levels of detection features and anchors of different sizes. Compared with these methods, our detector is more efficient since it is specially designed to aggressively leverage the small faces during training. More specifically, large faces are automatically ignored during training due to the design of putting only small anchors on a single high-resolution feature map, so that the model can fully focus on the small hard faces. Experimental results also demonstrate the effectiveness of our design.

To conclude, in this chapter, we propose a novel face detector with the following contributions:

- We propose a hard image mining strategy, to improve the robustness of our detector to those extremely hard faces. This is done without any extra modules, parameters or computation overhead added on the existing detectors.

- We design a single shot detector with only one detection feature map, which focuses on small faces with a specific range of sizes. This allows our model to be simple and focus on difficult small faces without struggling with scale variance.

- Our face detector achieves state-of-the-art level performance on all popular face detection benchmarks, including WIDER FACE, FDDB, Pascal Faces, and AFW. We achieve 95.7, 94.9 and 89.7 on `easy`, `medium` and `hard` WIDER FACE `val` dataset. Our method also achieves APs of 99.00 and 99.60 on Pascal Faces and AFW respectively, as well as a TPR of 98.7 on FDDB.

5.2 Related Work

5.2.1 Hard Example Mining

Hard example mining is an important strategy to improve model quality, and has been studied extensively in image classification [88] and general object detection [87], [89]. The main idea is to find some hard positive and hard negative examples at each training step, and put more effort into training on those hard examples [90], [91]. Recently, with modern detection frameworks proposed to boost the performance, OHEM [87] and Focal Loss [89] have been proposed to select hard examples. OHEM computes the gradients of the networks by selecting the proposals with highest losses in every mini-batch; while Focal Loss aims at naturally putting more focus on hard and misclassified examples by adding a factor to the standard cross entropy criterion. However, these algorithms mainly focus on anchor-level or proposal-level mining, and they will exhibit less effectiveness on easy images where there are no hard anchors/proposals. In face detection, most training images are easy as shown in Figure 5.1, while some extremely hard images drag the final performance down. To this end, we propose to exploit hard example mining on image level, *i.e.* hard image mining, to improve the quality of face detector on extremely hard faces. Our hard image mining can be used in conjunction with OHEM, to put more training efforts on hard images and make OHEM more effective on the selected hard images.

5.2.2 Face Detection Architecture

Recent state-of-the-art face detectors are generally built based on Faster-RCNN [26], R-FCN [28] or SSD [17]. SSH [32] exploits the RPN (Region Proposal Network) from Faster-RCNN to detect faces, by building three detection feature maps and designing six anchors with different sizes attached to the detection feature maps. S^{3}FD [84] and PyramidBox [82], on the other hand, adopt SSD as their detection architecture with six different detection feature maps. Different from S^{3}FD, PyramidBox exploits a feature pyramid-style structure to combine features from different detection feature maps. Our proposed method, on the other hand, only builds single level detection feature map, based on VGG16, for classification and bounding-box regression, which is both simple and effective.

5.2.3 Anchor Design and Matching

Usually, anchors are designed to have different sizes to detect objects with different scales, in order to build a scale-invariant detector. SSD as well as its follow-up detectors S^{3}FD and PyramidBox, have six sets of anchors with different sizes, ranging from (16×16) to (512×512), and their network architectures have six levels of detection feature maps, with resolutions ranging from $\frac{1}{4}$ to $\frac{1}{128}$, respectively. Similarly, SSH has the same anchor setting, and those anchors are attached to three levels of detection feature maps with resolutions ranging from $\frac{1}{8}$ to $\frac{1}{32}$. The difference between SSH and S^{3}DF is that in SSH, anchors with two neighboring sizes share the same detection feature map, while in S^{3}DF, anchors with different sizes are attached to different detection feature maps.

SNIP [92] discussed an alternative approach to handle scales. It showed that CNNs are not robust to changes in scale, so training and testing on a limit range of scales of an image pyramid can be a more optimal strategy. In this chapter, we exploit a similar idea by limiting the anchor sizes to be (16×16), (32×32) and (64×64), which correspond to only small faces. Then those faces with either too small or too big sizes will not be matched to any of the anchors, thus will be ignored during the training and testing. By removing those large anchors with sizes larger than (64×64), our network focuses more on small faces which are potentially more difficult. To process large faces, we use multi-scale training and testing to resize them to match our small anchors. Experimental results show this design performs well on both small and big

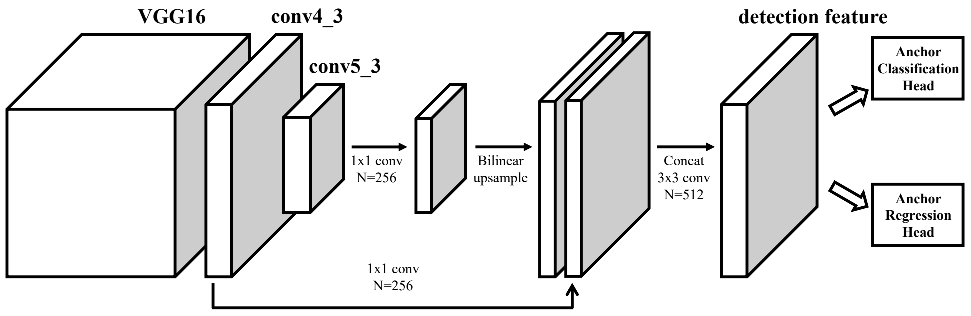

Figure 5.3. The framework of our face detector. We take VGG16 as our backbone CNN, and we fuse two layers (*conv4_3* and *conv5_3*) after dimension reduction and bilinear upsampling, to generate the final detection feature map. Based on that, we add a detection head for classification and bounding-box regression.

faces, although it has fewer detection feature maps and anchor sizes.

5.3 Proposed Method

In this section, we introduce our proposed method for effective face detection. We first discuss the architecture of our detector in Subsection 5.3.1, then we elaborate our hard image mining strategy in Subsection 5.3.2, as well as some other useful training techniques in Subsection 5.3.3.

5.3.1 Single-level Small Face Detection Framework

The framework of our face detector is illustrated in Figure 5.3. We use VGG16 network as our backbone CNN, and combine *conv4_3* and *conv5_3* features, to build the detection feature map with both low-level and high-level semantic information. Similar to SSH [32], we apply 1×1 convolution layers on the *conv4_3* and *conv5_3* feature maps to reduce dimension, and then apply a 3×3 convolution layer on the concatenation of these two dimension reduced feature maps. The output feature of the 3×3 convolution layer is the final detection feature map, which will be fed into the detection head for classification and bounding-box regression.

The detection feature map has a resolution of $\frac{1}{8}$ of the original image (of size

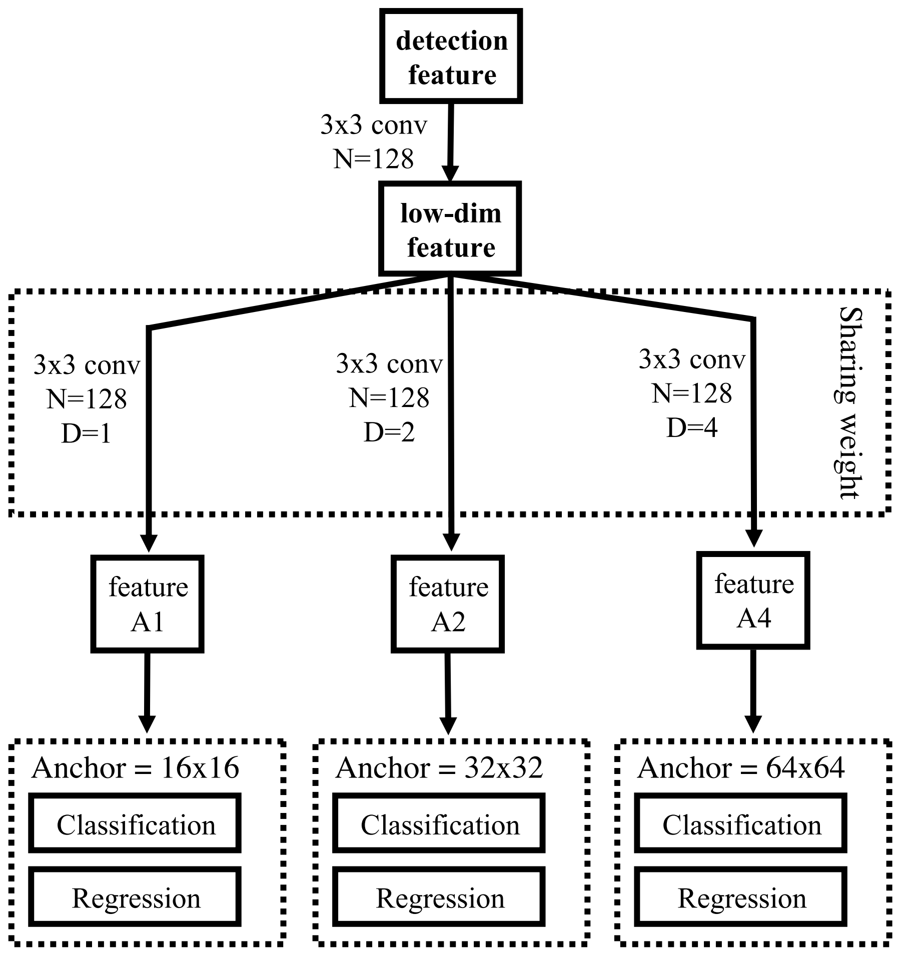

Figure 5.4. The framework of our dilated detection head for classification and regression. Based on the detection feature map from the backbone CNN, we first perform dimension reduction to reduce the number of channels from 512 to 128. Then we put three convolution layers with the shared weight and different dilation rates, to generate final detection and classification features.

$H \times W$). We attach three anchors at each point in the feature map grid as the default face detection boxes. Then we do classification and bounding-box regression on those $3 \times \frac{H}{8} \times \frac{W}{8}$ anchors. Unlike many other face detectors which build multiple feature maps to detect faces with a variant range of scales, inspired by SNIP [92], faces are trained and inferred with roughly the same scales. We only have one detection feature map, with three sets of anchors attached to it. The anchors have sizes of (16×16), (32×32) and (64×64), and the aspect ratio is set to be 1. By making this configuration, our network only trains and infers on faces with small and medium sizes; and we propose to handle large faces by shrinking the images in both training and testing phases. We argue that there is no speed or accuracy degradation for large faces, since inferring on the shrunken images (with the short side containing 100 or 300 pixels) is very fast, and the shrunken large faces will still have enough information to be recognized.

To compensate for the difference of anchor sizes attached to the same detection feature map, we propose a detection head which uses different dilation rates for anchors with different sizes, as shown in Figure 5.4. The intuition is that in order to detect faces with different sizes, different effective receptive field sizes are required. This naturally requires the backbone feature map to be invariant to scales. To this end, we adopt different dilation rates for anchors with different sizes. For anchors with size (16×16), (32×32) and (64×64), we use convolution layers with a kernel size of 3 and dilation rates of 1, 2 and 4 respectively to gather input features at different scales. These three convolution layers share the weights to reduce the model complexity. With this design, the input of the 3×3 convolution, will be aligned to the same location of faces, regardless of the size of faces and anchors. Subsection 5.5.1 will show the effectiveness of this multi-dilation design.

5.3.2 Hard Image Mining

Different from OHEM, which selects proposals or anchors with the highest losses within a single image, we propose a novel hard image mining strategy at the image level within a dataset. The intuition is that most images in the dataset are very easy, and we can achieve a very high accuracy even on the `hard` subset of the WIDER FACE `val` dataset with our baseline model, while there are still some extremely challenging images with occlusion, illumination, makeup and pose/scale variance. We believe not all training images should be treated equally, and well-recognized images will not help towards training a more robust face detector. To put more attention on training hard images instead of easy ones, we use a subset $\mathcal{D}'$ of all training images $\mathcal{D}$, to contain hard ones for training. At the beginning of each training epoch, we build $\mathcal{D}'$ based on the difficulty scores obtained in previous epochs.

We initially use all training images to train our model (*i.e.* $\mathcal{D}' = \mathcal{D}$) for the first epoch. This is very straightforward since our initial ImageNet pre-trained model will only give random guess towards face detection. In this case, there is no easy image and, every image is considered as hard image and fed to the network for training at the first epoch. During the training procedure, we dynamically assign different difficulty scores to each of the training images, which is defined by the metric Worst Positive Anchor Score (WPAS):

$$\mathrm{WPAS}(I;\Theta) = \min_{a \in \mathcal{A}(I)^+} \frac{\exp(l(I;\Theta)_{a,1})}{\exp(l(I;\Theta)_{a,1}) + \exp(l(I;\Theta)_{a,0})}$$

where $\mathcal{A}(I)^+$ is the set of positive anchors for image I, with an IoU over 0.5 against at least one of the ground-truth boxes, l is the classification logit before the softmax layer and $l(I;\Theta)_{a,1}$, $l(I;\Theta)_{a,0}$ are the logits of anchor a for the image I to be foreground face and background respectively. All images are initially marked as hard, and any image with WPAS greater than a threshold th will be regarded as easy, since all positive anchors have been correctly recognized.

At the beginning of each epoch, we first randomly shuffle the training dataset to generate the complete training list $\mathcal{D} = [I_{i_1}, I_{i_2}, \cdots, I_{i_n}]$ for the following epoch of training, where $i_1, \cdots, i_n$ is a random permutation of $1, \cdots, n$. Then given an image marked as easy, we remove it from $\mathcal{D}$ with a probability of p. The remaining training list $\mathcal{D}' = [I_{i_{j_i}}, I_{i_{j_2}}, \cdots, I_{i_{j_k}}]$ (a sublist of $\mathcal{D}$), which focuses more on hard images, will be used for training at this epoch. Note that for multi-GPU training, each GPU will maintain its own training list $\mathcal{D}'$ independently. In our experiments, we set the probability p to 0.7, and the threshold th to 0.85. We test with different p and th and find out that the model performs consistently well when th falls between 0.3 and 0.85

and p falls between 0.5 and 0.7.

5.3.3 Training Strategy

5.3.3.1 Multi-scale Training and Anchor Matching

Since we only have anchors covering a limited range of face scales, we train our model by varying the sizes of training images. During the training phase, we resize the training images so that the short side of the image contains s pixels, where s is randomly selected from $\{400, 800, 1200\}$. We also set an upper bound of 2000 pixels to the long side of the image considering the GPU memory limitation.

For each anchor, we assign a label $\{+1, 0, -1\}$ based on how well it matches with any ground-truth face bounding box. If an anchor has an IoU (Intersection over Union) over 0.5 against a ground-truth face bounding box, we assign $+1$ to that anchor. On the other hand, if the IoU against any ground-truth face bounding box is lower than 0.3, we assign 0 to that anchor. All other anchors will be given -1 as the label, and thus will be ignored in the classification loss. By doing so, we only train on faces with designated scales. Those faces with no anchor matching will be simply ignored, since we do not assign the anchor with largest IoU to it (thus assign the corresponding anchor label $+1$) as Faster-RCNN does. This anchor matching strategy will ignore the large faces, and our model can put more capacity on learning different face patterns on hard small faces instead of memorizing the change in scales.

For the regression loss, all anchors with IoU greater than 0.3 against ground-truth faces will be taken into account and contribute to the smooth $\ell 1$ loss. We use a smaller threshold (*i.e.* 0.3) because (1) this will allow imperfectly matched anchors to be able to localize the face, which may be useful during the testing and (2) the regression task has less supervision since unlike classification, there are no negative anchors for computing loss and the positive anchors are usually sparse.

5.3.3.2 Anchor-level Hard Example Mining

OHEM has been proven to be useful for object detection and face detection in [17], [32], [87]. During our training, in conjunction with our newly proposed hard image mining, we also use the traditional hard anchor mining method to focus more on the hard and misclassified anchors. Given a training image with size $H \times W$, there are $3 \times \frac{H}{8} \times \frac{W}{8}$ anchors at the detection head, and we only select 256 of them to be involved in computing the classification loss. For all positive anchors with IoU greater than 0.5 against ground-truth boxes, we select the top 64 of them with lowest confidences to be recognized as faces. After selecting positive anchors, $(256 - \#pos\_anchor)$ negative anchors with highest face confidences are selected to compute the classification loss as hard negative anchors. Note that we only perform OHEM for classification loss, and we keep all anchors with IoU greater than 0.3 for computing regression loss, without selecting a subset based on either classification loss or bounding-box regression loss.

5.3.3.3 Data Augmentation

Data augmentation is extremely useful to make the model robust to light, scale changes and small shifts [17], [82]. In our proposed method, we exploit cropping and photometric distortion as data augmentation. Given a training image after resizing, we crop a patch of it with a probability of 0.5. The patch has a height of H' and a width of W' which are independently drawn from $\mathcal{U}(0.6H, H)$ and $\mathcal{U}(0.6W, W)$, where $\mathcal{U}$ is the uniform distribution and H, W are the height and width of the resized training image. All ground-truth boxes whose centers are located inside the patch are kept. After the random cropping, we apply photometric distortion following SSD by randomly modifying the brightness, contrast, saturation and hue of the cropped images.

5.4 Experiments

To verify the effectiveness of our model and proposed method, we conduct extensive experiments on popular face detection benchmarks, including WIDER FACE [5], FDDB [93], Pascal Faces [74] and AFW [75]. It is worth noting that the training is only performed on the `train` set of WIDER FACE, and we use the same model for the evaluation on all these datasets without further fine-tuning.

5.4.1 Experiment Settings

We train our model on the `train` subset of WIDER FACE, which has 12880 images with 159k faces annotated. We flip all images horizontally, to double the size of the training dataset to 25760. For each training image, we first randomly resize it, and then we use the cropping and photometric distortion data augmentation methods discussed in the previous section to pre-process the resized image. We use an ImageNet pre-trained VGG16 [10] model to initialize our network backbone, and our newly introduced layers are randomly initialized with Gaussian initialization. We train the model with the iter-size to be 2, for 46k iterations, with a learning rate of 0.004, and then for another 14k iterations with a smaller learning rate of 0.0004. For the training, we use 4 GPUs to simultaneously to compute the gradient and update the weight by synchronized SGD with Momentum [94]. The first two blocks of the VGG16 backbone are frozen during the training, and the rest layers of VGG16 are set to have a doubled learning rate.

Since our model is designed for and trained on only small faces, we build a multi-scale image pyramid for testing in order to deal with faces larger than our anchors. Specifically, we resize the testing image so that the short side contains 100, 300, 600, 1000 and 1400 pixels for the evaluation on WIDER FACE dataset. We also follow some testing-time strategies such as horizontal flip and bounding-box voting [64] as

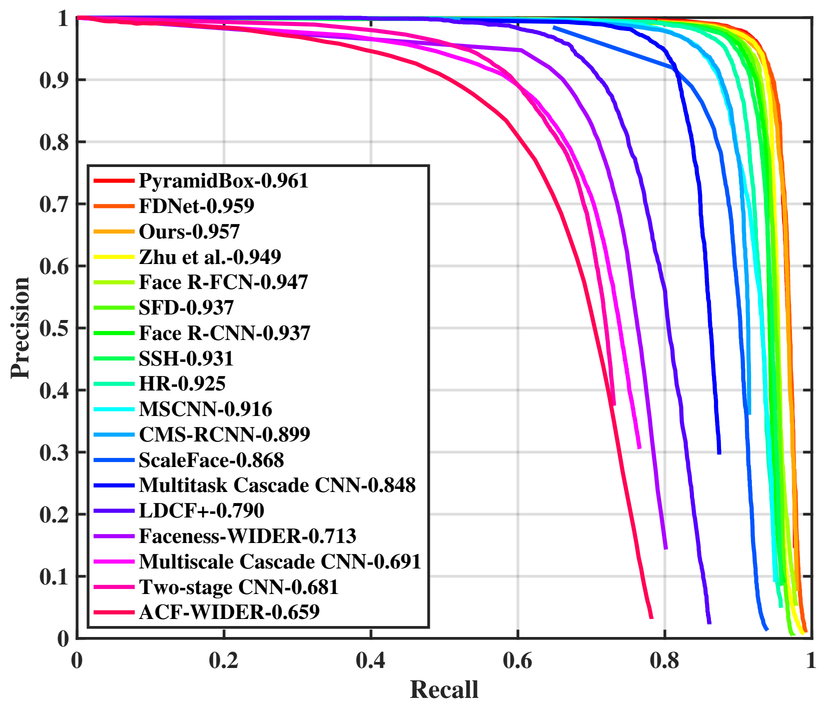

Figure 5.5. Precision-recall curve on the `easy` subset of WIDER FACE `val` dataset.

used in PyramidBox [2] [82].

5.4.2 Experiment Results

5.4.2.1 WIDER FACE

WIDER FACE dataset includes 3226 images and 39708 faces labelled in the `val` dataset, with three subsets – `easy`, `medium` and `hard`. In Figure 5.5, 5.6 and 5.7, we show the precision-recall (PR) curve and average precision (AP) for our model compared with many other state-of-the-arts [5], [6], [32], [82], [84]–[86], [95]–[105] on these three subsets. As we can see, our method achieves the state-of-the-art level accuracy on the `hard` subset. Since the `hard` subset is a **superset** of the `small` and `medium` subsets, and contains all faces taller than 10 pixels, the performance on `hard` set can represent the performance on the full testing dataset more accurately. Our performance on the `medium` subset is comparable to the most recent state-of-the-art

[2]https://github.com/PaddlePaddle/models/blob/release/1.8/PaddleCV/face_detection/widerface_eval.py

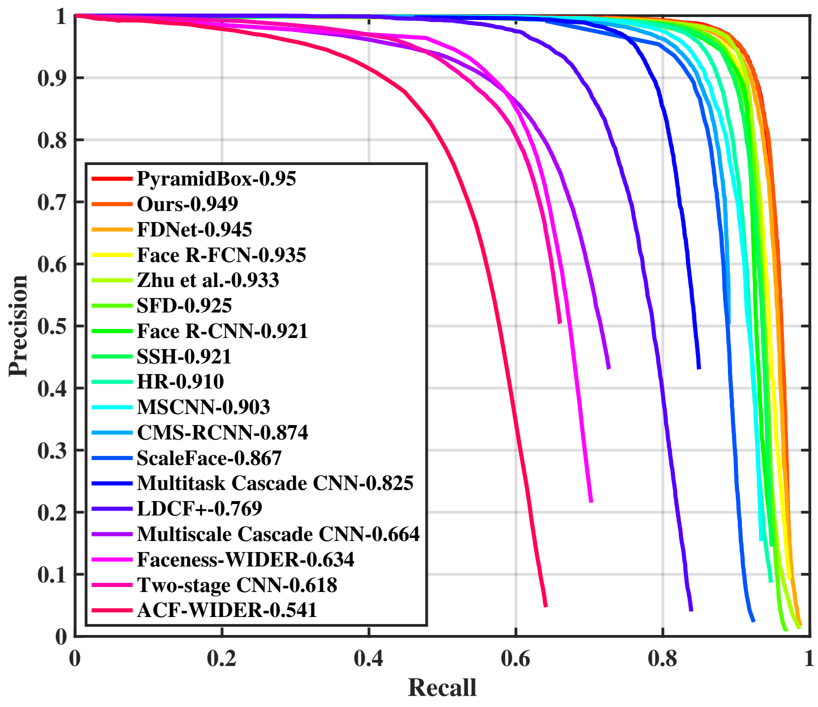

Figure 5.6. Precision-recall curve on the `medium` subset of WIDER FACE `val` dataset.

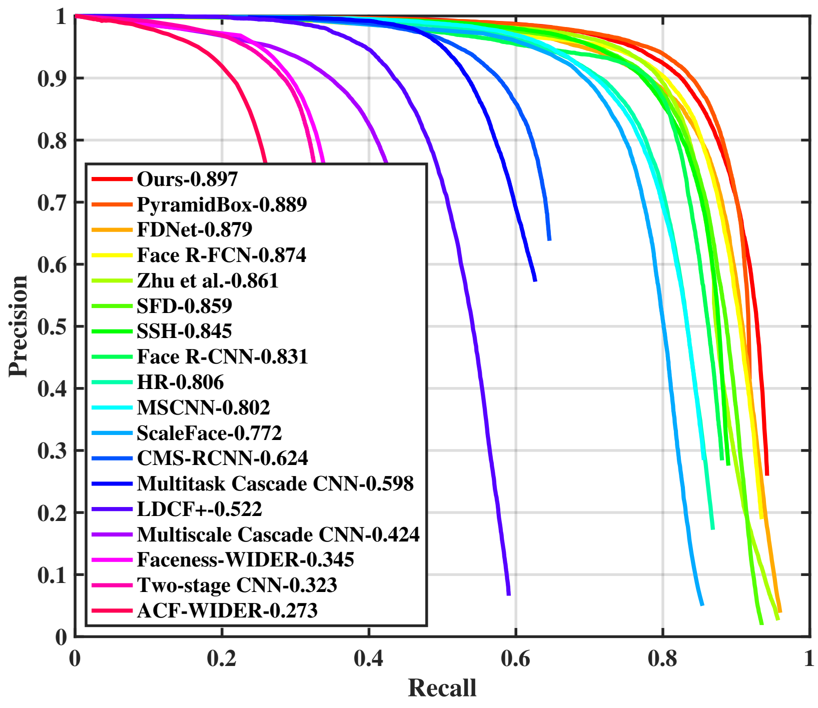

Figure 5.7. Precision-recall curve on the `hard` subset of WIDER FACE `val` dataset.

and the performance on the `easy` subset is a bit worse since our method focuses on learning hard faces, and the architecture of our model is simpler compared with other

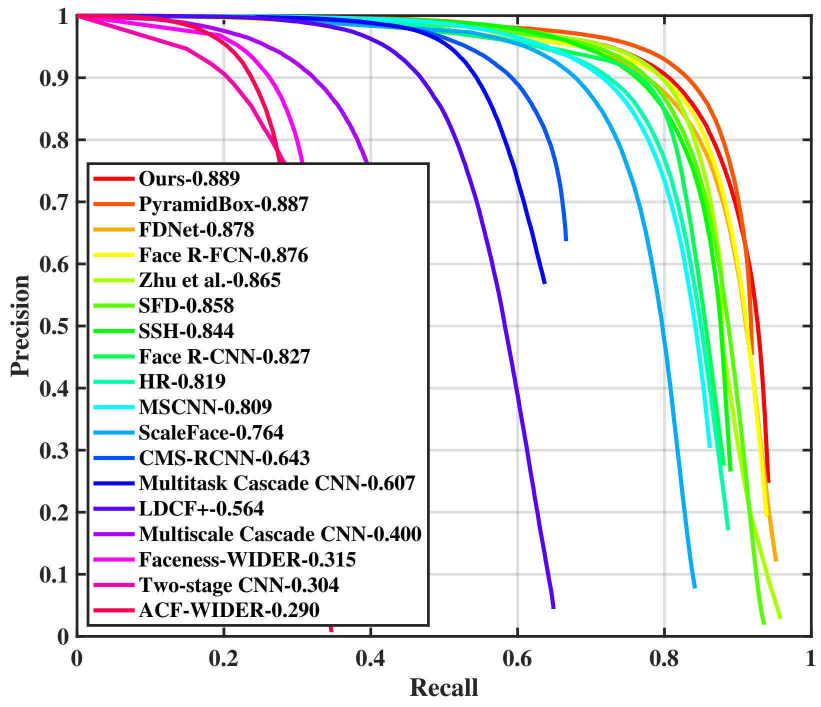

Figure 5.8. Precision-recall curve on the `hard` subset of WIDER FACE `test` dataset.

state-of-the-arts.

In addition to the `val` dataset, there is also a WIDER FACE `test` dataset with no annotations provided publicly. It contains 16097 images, and is evaluated by the official WIDER FACE author team. We report the performance of our method at Figure 5.8 for the `hard` subset.

5.4.2.2 FDDB

FDDB dataset includes 5171 faces annotated on a set of 2845 images, and we use our model trained on WIDER FACE `train` subset to conduct the inference and evaluation for the FDDB dataset. We use the raw rectangular bounding-box result without fitting it into ellipse to compute ROC. We show the discontinuous ROC curve at Figure 5.9 compared with other state-of-the-arts [74], [75], [82], [84], [103], [106]–[109], and our method achieves a very strong performance with a TPR of 98.7% under the constraint of having 1000 false positive predictions.

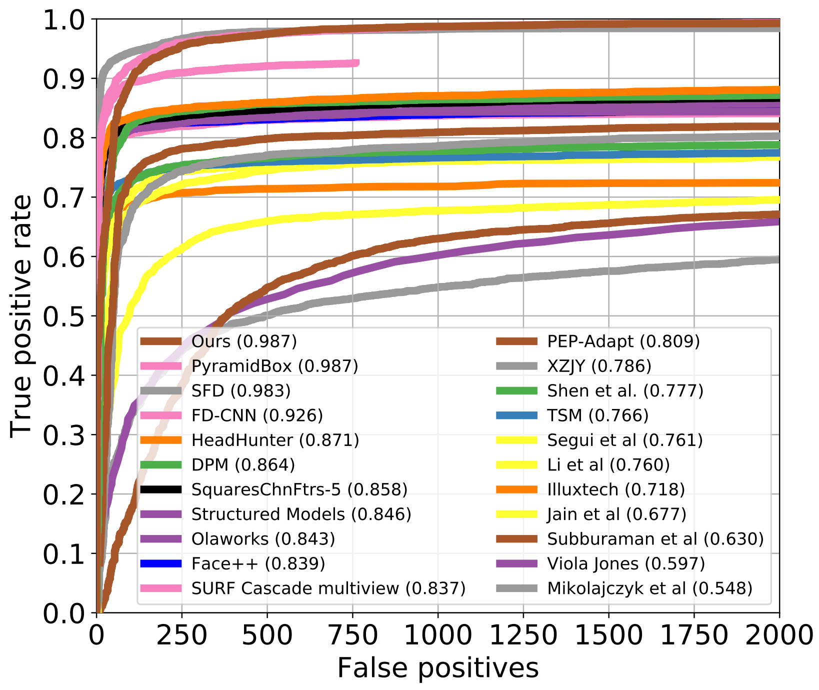

Figure 5.9. Receiver operating characteristic curve on the FDDB dataset.

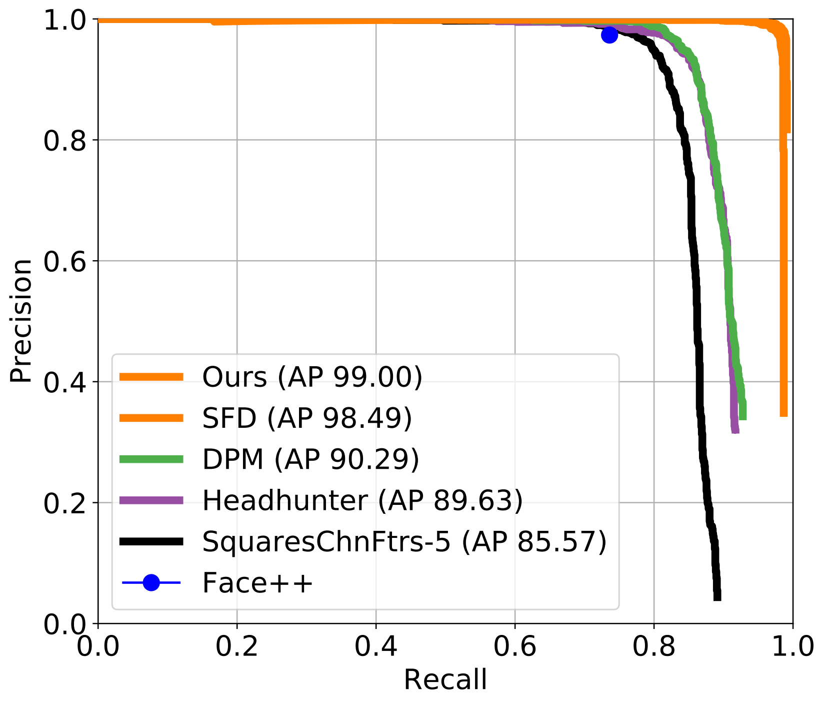

Figure 5.10. Precision-recall curve on the Pascal Faces dataset.

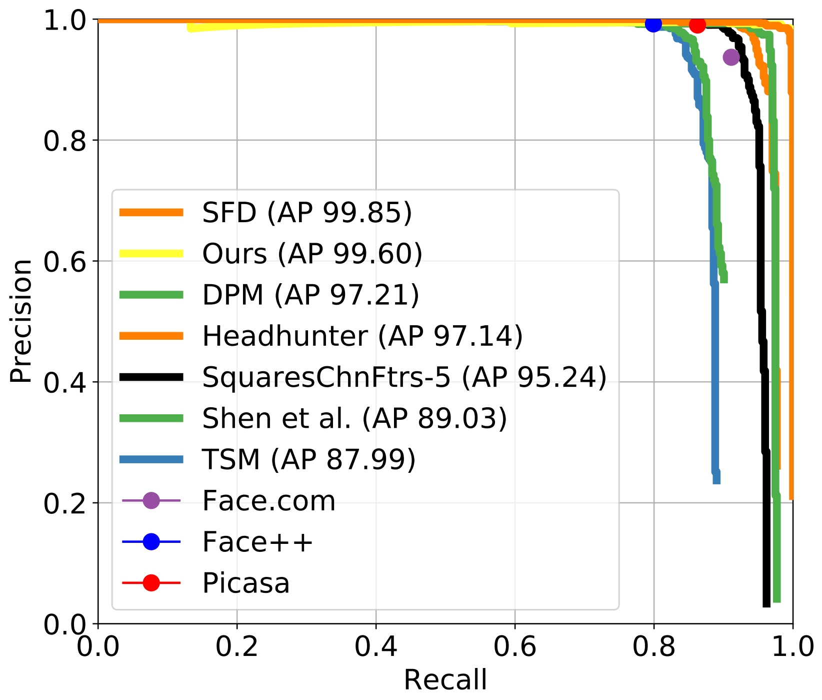

Figure 5.11. Precision-recall curve on the AFW dataset.

5.4.2.3 Pascal Faces

Pascal Faces dataset includes 1335 labeled faces on a set of 851 images extracted from the Pascal VOC dataset. We show the PR curve at Figure 5.10 compared with [84], [107], and our method achieves a new state-of-the-art performance of 99.0.

5.4.2.4 AFW

AFW dataset includes 473 faces labelled in a set of 205 images. As shown in Figure 5.11 compared with [75], [84], [107], [108], our method achieves state-of-the-art and almost perfect performance, with an AP of 99.60.

Method	`easy`	`medium`	`hard`
Baseline-Three	95.0	93.8	88.5
+ HIM	95.5	94.5	89.0
Baseline-Single	95.1	94.2	89.1
+ HIM	95.4	94.8	89.6
+ DH	95.4	94.5	89.3
+ DH + HIM	**95.7**	**94.9**	**89.7**

Table 5.1. Ablation experiments. Baseline-Three is a face detector similar to SSH with three detection feature maps trained with the same hyperparameters. Baseline-Single is our proposed detector with single detection feature map shown in Figure 5.3. HIM and DH represents hard image mining and dilated head architecture (Figure 5.4). The same testing strategy is used for all entries.

5.5 Ablation Study and Diagnosis

5.5.1 Ablation Experiments

In order to verify the performance of our single level face detector, as well as the effectiveness of our proposed hard image mining, the dilated-head architecture for anchor box classification and regression, we conduct various ablation experiments on the WIDER FACE `val` dataset. All results are summarized in Table 5.1. From Table 5.1, we can see that our single level baseline model can already achieve performance comparable to the current state-of-the-art face detector, especially on the hard subset. Our model with single detection feature map performs better than the one with three detection feature maps, despite its shallower structure, fewer parameters and anchors. This confirms the effectiveness of our simple face detector with single detection feature map focusing on small faces.

We also separately verify our newly proposed hard image mining (HIM) and dilated head architecture (DH). HIM can improve the performance on the hard subset significantly without involving more complex network architecture nor computation overhead, and this benefit holds for both our three-level baseline and single-level baseline. DH itself can also boost the performance, which shows the effectiveness

th	0.0	0.3	0.5	0.85	0.95	1.0
AP@hard	89.0	89.4	89.4	89.6	89.1	89.1

p	0.0	0.3	0.5	0.7	0.9	1.0
AP@hard	89.1	89.2	89.4	89.6	89.2	87.6

Table 5.2. Hyper-parameter ablation study. All experiments are conducted for the Baseline-Single+HIM version described in 5.5.1. When diagnosing the effect of th and p, the other hyperparameter p and th are fixed to the best values (*i.e.*, 0.7 and 0.85) respectively.

of designing larger receptive field sizes for larger anchors. Combining HIM and DH together can achieve the best performance, as shown in the last row.

5.5.2 Diagnosis of Hyperparameters

In this subsection we diagnose how the hyperparameters affect the model quality. Our method is not sensitive to the hyperparameters in a certain range, as shown in the ablation study (Table 5.2). When $0.3 \leq th \leq 0.85$, our method yields a large performance gain consistently, except for when th is too high or too low, since th with extreme values (*i.e.*, close to 0 or 1) makes all images are considered as easy or hard equally regardless of their actual difficulties. For the hyperparameter p, a similar tendency remains. Only when $p = 1.0$, the performance goes down drastically since those easy images will no longer be trained and the training images are limited to the hard ones, otherwise our method produces consistent performance gain.

5.5.3 Diagnosis of OHEM as Hard Face Mining

In this subsection we investigate the possibility of modifying online hard example mining (OHEM) [87], by deliberately ignoring easy faces in each training image and keeping only hard positive proposals/anchors in the training, to make the network focus more on hard faces. Note that in our original implementation of OHEM as described in Subsection 5.3.3, we select the top 64 hardest positive anchors for each training image, which cover all ground-truth faces regardless of their difficulties in

Method	`easy`	`medium`	`hard`
Baseline-Single	95.1	94.2	89.1
+ HIM	95.4	94.8	89.6
+ OHEM_HARD_FACE	95.0	94.2	89.1

Table 5.3. Diagnose of using OHEM to mine hard faces. All entries are based on the Baseline-Single+HIM version described in 5.5.1.

PD	Crop	`easy`	`medium`	`hard`
No	No	94.5	93.9	88.4
Yes	No	94.5	94.1	88.8
Yes	Yes	95.1	94.2	89.1

Table 5.4. Diagnosis of data augmentation. PD indicates photometric distortion. All entries are based on our Baseline-SingleLevel configuration without HIM and DH.

most cases. In this ablation, we modify the OHEM to mimic the settings used for HIM (see 5.3.2), and disable a positive anchor with a probability of p if the score of that positive anchor is greater than th, where p is 0.7 and th is 0.85. All other settings are kept the same and we do not use the dilated head architecture (DH) here. The results are summarized in Table 5.3, and the performance of using OHEM to mine hard faces (listed as OHEM_HARD_FACE in the table) is inferior compared with our HIM. It confirms our intuition that most images are easy without hard faces, and mining hard faces on those easy images is not helpful to train a more robust detector, so it is important to select and focus on hard images.

5.5.4 Diagnosis of Data Augmentation

We investigate the effectiveness of the photometric distortion as well as the cropping mechanisms as discussed in Subsection 5.3.3.3. The ablation results evaluated on WIDER FACE `val` dataset are shown in Table 5.4. Both photometric distortion and cropping can contribute to a more robust face detector.

Testing Scales (x100)	`easy`	`medium`	`hard`
[6, 10, 14]	78.2	85.7	86.1
[3, 6, 10, 14]	91.3	92.6	88.8
[1, 3, 6, 10, 14]	95.7	94.9	89.7

Table 5.5. Diagnosis of multi-scale testing. All results are evaluated with the same model with HIM and DH.

5.5.5 Diagnosis of Multi-scale Testing

Our face detector with one detection feature map is designed for small face detection, and our anchors are only capable of capturing faces with sizes ranging from (16×16) to (64×64). As a result, it is critical to adopt multi-scale testing to deal with large faces. Different from SSH, S^3FD and PyramidBox, our testing image pyramid includes some extremely small scales (*i.e.* short side contains only 100 or 300 pixels). In Table 5.5, we show the effectiveness of these extremely small scales to deal with easy and large images. Our full-size version evaluation resizes the image so that the short side contains 100, 300, 600, 1000 and 1400 pixels respectively, to build an image pyramid. We diagnose the impact of the extra small scales (*i.e.* 100 and 300) by removing them from the image pyramid.

As shown in Table 5.5, the extra small scales are crucial to detect easy faces. Without resizing the short side to contain 100 and 300 pixels, the performance on `easy` subset is only 78.2, which is even lower than the performance on `medium` and `hard` which contain much harder faces. We will show in the next subsection that these extra small scales (100 and 300) lead to negligible computation overhead, due to the lower resolution.

5.5.6 Analysis of Accuracy/Speed Trade-off

We evaluate the speed of our method as well as some other popular face detectors in Table 5.6. For fair comparison, we run all methods on the same machine, with a

Method	MS	HF	Time	G-Mem	AP-h
SSH	Yes	No	1.00	6.1	84.5
S^{3}FD	Yes	Yes	1.34	6.2	85.2
PyramidBox	Yes	Yes	2.24	11.9	88.9
Ours*	Yes*	Yes	1.59	**5.3**	86.1
Ours	Yes	No	**0.84**	**5.3**	89.3
Ours	Yes	Yes	1.70	**5.3**	**89.7**

Table 5.6. Diagnosis of inference speed. MS and HF indicate multi-scale testing and horizontal flip; Time is the inference time (in second) for a single image; G-Mem is the GPU memory usage in gigabyte; AP-h is the average precision on the `hard` subset of WIDER FACE `val` set. Ours* indicates our detector without extra small scales. All entries are evaluated with a single nVIDIA Titan X Maxwell.

single Titan X Maxwell GPU, and Intel Core i7-4770K 3.50GHz. All methods except for PyramidBox are based on Caffe1 implementation, which is compiled with CUDA 9.0 and CUDNN 7. For PyramidBox, we follow the official fluid code and the default configurations[3]. We use the official PaddlePaddle with CUDA 9.0 and CUDNN 7[4].

For SSH, S^{3}FD and Pyramid, we use the official inference code and configurations. For SSH, we use multi-scale testing with the short side containing 500, 800, 1200 and 1600 pixels, and for S^{3}FD, we execute the official evaluation code with both multi-scale testing and horizontal flip. PyramidBox takes a similar testing configuration as S^{3}FD. As shown in Table 5.6, our detector can outperform SSH, S^{3}FD and PyramidBox significantly with a smaller inference time. Based on that, using horizontal flip can further improve the performance slightly. In terms of GPU memory usage, our method uses only a half of what PyramidBox occupies, while achieving better performance.

Ours* in Table 5.6 indicates our method without extra small scales in inference, *i.e.*, evaluated with scales [600, 1000, 1400]. It is only 6.5% faster than evaluation with [100, 300, 600, 1000, 1400] (1.59 compared with 1.70). This confirms that although our face detector is only trained on small faces, it can perform well on large faces, by

[3]https://github.com/PaddlePaddle/models/blob/release/1.8/PaddleCV/face_detection/widerface_eval.py

[4]`pip install paddlepaddle-gpu`

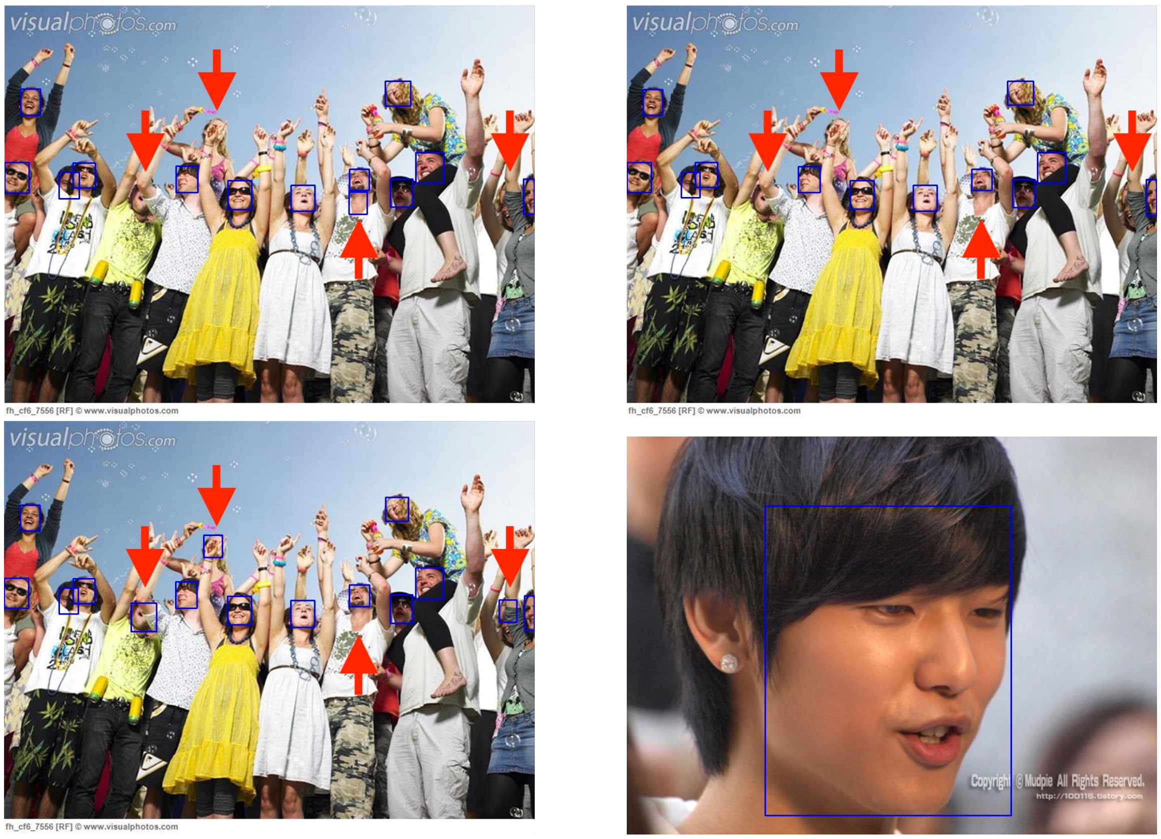

Figure 5.12. Qualitative example. Left-upper: baseline. Right-upper: +DH. Left-lower: +DH+HIM. The first three subfigures are the results on the same image, and the differences are highlighted by red arrows. Right-lower: Large face detected by our final model. Detected boxes with scores greater than 0.5 are plotted.

simply shrinking the testing image with negligible computation overhead.

5.5.7 Qualitative Examples

We show qualitative examples in Fig 5.12. It is observed from the first three subfigures (with differences highlighted by red arrows) that both DH and HIM can improve detection quality especially for hard faces. In the lower-right subfigure, we also show an example of detecting large face. As we explained, the image will be shrunk and then the faces will be successfully detected.

5.6 Conclusion and Future Works

To conclude, we propose a novel face detector to focus on learning small faces on hard images, which achieves superior performance on all popular face detection benchmarks. We identify that there are many easy training images with no hard faces which are less useful to build a robust detector, and traditional online hard example mining (OHEM) is unable to handle this imbalance between images. Based on this, we propose a hard image mining strategy by dynamically assigning difficulty scores to training images, and re-sampling subsets with more hard images and fewer easy images for training at each epoch. We also design a single-stage face detector with only one detection feature map, to train and test on small faces. With these designs, our model can put more attention on learning small hard faces instead of memorizing change of scales. Extensive experiments and ablations have been carried out to show the effectiveness of our method, and our face detector achieves the state-of-the-art performance on all popular face detection benchmarks, including WIDER FACE, FDDB, Pascal Faces and AFW. Our face detector also enjoys a fast multi-scale inference speed and less GPU memory usage. Our proposed method is flexible and can be applied to other backbones and tasks, which we remain as future work.

The proposed method improves the training strategy for more focuses on the hard images, but it does not improve its capability of dealing with hard images, since the model architecture remains unchanged. *I.e.*, for the extremely hard faces, the model may be inherently incapable of learning to detect them. So in addition to putting more attentions on hard faces, we also need to improve the network architecture designs for learning them. Possible extension directions could be recovering/reconstructing clearer/larger faces for the hard ones our algorithm identified [110], and redesigning the detection anchors for the hard faces [85].

Chapter 6

Efficiently Bridging 3D Context For Lesion Detection in CT Images

Except for face detection, another important application of object detection lies in computer-aided medical diagnosis, which can help to reduce the workload of doctors and conduct medical examinations for more people. In this chapter, we put our focus on this direction, and investigate some unique properties and challenges for it. To be specific, we investigate how to efficiently and effectively incorporating 3D context information for object detections in medical images, of which 3D information are prevalent. Lesion detection in CT (computed tomography) scan images is an important yet challenging task due to the low contrast of soft tissues and similar appearance between lesion and the background. Exploiting 3D context information from neighboring slices has been studied extensively to improve detection accuracy. However, previous methods either use a pure 3D CNN which is less computationally efficient and cannot take use of pre-trained 2D weights; or simply concatenate feature maps of independent 2D CNNs to obtain 3D context information, which is less effective to capture 3D knowledge. To address these issues, we design a hybrid detector to combine benefits from both of the above methods. We propose to build several light-weighted 3D CNNs as subnets to bridge 2D CNNs' intermediate features, so that 2D CNNs are connected with each other and interchange 3D context information while running the feed-forwarding pass. Comprehensive experiments in the publicly available DeepLesion dataset show that our method can combine 3D knowledge effectively and provide higher quality backbone features. Our detector surpasses the current state-of-the-art by a large margin with comparable speed and GPU memory consumption.

6.1 Motivation and Overview

Lesion detection is an essential task for clinical applications such as computer-aided diagnosis. With the emergence of modern CNNs, object detection in 2D natural images has been developed quickly and achieves promising performance [17], [21], [26], [28]. However, it is still unclear how to adapt these algorithms into CT scans effectively. The main gap is how to efficiently involve 3D context information into these detectors. This problem has attracted many research attentions [36], [111], [112], due to its importance for the success of lesion detection.

Current solutions come in two folds. One uses fully 3D-connected CNNs, which can directly exploit 3D knowledge, for classification and regression. However, this method conducts 3D convolution throughout the network and is less computational efficient. It is also unable to make use of ImageNet pre-trained weights, thus only achieves inferior lesion detection accuracy as discussed in [36]. To alleviate the issues of 3D CNNs, other studies are exploring how to combine 2D CNN features from consecutive CT slices for classification and regression, so as to better utilize the 3D context information. Yan *et al.* [36] follows R-FCN [28] which uses a Region Proposal Network (RPN) to predict suspicious regions (*i.e.* proposals) and a Region Classification Network (RCN) to further classify and regress those suspicious regions. [36] proposes to concatenate backbone feature maps from neighboring CT slices to feed into RCN, in order to gather 3D information in the RCN subnet. Under this pipeline, a backbone network can take the whole CT scans as the input and can be trained in an end-to-end manner, from ImageNet pre-trained weights. However, the backbone networks are still independent 2D CNNs, and no 3D information can be aggregated until the final backbone features are computed. Another possible deficiency is that the central CT slice and the contextual CT slices share symmetric architectures and weights, which may be less optimal since we expect to distill different and complementary knowledge from those different slices.

We propose a hybrid detector combining advantages of fully 3D-connected CNN detectors (strong knowledge of 3D context) [111], [112] and 2D CNN concatenated detectors (efficiency and ability to use ImageNet pre-trained weights) [36]. Similar to [36], we use 2D CNNs for CT slices at different axial locations as our backbone. However, as discussed before, this is less optimal since these isolated 2D CNNs cannot extract and exploit 3D context information. To address this problem, we propose light-weighted 3D CNN subnets named 3D Fusion Modules (3DFMs) to bridge those 2D CNNs, allowing information to flow in-between different slices. These subnets connect the internal layers of 2D CNNs, so that each 2D CNN can distill knowledge from its neighbor 2D CNNs, to exploit 3D information and focus on different knowledge. The main difference between [36] and our method is that in [36], the 3D context information is not exploited in the layers before the RCN, and the RCN cannot fully utilize 3D context since the its input features only have high-level semantics without low-level details, and the RCN has a very shallow structure which is incapable of learning rich 3D information; on the contrary, in our method, the 3D information is exploited gradually throughout our backbone CNNs, and 3DFMs learn 3D information at low-level, mid-level and high-level layers. Our design breaks the isolation among 2D CNNs, and enables them to distill different knowledge from different input slices, thus the backbone provides stronger features with richer 3D context encoded.

3DFMs introduce only a few parameters and a small computation overhead, while greatly improve the detection accuracy. Experiments on DeepLesion [15] show our

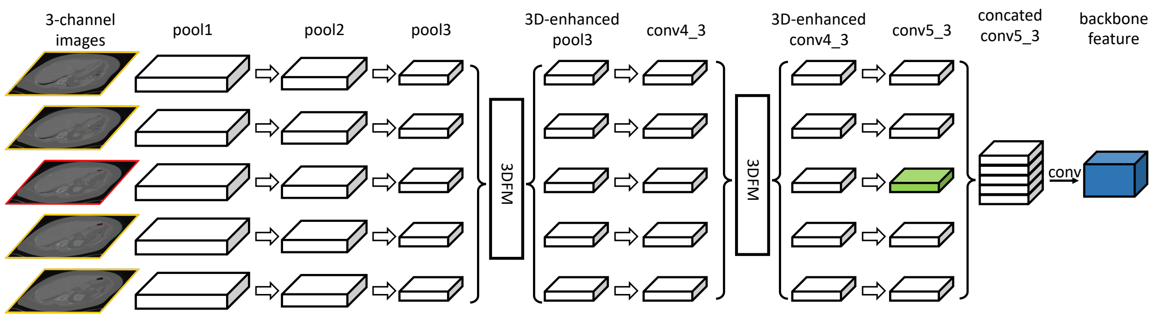

Figure 6.1. Backbone of our hybrid lesion detector. Different rows illustrate different 2D CNNs for the corresponding images. The ground-truth boxes are labelled in the central image (with red boundary) and other 3-channel images (with yellow boundary) are served as 3D context. The central *conv5_3* feature (marked in green) is used in RPN and the fused feature (marked in blue) is used in RCN. Best view in color.

hybrid detector significantly improves the sensitivities at every false positive (FP) rate and on every lesion type. With 27 CT scan slices as input, hybrid detector improves the average sensitivities by 1.4 and the sensitivity at $\frac{1}{8}$ FP per image by 2.7. Our method surpasses [36] and achieves a new state-of-the-art.

6.2 Methodology

6.2.1 Overview Pipeline

The backbone of our detector is shown in Figure 6.1. Following [36], to make use of ImageNet pre-trained weights, we combine 3 adjacent CT slices into a 3-channel input like a natural image, to feed to VGG16 [11], which serves as the backbone 2D CNN of our detector. When considering more 3D context, we combine context slices into 3-channel images and feed them to different VGG16 branches. Each VGG16 branch takes a 3-channel image as input, and generates a *conv5_3* feature map as output. The *conv5_3* feature from the central slice (marked in green) is used in the Region Proposal Network (RPN) to generate proposals, and the concatenation of *conv5_3* features from all slices (marked in blue) is used in the Region Classification Network (RCN) to classify and regress proposals. However, unlike [36], where 2D

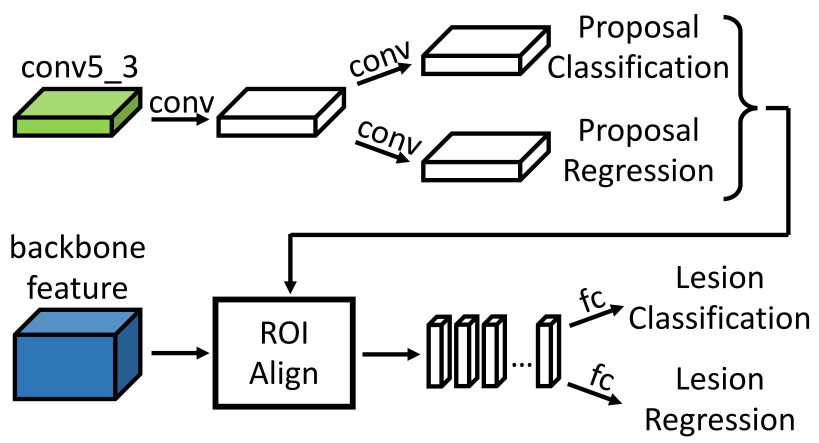

Figure 6.2. RPN (in the top row) and RCN (in the bottom row) sub-networks.

CNNs feed-forward isolatedly, we use a novel and efficient 3D Fusion Module (3DFM) to bridge internal features from different 2D CNNs to build a hybrid backbone. The hybrid detector backbone can better exploit 3D context and make different 2D CNNs to learn different patterns, while utilizing ImageNet pre-trained weights. Details of 3DFM are discussed in Subsection 6.2.2.

Given the backbone of our hybrid lesion detector, we follow [26] to employ an RPN and an RCN to generate and classify proposals. As Figure 6.2 shows, we use the *conv5_3* feature of the central branch (marked in green) to generate proposals, by performing anchor classification and regression on each feature grid, and use ROIAlign [13] to generate features from the concatenated feature of different branches (marked in blue), for each proposal. Finally, those features are used to classify and regress the proposals by the detection head, and generate lesion detection results.

6.2.2 3D Fusion Module

3D context has been shown to be extremely important to detect objects in CT scan images [36], [111], [112]. However, existing methods to utilize 3D information are either computationally expensive and cannot make use of ImageNet pre-trained weights, or inefficient which naively concatenate features from different slices. In this chapter, we

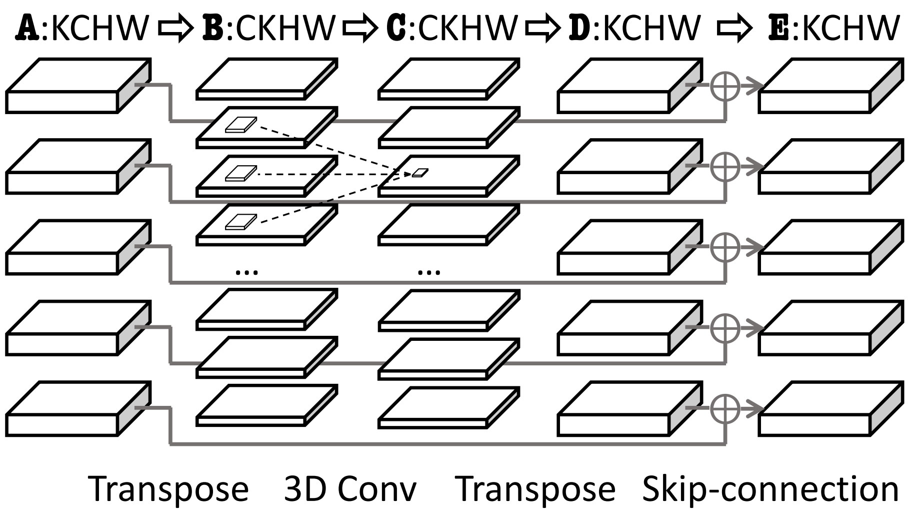

Figure 6.3. 3D Fusion Module. K is 5 in this example.

propose an efficient and computationally cheap 3D Fusion Module (3DFM), as shown in Figure 6.3, to combine 3D context information in the backbone 2D CNNs.

3DFM takes internal features ($A_i \in \mathbb{R}^{C\times H\times W}$, where C, H and W are the channel, height and width of the feature map) from the backbone CNNs as inputs, as shown in the first column in Figure 6.3. Given K 2D backbones, there will be K intermediate features for the input, and each of them is generated from a 3-channel CT image as shown in Figure 6.1. We first concatenate them to build a 4D tensor $\mathbf{A} \in \mathbb{R}^{K\times C\times H\times W}$, and transpose it make the channel to be the first dimension ($\mathbf{B} \in \mathbb{R}^{C\times K\times H\times W}$), as shown in the second column in Figure 6.3. A 3D convolution is used to gather 3D context information to generate a 3D fused feature map $\mathbf{C} \in \mathbb{R}^{C\times K\times H\times W}$. The kernel size is $3\times 1\times 1$ corresponding to the K, H and W dimensions, so we are utilizing the context along the axial direction by convolving across neighbor slices. We use $3\times 1\times 1$ instead of $3\times 3\times 3$ because the context along the other two directions is already considered in the 2D convolutions in the backbone CNN, and thus we only need to consider the axial direction to reduce computation/memory overhead. Finally $\mathbf{C}$ is transposed backed to $K\times C\times H\times W$ as $\mathbf{D}$, and the sum of $\mathbf{A}$ and $\mathbf{D}$ (noted as $\mathbf{E}$) is split to K feature maps with shape $C\times H\times W$, which are used in the backbone

2D CNNs for future processing.

3DFM is flexible and can be inserted anywhere in the backbone CNNs to fuse the 3D information. In our detector, we insert 3DFMs in a sparse manner: only at the *pool3* and *conv4_3* layers in VGG16, as in Figure 6.1. These 3DFMs will combine those independent 2D VGG16 branches into a sparsely bridged 3D CNN, which will serve as the backbone CNN of our detector. Extensive experiments show our design is light-weighted and takes very little computation/memory overhead, while effectively exploiting 3D context knowledge and improving the accuracy significantly.

6.3 Experiments

6.3.1 Implementation Details

Our hybrid detector is implemented with Tensorflow [113]. We use VGG16 as our backbone CNN, and remove the *pool4* layer to keep the output resolution to be $\frac{1}{8}$ of the input image. We take the same CT scan image preprocessing as in [36], which rescales the CT intensity to 0-255, resizes the images and clips the black border. We use the horizontal flip data augmentation which is very common for object detection. For each input example, we take adjacent 3, 9, 15, 21 or 27 CT slices to generate 1, 3, 5, 7 or 9 input images with 3 channels each, to evaluate the efficacy of hybrid detector at different 3D context richness levels, and to make a fair comparison with the current state-of-the-art 3DCE [36]. For the training, we take a batch size of 2, and train the hybrid detector for 120k iterations. The initial learning rate is 10^{-3} and is reduced to 10^{-4} after the first 90k iterations. We take the official `train/test` subset split to train and report accuracy. Comprehensive experiments and ablation studies are reported in the following subsections.

Settings	0.125	0.25	0.5	1	AVG@0.125:8
	2	4	8	16	
Baseline - 3 slices	31.52	43.95	57.19	68.51	64.28
	77.47	83.59	87.77	90.66	
3DCE [36] - 9 slices	-	-	59.32	70.68	-
	79.09	84.34	87.81	89.62	
Baseline - 9 slices	35.48	48.84	62.42	73.06	67.94
	80.73	85.82	89.22	91.21	
Hybrid - 9 slices	38.25	50.66	62.97	73.20	68.65
	80.66	85.80	89.04	91.21	
Baseline - 15 slices	37.53	51.23	63.97	74.53	69.17
	81.39	86.15	89.37	91.28	
Hybrid - 15 slices	40.33	53.01	65.26	75.78	70.49
	82.44	86.84	89.76	**91.69**	
Baseline - 21 slices	38.81	52.32	64.93	75.25	69.93
	82.19	86.61	89.44	91.25	
Hybrid - 21 slices	40.74	53.80	66.06	75.66	70.79
	82.60	86.88	89.79	91.62	
3DCE [36] - 27 slices	-	-	62.48	73.37	-
	80.70	85.65	89.09	91.06	
Baseline - 27 slices	38.43	52.09	65.03	75.10	69.67
	81.88	86.05	89.10	91.05	
Hybrid - 27 slices	**41.12**	**53.83**	**66.32**	**76.27**	**71.04**
	82.89	**87.01**	**89.84**	**91.69**	

Table 6.1. Performance (%) on the `test` split for DeepLesion dataset. $0.125, \cdots, 16$ represent the number of FPs per image.

6.3.2 Experiment Results

To evaluate the efficacy of our method, we conduct extensive experiments on DeepLesion [15]. Following the metric used in LUNA challenge [114], we compute the sensitivity at 7 pre-defined false positive (FP) per image rates: $\frac{1}{8}$, $\frac{1}{4}$, $\frac{1}{2}$, 1, 2, 4 and 8 FPs per input example, as well as the average sensitivity at these 7 pre-defined FP rates. We also compute the sensitivity at the FP per image rate of 16, to compare with the 3DCE [36]. For all our baselines and hybrid detectors, we train and evaluate for four times, and report the average performance, to alleviate the randomness caused by initialization and training data shuffling.

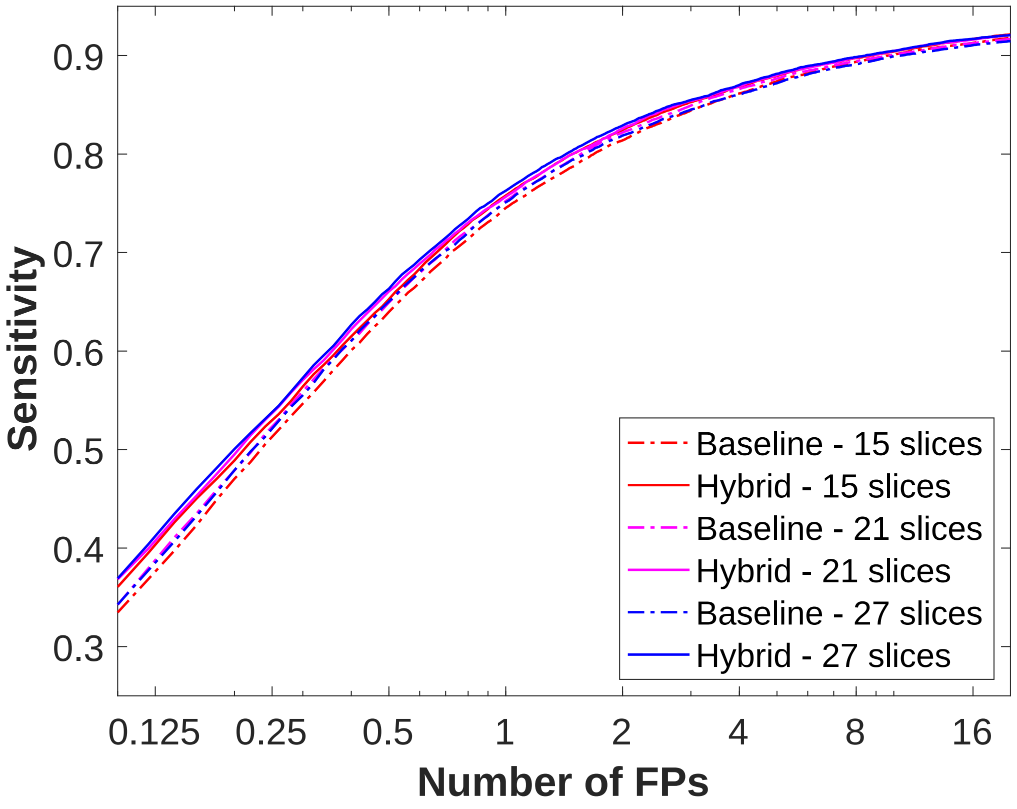

Figure 6.4. FROCs of Baseline and Baseline+3DFM (Hybrid). Best view in color.

The results on the official `test` set are shown in Table 6.1. We compare our method with our Faster-RCNN [26] based baseline, as well as 3DCE which is the current state-of-the-art and already surpasses fully 3D-connected detectors. 'Baseline' in the table is a Faster-RCNN based detector with feature concatenation after the backbone CNN, and 'Hybrid' is 'Baseline' equipped with 3DFMs illustrated in Figure 6.3. We also plot the free-response receiver operating characteristic (FROC) curves for our baseline and hybrid detectors in Figure 6.4. In the table and figure, we find that our hybrid detector with 3DFM is very effective in improving the detection quality. The sensitivity consistently goes up at all FP rate levels significantly with 27 slices as input, especially in the high precision case (*i.e.* fewer FPs per image). For example, our method outperforms the baseline by a relative improvement of 7% (38.43 vs 41.12) when the number of false positives is limited to 0.125 in average per input example

Settings	AVG@0.125:8	Runtime (s)	FPS	GPU memory (GB)
Baseline - 9 slices	67.94	246	19.58	0.455
Hybrid - 9 slices	68.65	256	18.82	0.459
Baseline - 15 slices	69.17	345	13.96	0.693
Hybrid - 15 slices	70.49	369	13.05	0.696
Baseline - 21 slices	69.93	452	10.66	0.930
Hybrid - 21 slices	70.79	479	10.06	0.934
Baseline - 27 slices	69.67	564	8.54	1.167
Hybrid - 27 slices	71.04	608	7.92	1.171

Table 6.2. Performance on the official `test` split for DeepLesion dataset.

with 27 slices. Our hybrid detector surpasses 3DCE greatly with the same `train`/`test` sets and achieves a new state-of-the-art.

6.4 Ablation Studies

6.4.1 Inference Speed and Memory Overhead

Our 3D Fusion Modules (3DFMs) efficiently combine 3D context information in the backbone 2D CNNs. To quantitatively evaluate the computation/memory overhead, we run all our baselines and detectors on a machine with a single nVIDIA Titan Xp GPU. We report the total runtime for the official `test` set (4817 samples) and the max GPU memory consumed for inference. Results are shown in Table 6.2. It can be found that our 3DFMs introduce very small computation overhead and negligible GPU memory overhead. 'Hybrid - 15 slices' can even surpass 'Baseline - 27 slices' with much fewer input slices and faster speed. This verifies the efficiency of our method, which enables our method to be applied to larger and more complex datasets.

6.4.2 Architecture of 3DFM

In this subsection, we compare our 3D Fusion Module with some other potential architectures combining 3D context information:

- 3DFM Without Skip Connection: the 3D context information bridging module

Settings	0.125 2	0.25 4	0.5 8	1 16	AVG@0.125:8
Baseline	38.43 81.88	52.09 86.06	65.03 89.10	75.10 91.05	69.67
Hybrid (Ours)	41.12 82.89	53.83 87.01	66.32 89.84	76.27 91.69	71.04
W/O Skip Connection	42.10 81.79	54.22 86.11	66.29 88.93	75.15 90.78	70.66
W/O 3D Conv	39.96 81.96	53.46 86.72	65.66 89.71	75.36 91.56	70.40
3DFM@4	40.56 82.49	53.37 86.77	65.62 89.56	75.91 91.57	70.62
3DFM@234	40.87 82.90	54.27 87.18	66.45 90.15	76.35 92.00	71.17

Table 6.3. Ablation of 3DFM architecture and the number of 3DFM Instances. See details in Subsection 6.4.2 and 6.4.3.

is the same as 3DFM (see Figure 6.3), but does not have the skip connection to combine the original backbone features with the 3D fused features.

- 3DFM Without 3D Conv: the 3D context information bridging module concatenates the K backbone features with size of $C \times H \times W$ to a thicker tensor $KC \times H \times W$, and takes a 1×1 2D Conv to fuse information from different slices. Note that in this implementation, the 3D information is also learnt, but in a different and fuzzier way.

All experiments are conducted on the 27-slice inputs, and results are summarized in Table 6.3. Both architectures described above achieve inferior performance compare with our proposed method: without skip connection, it has lower sensitivities at high FP levels (*i.e.* with more than 2 false positives per input example) even compared with our baseline detector; and using 2D Conv on a concatenated feature map leads to inferior sensitivities for all FP levels.

Type	BN	AB	ME	LV	LU	KD	ST	PV
Baseline	72.69	84.07	87.27	90.04	89.70	85.73	76.99	83.50
Hybrid (Ours)	**73.84**	**84.63**	**88.43**	**91.14**	**90.50**	**86.16**	**77.91**	**85.64**

Table 6.4. Sensitivities of different types of lesion at 4 false positive per image. Our detector outperforms baseline on all 8 types.

6.4.3 Number of 3DFM Instances

3DFMs can bridge the 3D context information in the 2D CNN backbones, and can be inserted anywhere in the 2D CNNs. In our final detector, we insert 3DFMs at the *pool3* and *conv4_3* layers in VGG16 as in Figure 6.1. We also conduct diagnostic experiments by 1) inserting 3DFM at only *conv4_3* layer and 2) inserting 3DFMs at *pool2*, *pool3* and *conv4_3* layers. Results are listed as '3DFM@4' and '3DFM@234' in Table 6.3. Compared with '3DFM@4', adding another 3DFM at *pool3* significantly improve the performance from 70.62 to 71.04. However, adding an extra 3DFM at *pool2* will only give a marginal performance gain. For simplicity, we use only two 3DFMs in our final detector.

6.4.4 Analysis on Different Lesion Types

We test our hybrid detector on different lesion types in DeepLesion [15]. There are 8 types of lesion labelled in the dataset, and the abbreviations are in the parentheses: bone (BN), abdomen (AB), mediastinum (ME), liver (LV), lung (LU), kidney (KD), soft tissue (ST) and pelvis (PV). In Table 6.4, we evaluate the sensitivities of our baseline detector and our hybrid detector equipped with 3DFM, at the threshold of 4 FPs per image (given input examples of 27 slices). The results further confirm that our hybrid detector can improve the detection quality under all 8 lesion types, thus it is very general with consistent gains. We also show some qualitative results in Figure 6.5, where our baseline detector fails to detect the lesion, but the 3DFM equipped hybrid detector detects them with scores greater than 0.9 at 4 FP per image

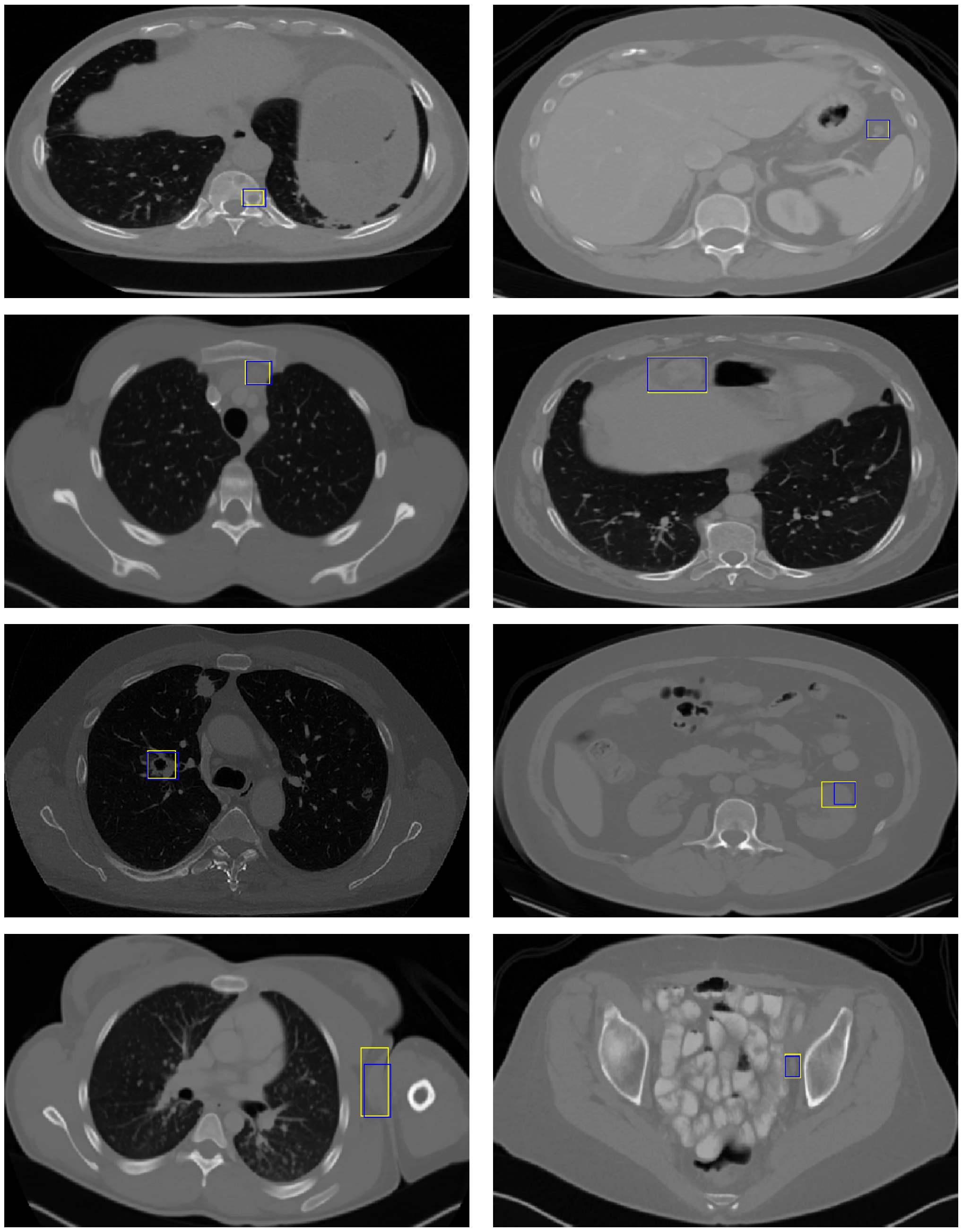

Figure 6.5. Detection examples of eight types. Yellow and blue boxes are for ground-truth and detection result. All examples are detected by our hybrid detector while missed by our baseline detector.

threshold. We observe that our detector is able to find difficult lesions such as small or low-contrast lesions.

6.5 Conclusion and Future Works

In this chapter, we focus on applying object detection for medical image analysis, especially for lesion detection given CT scan images. In order to make better use of the 3D context information existing in the CT scans, we propose a hybrid detector which bridges 3D context information in 2D CNN backbones. Based on a baseline detector which takes adjacent CT scan images independently with the same 2D CNN, we enhance the backbone feature quality by fusing 3D context knowledge via 3DFMs, which are designed and built to be lightweight and effective. Extensive experiments have been conducted on the DeepLesion dataset to show the efficacy of our hybrid detector, which improves the sensitivity at all false positive levels. The improvement is consistent under different settings (*e.g.*, number of input slices and lesion types). Qualitative analysis also suggests that our method outperforms the baseline method even for some extremely difficult cases. Our approach surpasses existing methods and thus establishes a new state-of-the-art. The superior performance demonstrates its potential usage for different clinical applications.

Future works could lie in automatically searching the architecture which has the strongest ability of exploiting 3D information among the input slices. This currently proposed method is manually designed and hand-crafted. It could not be guaranteed to be optimal, and different datasets may require different designs. A better and more universal strategy could be neural architecture search [115], which can determine 2D, 3D or pseudo 3D convolutions at each layer automatically.

Chapter 7

Spatio-Temporal-Interactive Network for Pedestrian Detection and Trajectory Prediction

Recently, more and more research attention has been attracted by autonomous driving, in which object detection plays an important role. One of the fundamental differences for detection in autonomous driving is the temporal information, as the inputs come in as sequences naturally. On the one hand, we have access to both the past and current information which is much richer than the tasks discussed in the previous chapters; on the other hand, we need to detect objects for both current and future frames, and it could be a much harder task. In this chapter, we focus on the aforementioned issue by investigating object detection in a sequence of frames and extend the detection from observed (*i.e.* current) frames into future frames, *i.e.*, predicting objects' future trajectory. Specifically, we tackle the problem of detecting pedestrians and predicting future trajectories for them. It is a critical task for numerous applications, including but not limited to autonomous driving. Previous methods either treat the detection and prediction as separate tasks which are not end-to-end trainable, or simply add a trajectory regression head on top of a detector which could not capture the detailed information before making predictions. In this work, we present a novel end-to-end two-stage network: Spatio-Temporal-Interactive Network (STINet). In addition to 3D geometry modeling of pedestrians, we model the temporal information for each of the pedestrians. To do so, our method predicts both current and past locations in the first stage, so that each pedestrian can be linked across frames and the comprehensive spatio-temporal information can be captured in the second stage. Also, we model the interaction among objects with an interaction graph, to gather the information among the neighboring objects. Comprehensive experiments on the Lyft Dataset and the recently released large-scale Waymo Open Dataset for both object detection and future trajectory prediction validate the effectiveness of the proposed method. For the Waymo Open Dataset, we achieve a bird-eyes-view (BEV) detection AP of 80.73 and trajectory prediction average displacement error (ADE) of 33.67cm for pedestrians, which establish the state-of-the-art for both tasks.

7.1 Motivation and Overview

To drive safely and smoothly, self-driving cars (SDC) not only need to detect where the objects are at the current frame (*i.e.* object detection), but also need to predict where they will be in the future (*i.e.* trajectory prediction). Among different objects, pedestrian is an important and difficult type. The difficulty comes from the complicated properties of pedestrian appearance and behavior, *e.g.* articulate shape and interpersonal relations [116]. In this chapter, we tackle the problem of joint pedestrian detection and trajectory prediction from a sequence of point clouds, as illustrated in Figure 7.1.

Traditionally, this problem is tackled by dividing the perception pipeline into multiple modules: object detection [17], [20], [21], [26], [41], [43], [44], [89], tracking [117] and trajectory prediction [116], [118], [119]; latter modules take the outputs from the former modules. Although such strategy makes each sub-module easy to design and implement, it sacrifices the potential advantage of joint optimization. Latter modules can lose critical information bottle-necked by the interfaces between sub-modules, *e.g.*

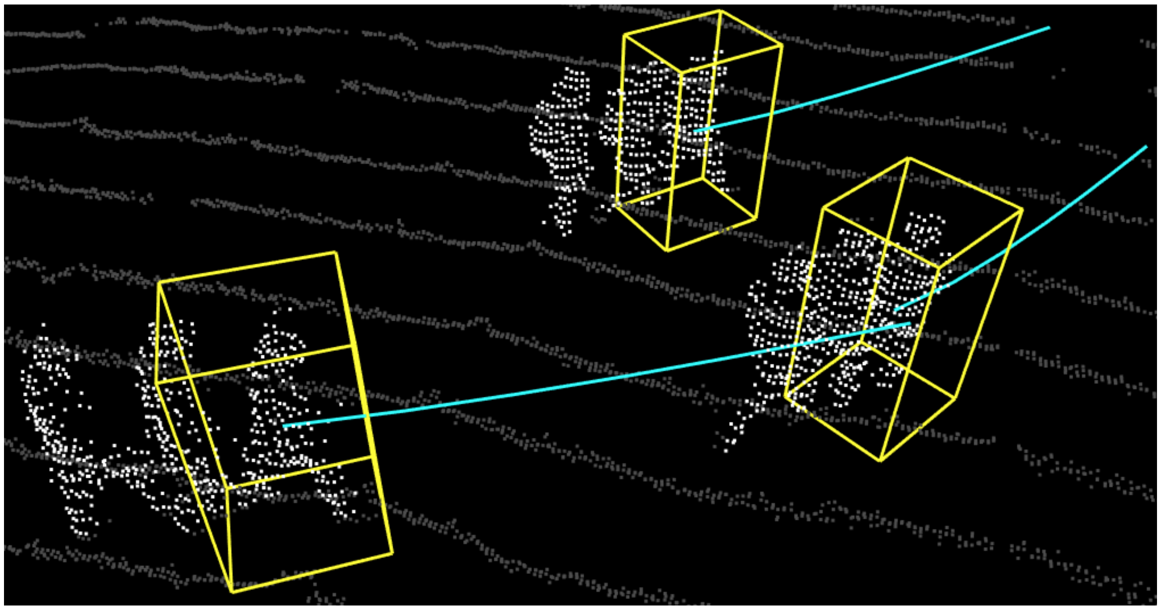

Figure 7.1. Given a sequence of current and past point clouds, our task is to detect pedestrians in the current frame, and predict the future trajectory of them. In this figure, white points are input point cloud sequence (stacked for visualization), yellow boxes are detected objects, and the cyan lines are predicted future trajectory.

a pedestrian's future trajectory depends on many useful geometry features from the raw sensor data, which may be abstracted away in the detection/tracking stage. To this end, researchers recently have proposed several end-to-end neural networks to detect objects and predict trajectories simultaneously. FaF [45] and IntentNet [46] are two of the representative methods, which are designed based on single stage detectors (SSD) [17]; in addition to original anchor classification and regression of SSD, they also regress a future trajectory for each anchor.

We observed that there are two major issues that are critical for joint detection and trajectory prediction, but are not addressed by previous end-to-end methods: 1) Temporal modeling on object level: existence and future trajectory of an object are embedded in both current and past frames. Current methods simply reuse single-stage detector and fuse the temporal information in the backbone CNN in an object-agnostic manner either via feature concatenation or 3D CNN [45], [46]. Such coarse level fusion can loss fine-grained temporal information for each object, which is critical for both tasks. 2) Interaction modeling among objects: the future trajectory of an object could be influenced by the other objects. *E.g.*, a pedestrian walking inside a group may tend to follow others. Existing methods [45], [46] do not explicitly model interactions among objects.

To address the aforementioned issues, we propose an end-to-end Spatio-Temporal-Interactive network (STINet) to model pedestrians' temporal and interactive information jointly. The proposed network takes a sequence of point clouds as input, detects current location and predicts future trajectory for pedestrians. Specifically, there are three sub-components in STINet: backbone network, temporal proposal generation network, and proposal prediction network. In the backbone net, we adopt a similar structure as PointPillars [43], and apply it on each frame of the point cloud, the output feature maps from multi-frames are then combined. The temporal proposal generation network takes feature maps from the backbone net and generates potential pedestrian instances with both their current and past locations (*i.e.* temporal proposals); such temporal proposals allow us to link the same object across different frames. In the third module (*i.e.* prediction network), we use the temporal proposals to explicitly gather the geometry appearance and temporal dynamics for each object. To reason the interaction among pedestrians, we build a graph layer to gather the information from surrounding pedestrians. After extracting the above spatial-temporal-interactive features for each proposal, the detection and prediction head uses the feature to regress current detection bounding box and future trajectory.

Comprehensive experiments are conducted on Waymo Open Dataset [120] and Lyft Dataset [4] to demonstrate the effectiveness of the STINet. Specifically, it achieves an average precision of 80.73 for bird-eyes-view pedestrian detection, and an average displacement error of 33.67 cm for trajectory prediction on Waymo Open Dataset. It achieves real-time inference speeds and takes only 74.6 ms for inference on a range of 100m by 100m.

The main contributions of our work come in four folds:

- We build an end-to-end network tailored to model pedestrian past, current and future simultaneously.
- We propose to generate temporal proposals with both current and past boxes. This enables learning a comprehensive spatio-temporal representation for pedestrians with their geometry, dynamic movement and history path in an end-to-end manner without explicitly associating object across frames.
- We propose to build a graph among pedestrians to reason the interactions to further improve trajectory prediction quality.
- We establish the state-of-the-art performance for both detection and trajectory prediction on the Lyft Dataset and the recent large-scale challenging Waymo Open Dataset.

7.2 Related Work

7.2.1 Temporal Proposals

Temporal proposals have been shown beneficial in action localization in [121], [122]. They show that associating temporal proposals from different video clips can help to leverage the temporal continuity of video frames. Tang *et al*. [123] proposes to link temporal proposals throughout the video to improve video object detection. In our work, we also exploit temporal proposals and step further to investigate and propose how to build comprehensive spatio-temporal representations of proposals to improve future trajectory prediction. This is a hard task since there are no inputs available for the future. Also we investigate to learn interactions between proposals via a graph. We show that these spatio-temporal features can effectively model objects' dynamics and provide accurate detection and prediction of their future trajectory.

7.2.2 Relational Reasoning

An agent's behavior could be influenced by other agents and it is naturally connected to relational reasoning [124], [125]. Graph neural networks have shown its strong capability in relational modeling in recent years. Wang *et al.* [126] formulates the video as a space-time graph, show the effectiveness on the video classification task. Sun *et al.* [127] designs a relational recurrent network for action detection and anticipation. Yang *et al.* [128] proposes to build an object relationship graph for the task of scene graph generation.

7.2.3 Trajectory Prediction

Predicting the future trajectory of objects is an important task, especially for autonomous driving. Previous research has been conducted based on perception objects as inputs [116], [118], [119], [129], [130]. Recently FaF [45] and IntentNet [46] focus on end-to-end trajectory prediction from raw point clouds as input. However, they simply re-use single-stage detection framework and add new regression heads on it. In our work, we exploit temporal region proposal network and explicitly model Spatio-Temporal-Interaction (STI) representations of pedestrians, and our experiments show that the proposed STI modeling is superior on both detection and trajectory prediction for pedestrians.

7.3 Methodology

In this section, we discuss our proposed network in details. The overview of our proposed method is shown in Figure 7.2, which can be divided into three steps. For each of these steps, we discuss in the following subsections.

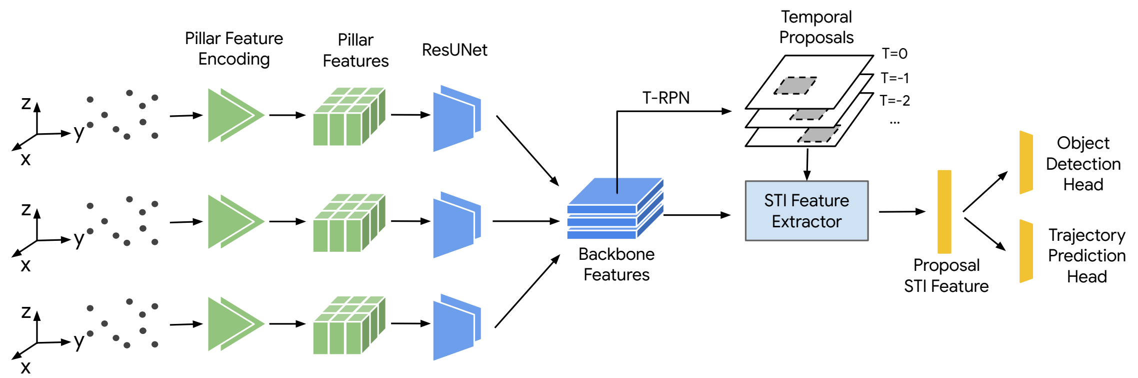

Figure 7.2. The overview of the proposed method. It takes a sequence of point clouds as input, detects pedestrians and predicts their future trajectories simultaneously. The point clouds are processed by Pillar Feature Encoding [41], [43] to generate Pillar Features. Then each Pillar Feature is fed into a backbone ResUNet [131] to get backbone features. A Temporal Region Proposal Network (T-RPN) takes backbone features and generated temporal proposal with past and current boxes for each object. Spatio-Temporal-Interactive (STI) Feature Extractor learns features for each temporal proposal which are used for final detection and trajectory prediction.

7.3.1 Backbone Network

The backbone of our network is illustrated in Figure 7.3. The input is a sequence of point clouds with t' frames noted as $[\mathrm{PC}_{-(t'-1)}, \mathrm{PC}_{-(t'-2)}, \cdots, \mathrm{PC}_0]$, which corresponds to the lidar sensor input from the past $t'-1$ frames as well as the current frame. All point clouds are calibrated to SDC's pose at the current frame so that the ego-motion is discarded. To build rich pillar features while keeping a feasible memory usage, we generate t pillar features from the t' input frames. Consecutive t'/t point clouds $\mathrm{PC}_{-(j+1)t'/t+1}, \cdots, \mathrm{PC}_{-jt'/t}$ are processed with Voxelization [41], [43] and then concatenated to generate a pseudo image I_j (*i.e.* Pillar Features) with shape $H \times W \times C_{in}$. Thus the output of Pillar Feature Encoding is a sequence of t Pillar Features $[I_{-(t-1)}, I_{-(t-2)}, \cdots, I_0]$.

Next we adopt a similar backbone CNN network proposed in [131], as shown in the lower part of Figure 7.3. Each of the Pillar Features I_j is first processed by three ResNet-style blocks to generate intermediate features with shape $\mathbb{R}^{H\times W\times C_0}$,$\mathbb{R}^{\frac{1}{2}H\times\frac{1}{2}W\times C_1}$ and

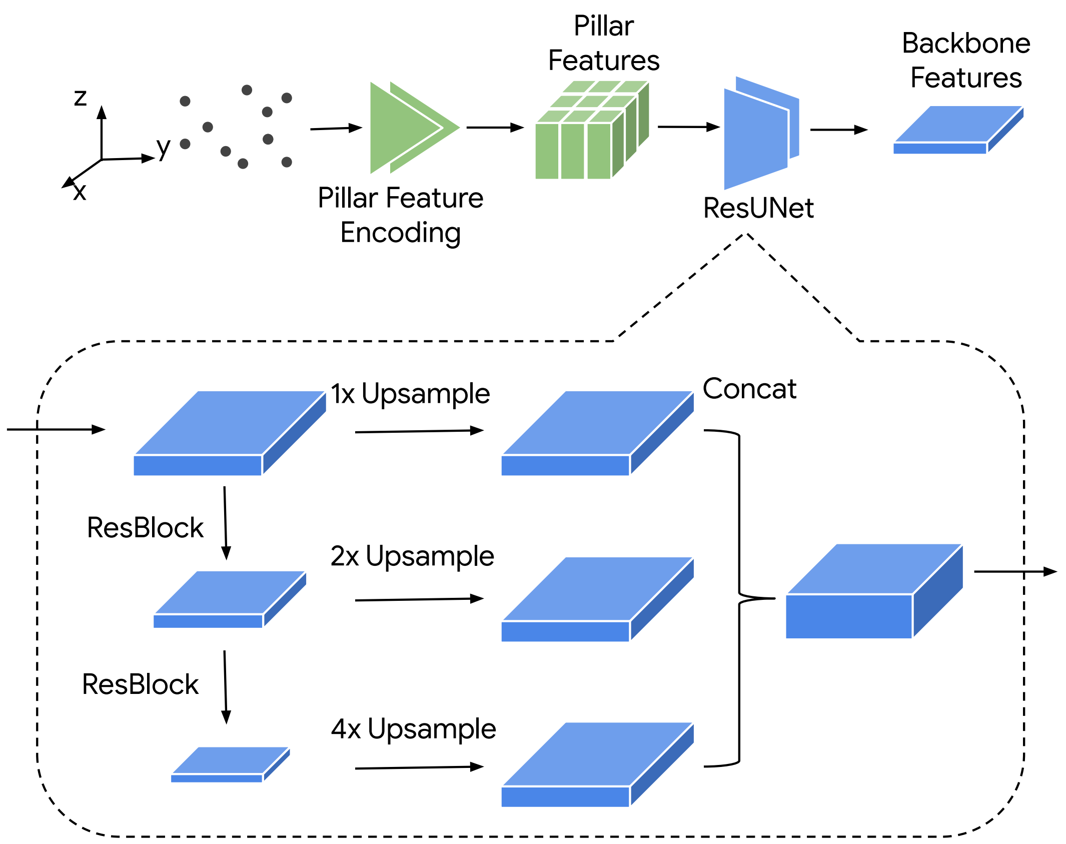

Figure 7.3. Backbone of proposed network. Upper: overview of the backbone. The input point cloud sequence is fed to Voxelization and Point net to generate pseudo images, which are then processed by ResNet U-Net to generate final backbone feature sequence. Lower: detailed design of ResNet U-Net.

$\mathbb{R}^{\frac{1}{4}H \times \frac{1}{4}W \times C_2}$. Then we use deconvolution layers to upsample them to the same spatial shape with I_j. The concatenation of the upsampled features serve as the backbone feature of I_j, noted as B_j.

7.3.2 Temporal Proposal Generation

In order to explicitly model objects' current and past knowledge, we propose a temporal region proposal network (T-RPN) to generate object proposals with both current and past boxes (*i.e.*, temporal proposals). T-RPN takes the backbone feature sequence $[B_{-(t-1)}, B_{-(t-2)}, \cdots, B_0]$ as the input, concatenates them in the channel dimension and applies a 1×1 convolution to generate a temporal-aware feature map. Classification, current frame regression and past frames regression are generated by applying 1×1 convolutional layers over the temporal-aware feature map, to classify and regress the pre-defined anchors.

The temporal region proposal network is supervised by the ground-truth objects' current and past locations. For each anchor $\mathbf{a} = (x^a, y^a, w^a, l^a, h^a)$ (x, y, w, l, h correspond to x coordinate of box center, y coordinate of box center, width of box, length of box and heading/yaw of box respectively, and we keep this notation for the rest of this chapter), it is assigned to a ground-truth object with largest IoU of the current frame box $\mathbf{gt} = (x_0^{gt}, y_0^{gt}, w^{gt}, l^{gt}, h_0^{gt})$. Similar to SECOND [42], we compute the regression target in order to learn the difference between the pre-defined anchors and the corresponding ground-truth boxes. For the current frame, we generate a 5-d regression target $\mathbf{d}_0^a = (dx_0^a, dy_0^a, dw^a, dl^a, dh_0^a)$ for each anchor/ground-truth pair:

$$dx_0^a = (x_0^{gt} - x^a)/\sqrt{(x^a)^2 + (y^a)^2} \tag{7.1}$$

$$dy_0^a = (y_0^{gt} - y^a)/\sqrt{(x^a)^2 + (y^a)^2} \tag{7.2}$$

$$dw^a = \log \frac{w^{gt}}{w^a} \tag{7.3}$$

$$dl^a = \log \frac{l^{gt}}{l^a} \tag{7.4}$$

$$dh_0^a = \sin \frac{h_0^{gt} - h^a}{2} \tag{7.5}$$

With similar equations, we also compute $t-1$ past regression targets for anchor $\mathbf{a}$ against the same ground-truth object: $\mathbf{d}_j^a = (dx_j^a, dy_j^a, dh_j^a)$ for $j \in \{-1, -2, \cdots, -(t-1)\}$. Width and length are not considered for the past regression since we assume the object size does not change across different frames. For each anchor $\mathbf{a}$, the classification target s^a is assigned as 1 if the assigned ground-truth object has an IoU greater than th^+ at the current frame. If the IoU is smaller than th^- (where $th^+ >= th^-$), classification target is assigned as 0. Otherwise the classification target is -1 and the anchor is ignored for computing losses and gradients.

For each anchor $\mathbf{a}$, T-RPN predicts a classification score $\hat{s}^a$, a current frame regression vector $\hat{\mathbf{d}}_0^a = (\hat{dx}_0^a, \hat{dy}_0^a, \hat{dw}^a, \hat{dl}^a, \hat{dh}_0^a)$ and $t-1$ past frame regression vectors $\hat{\mathbf{d}}_j^a = (\hat{dx}_j^a, \hat{dy}_j^a, \hat{dh}_j^a)$ from the aforementioned 1×1 convolutional layers. The objective of T-RPN is the weighted sum of classification loss, current frame regression loss and past frame regression losses as defined in the equations below, where $\mathbb{1}(x)$ is the indicator function and returns 1 if x is true otherwise 0.

$$\mathcal{L}_{\text{T-RPN}} = \lambda_{\text{cls}}\mathcal{L}_{\text{cls}} + \lambda_{\text{cur\_reg}}\mathcal{L}_{\text{cur\_reg}} + \lambda_{\text{past\_reg}}\mathcal{L}_{\text{past\_reg}} \tag{7.6}$$

$$\mathcal{L}_{\text{cls}} = \frac{\sum_{\mathbf{a}} \text{CrossEntropy}(s^a, \hat{s}^a)\mathbb{1}(s^a \geq 0)}{\sum_{\mathbf{a}} \mathbb{1}(s^a \geq 0)} \tag{7.7}$$

$$\mathcal{L}_{\text{cur\_reg}} = \frac{\sum_{\mathbf{a}} \text{SmoothL1}(\mathbf{d}_0^a, \hat{\mathbf{d}}_0^a)\mathbb{1}(s^a \geq 1)}{\sum_{\mathbf{a}} \mathbb{1}(s^a \geq 1)} \tag{7.8}$$

$$\mathcal{L}_{\text{past\_reg}} = \sum_{j=1}^{t-1} \frac{\sum_{\mathbf{a}} \text{SmoothL1}(\mathbf{d}_{-j}^a, \hat{\mathbf{d}}_{-j}^a)\mathbb{1}(s^a \geq 1)}{\sum_{\mathbf{a}} \mathbb{1}(s^a \geq 1)} \tag{7.9}$$

For proposal generation, predicted classification scores and bounding-box regression vectors are applied on pre-defined anchors to generate temporal proposals, by reversing Equations 7.1-7.5 (*i.e.*, as the following equations). Thus each temporal proposal has a confidence score as well as the regressed boxes for the current and past frames. After that, non-maximum suppression is applied on the current frame boxes of temporal proposals to remove redundancy.

$$\hat{x}_j = x^a + \hat{dx}_j^a \cdot \sqrt{(x^a)^2 + (y^a)^2} \tag{7.10}$$

$$\hat{y}_j = y^a + \hat{dy}_j^a \cdot \sqrt{(x^a)^2 + (y^a)^2} \tag{7.11}$$

$$\hat{w} = w^a \cdot \exp\{\hat{dw}^a\} \tag{7.12}$$

$$\hat{l} = l^a \cdot \exp\{\hat{dl}^a\} \tag{7.13}$$

$$\hat{h}_j = h^a + \arcsin(2\hat{dh}_j^a) \tag{7.14}$$

7.3.3 Proposal Prediction

7.3.3.1 Spatio-Temporal-Interactive Feature Extraction

Given the backbone features $[B_{-(t-1)}, \cdots, B_0]$ and the temporal proposals, spatio-temporal-interactive features are learned for each temporal proposal to capture the comprehensive information for detection and trajectory prediction. Different ways for modeling objects are combined to achieve this.

Local geometry feature: To extract object geometry knowledge, we use the proposal boxes at j-th frame (*i.e.*, x_j, y_j, w, l, and h_j) to crop features from B_j, as shown in the lower left part of Figure 7.4. This is an extension of traditional proposal feature cropping used in Faster-RCNN [26], to gather position-discarded local geometry features from each frame. To simplify the implementation on TPU, we rotate the 5-DoF box (x_j, y_j, w, l, h_j) to the closest standing (axis-aligned) box $(x_{min,j}, y_{min,j}, x_{max,j}, y_{max,j})$ for ROIAlign [13].

Local dynamic feature: As illustrated in the lower middle part of Figure 7.4, we use a meta box (drawn in yellow) which covers the whole movement of the pedestrian across all past and current frames to crop features from all B_j's. The meta box is the smallest box which contains all current and past proposal boxes. Formally, after

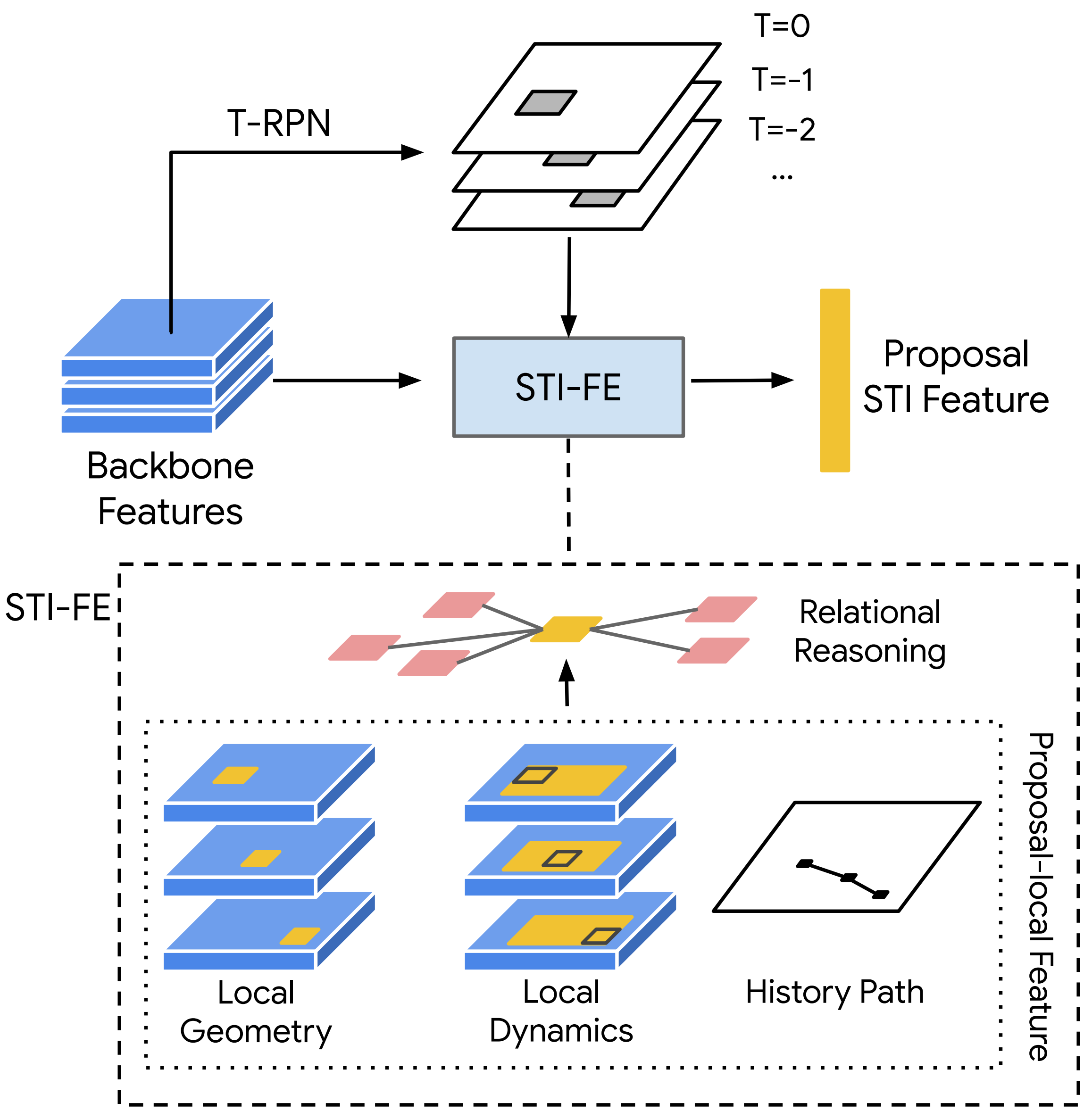

Figure 7.4. Spatial-Temporal-Interactive Feature Extractor (STI-FE): Local geometry, local dynamic and history path features are extracted given a temporal proposal. For local geometry and local dynamics features, the yellow areas are used for feature extraction. Relational reasoning is performed across proposals' local features to generate interactive features.

transferring all rotated proposal boxes (x_j, y_j, w, l, h_j) to the closest standing boxes $(x_{min,j}, y_{min,j}, x_{max,j}, y_{max,j})$, the meta box is computed with the following equations:

$$\begin{aligned} x_{min} &= \min_j(x_{min,j});\ y_{min} = \min_j(y_{min,j}) \\ x_{max} &= \max_j(x_{max,j});\ y_{max} = \max_j(y_{max,j}) \end{aligned} \tag{7.15}$$

This feature captures the direction, curvature and speed of the object, which are useful for future trajectory prediction.

History path feature: In order to directly encode objects' past movement, we exploit the location displacement across different history frames as the history path feature. To be specific, given a temporal proposal with x_j, y_j as the box centers, the history path feature is $\text{MLP}([x_0 - x_{-1}, y_0 - y_{-1}, x_0 - x_{-2}, y_0 - y_{-2}, \cdots, x_0 - x_{-(t-1)}, y_0 - y_{-(t-1)}])$.

To aggregate spatial and temporal knowledge for each proposal, the concatenation of local geometry feature and the local dynamic feature is fed into a ResNet block followed by a global average pooling. The pooled feature is then concatenated with the history path feature, and serves as the proposal-local feature, noted as f_i for the i-th temporal proposal.

As discussed before, the future trajectory of a pedestrian could be influenced by the surrounding pedestrians' behaviors. In order to model such interactions among pedestrians, we design an interaction layer which uses a graph to propagate information among objects, as shown in the middle part of Figure 7.4. Specifically, we represent each temporal proposal as a graph node i; the embedding of node i is noted as f_i, which is the corresponding proposal-local feature. The edge v_{ij} represents the interaction score between node i and node j. v_{ij} is learned from f_i and f_j, which can be represented as below.

$$v_{ij} = \alpha([\phi_1(f_i); \phi_2(f_j)]) \tag{7.16}$$

where α and ϕ's can be any learnable functions. In our implementation, we use fully-connected layer for α and ϕ's.

Given the interaction scores among all pairs of nodes, we can gather the information for each node from the neighboring nodes. Specifically, the interaction embedding g_i gathered for node i is calculated as follows:

$$g_i = \sum_j \frac{\exp\{v_{ij}\}}{V_i} \gamma([f_i; f_j]) \tag{7.17}$$

where $V_i = \sum_j \exp\{v_{ij}\}$ is the normalization constant, and γ is a mapping function (a fully-connected layer is adopted in our implementation).

7.3.3.2 Proposal Classification and Regression

Given proposal-local features f_i for each temporal proposals, two fully-connected layers are applied to do classification and regression respectively for the current frame. To be aligned with our intuitions, the proposal-local feature f_i combined with the interaction feature g_i is used to predict future frame boxes, by one fully-connected layer with $3t$ output channels where t is the number of future frames to predict and 3 stands for x coordinate, y coordinate and heading respectively. During the training, temporal proposals are assigned classification and regression targets with the same strategy discussed in Subsection 7.3.2 and the objective is the weighted sum of classification loss, current frame regression loss and future frames regression loss similar to Equations 7.6-7.9. During inference, each proposal is predicted with a classification score and current/future boxes. Non-maximum suppression is applied on them based on the IoU between their current boxes, to remove redundancy.

7.4 Experiment

7.4.1 Experiment Settings

Dataset: We conduct experiments on the Waymo Open Dataset (WOD) [120] and the Lyft Dataset (Lyft) [4]. WOD contains lidar data from 5 sensors as well as the bounding-box labels for 1000 segments. Each segment contains roughly 200 frames and has a length of 20 seconds. Train and validation subsets have 798 and 202 segments respectively. To model the history and predict the future, we take 1-second history frames and 3-second future frames for each example and extract examples from the center 16 seconds (*i.e.*, 1s∼17s) from each segment. Thus 126,437 train examples and 31,998 validation examples are extracted, and each of them contains history frames of 1 second and future frames of 3 seconds. We sample 6 frames including 5 history frames and the current frame, with $t_{\text{input}} = \{-1.0, -0.8, -0.6, -0.4, -0.2, 0\}$, and the point clouds from those frames are fed into the network as inputs. In order to build richer voxel features while saving computation and memory, every two frames are combined by concatenating the voxelization output features thus we have three pillar features as discussed in Subsection 7.3.1. For the future trajectory prediction, we predict trajectory for 6 future frames with $t_{\text{future}} = \{0.5, 1.0, 1.5, 2.0, 2.5, 3.0\}$. The range is 150m by 150m around the self-driving car, and we use a pillar size of 31.25cm by 31.25cm to generate pillar features of shape 480×480. Lyft contains lidar data from 1 sensor and labels for only 180 segments, with 140 and 40 segments for train and validation respectively. With the same settings, 14,840 and 4,240 examples are extracted for train and validation. Each example has 1-second history and 3-second future. We have $t_{\text{future}} = \{0.6, 1.2, 1.8, 2.4, 3.0\}$ for Lyft due to its 5Hz sampling rate.

Evaluation metric: The evaluation metric for the detection task is BEV AP (Bird-Eyes-View Average Precision) with the IoU threshold set to 0.5. Objects with fewer than 5 points inside the bounding-boxes are considered difficult and thus excluded during the evaluation. For trajectory prediction, we employ the metrics used in [46], [119]. For $t \in t_{\text{future}}$, we compute the DE@t (Displacement Error) and the HR@t (Hit Rate) with a displacement error threshold of 0.5m. We also compute the ADE (Average Displacement Error) which equals to $\frac{1}{|t_{\text{future}}|} \sum_{t \in t_{\text{future}}} \text{DE@}t$.

Implementation: Our models are implemented in TensorFlow and we train the model with Adam optimizer on TPUv3 for 140k and 70k iterations for Waymo Open Dataset and Lyft Dataset respectively. The learning rate is 4×10^{-4} and the batch-size is 1 per TPU. We use 32 TPU cores together for the training, thus the effective batch-size is 32. We also implement IntentNet [46] and Faster-RCNN [26] in TensorFlow as the baselines, which are noted as "IntentNet" and "MF-FRCNN". Our implemented IntentNet (1) takes multiple frames as input and share the same backbone net as STINet; (2) removes the intent classification part, and only regresses a future trajectory.

Model	MF	TS	BEV AP ↑
PointPillar [44]			68.57
MVF [44]			74.38
StarNet [40]			72.50
IntentNet [46][1]	✓		$79.43_{\pm 0.10}$
MF-FRCNN	✓	✓	$79.69_{\pm 0.19}$
STINet	✓	✓	$\mathbf{80.73_{\pm 0.26}}$

Table 7.1. Detection performance for different methods on WOD. MF indicates whether the corresponding model takes multiple frames as input. TS indicates whether the model has a two-stage framework. BEV AP is computed with an IoU threshold of 0.5. ↑ indicates the higher numbers are better for the corresponding metric.

MF-FRCNN refers to a Faster-RCNN [26] model with several changes: (1) It uses the same backbone net as STINet, please refer to Section 7.3.1; (2) for each object proposal, in addition to the bounding box, we also regress future trajectories and headings. Note that the difference between proposals from MF-FRCNN and our method is that MF-FRCNN only predicts the current box of objects, while our method exploits a novel Temporal RPN which also generates the corresponding history boxes associated to each current box.

7.4.2 Results on Waymo Open Dataset

The main results on Waymo Open Dataset of pedestrian detection and trajectory prediction are summarized in Table 7.1 and Table 7.2. For detection we compare our proposed method (in the last row) with the current state-of-the-art detectors [40], [44] and our method surpasses the off-the-shelf baselines by a very large margin, improving the BEV AP from 74.38 to 80.73. To analyze the contributions other than multi-frame inputs and different implementation details, we also compare STINet with our own implementation of IntentNet [46] and multi-frame Faster RCNN [26], which are noted as “IntentNet” and “MF-FRCNN” in the relevant tables. Our proposed method outperforms all baselines, and it confirms the effectiveness of our T-RPN and

[1]IntentNet without intent prediction head implemented by us.

Model	MF	TS	DE@1 ↓	DE@2 ↓	DE@3 ↓	ADE ↓
IntentNet	✓		$21.17_{\pm 0.02}$	$39.74_{\pm 0.07}$	$61.60_{\pm 0.12}$	$36.04_{\pm 0.12}$
MF-FRCNN	✓	✓	$20.87_{\pm 0.08}$	$39.23_{\pm 0.14}$	$60.59_{\pm 0.22}$	$35.57_{\pm 0.13}$
STINet	✓	✓	$\mathbf{19.63}_{\pm \mathbf{0.03}}$	$\mathbf{37.07}_{\pm \mathbf{0.08}}$	$\mathbf{57.60}_{\pm \mathbf{0.14}}$	$\mathbf{33.67}_{\pm \mathbf{0.07}}$

Model	MF	TS	HR@1 ↑	HR@2 ↑	HR@3 ↑
IntentNet	✓		$93.18_{\pm 0.03}$	$76.50_{\pm 0.08}$	$61.60_{\pm 0.12}$
MF-FRCNN	✓	✓	$93.45_{\pm 0.05}$	$76.69_{\pm 0.18}$	$61.57_{\pm 0.21}$
STINet	✓	✓	$\mathbf{94.36}_{\pm \mathbf{0.05}}$	$\mathbf{78.91}_{\pm \mathbf{0.06}}$	$\mathbf{64.43}_{\pm \mathbf{0.15}}$

Table 7.2. Trajectory prediction performance for different models on WOD. MF indicates whether the corresponding model takes multiple frames as input. TS indicates whether the model has a two-stage framework. ↑ and ↓ indicate the higher/lower numbers are better for the corresponding metric. DE and ADE are in centimeters. For models implemented by us, we train and evaluate the model for five times and compute the average and standard deviation shown around ± in the table.

Model	0∼2.5	2.5∼5	5∼7.5	7.5∼10	10∼ ∞
MF-FRCNN	63.07	90.44	93.27	88.00	77.15
STINet	**64.23**	**91.15**	**94.46**	**88.97**	**80.50**
Δ%	1.8%	0.8%	1.3%	1.1%	4.3%

Table 7.3. Bird-eyes-view average precision (BEV-AP) breakdown comparison of MF-FRCNN and STINet on WOD. Objects are split into five bins base on the future trajectory length with a bin size of 2.5m. Last row is the relative improvement of STINet.

the STI modeling of proposals.

In Table 7.2 we compare the trajectory prediction performance among our proposed method, IntentNet and MF-FRCNN. Our proposed method surpasses all competitors by a large margin, and the improvement is larger than the improvement on detection. It aligns with our intuition since T-RPN and STI modeling are designed to better model objects' movement and more useful to forecast their trajectory.

For a detailed comparison of STINet and MF-FRCNN, we evaluate the detection and trajectory prediction by breaking down the objects into five bins based on the future trajectory length in 3s. The five bins are 0∼2.5m, 2.5∼5m, 5∼7.5m, 7.5∼10m and 10m∼ ∞ respectively. We report BEV AP, ADE and the relative improvement in Table 7.3 and 7.4. The STINet is consistently better than MF-FRCNN for both tasks.

Model	0∼2.5	2.5∼5	5∼7.5	7.5∼10	10∼ ∞
MF-FRCNN	26.90	37.56	46.39	104.60	173.50
STINet	**26.73**	**35.42**	**41.18**	**89.74**	**137.17**
Δ%	0.6%	6.0%	11.2%	14.2%	20.9%

Table 7.4. Average displacement error (ADE, in centimeters) breakdown comparison of MF-FRCNN and STINet on WOD. Objects are split into five bins base on the future trajectory length with a bin size of 2.5m. Last row is the relative improvement of STINet.

Model	BEV AP ↑	DE@3 ↓	ADE ↓	HR@3 ↑
MF-FRCNN	33.90	82.61	51.11	49.74
STINet	**37.15**	**76.17**	**46.09**	**50.73**

Table 7.5. Detection and trajectory prediction performance on Lyft. The same metrics are used to evaluate detection and trajectory prediction performance.

For trajectory prediction on objects moving more than 5m, the relative improvements are significant and consistently more than 10%. It confirms that the proposed method can leverage the details of history information and provide much better trajectory predictions, especially for pedestrians with a larger movement.

7.4.3 Results on Lyft Dataset

The detection and trajectory prediction results on the Lyft Dataset are summarized in Table 7.5. The performances on both tasks are improved largely and the results confirm the effectiveness of proposed method a smaller-scale dataset.

7.4.4 Ablation Studies

In this subsection we conduct ablation experiments to analyze the contribution of each component and compare our model with potential alternative methods on the Waymo Open Dataset. The results are summarized below. For clarity, we only show DE@3, ADE and HR@3 for trajectory prediction. The other metrics have the same tendency.

Effect of local geometry and local dynamic features: The local geometry and local dynamic features are important component of STI modeling of temporal proposals, as introduced in Subsection 7.3.3.1. We conduct experiments to analyze

LG	LD	BEV AP ↑	DE@3 ↓	ADE ↓	HR@3 ↑
✓		80.38	64.15	37.67	58.46
	✓	79.69	59.71	34.96	62.22
✓	✓	**80.53**	**58.95**	**34.49**	**62.99**

Table 7.6. Ablation studies on local geometry and local dynamic features (noted as LG and LD in the table respectively). All entries are trained without History Path and Interactive features.

L+G	Path	DE@3 ↓	ADE ↓	HR@3 ↑
✓		58.95	34.49	62.99
✓	✓	**58.04**	**33.92**	**63.87**
†	✓	67.80	39.86	52.25

Table 7.7. Ablation studies on history path feature. † indicates the corresponding feature is used only for detection and ignored while prediction the trajectory.

Breakdown	I	DE@3 ↓	ADE ↓	HR@3 ↑
All		58.04	33.92	63.87
	✓	**57.60**	**33.67**	**64.43**
	Δ%	0.76%	0.74%	0.88%
Group		49.67	30.85	64.87
	✓	**48.89**	**30.40**	**65.55**
	Δ%	1.57%	1.46%	1.05%

Table 7.8. Ablation studies on interaction features. ‘I’ indicates whether the proposal interaction modeling is adopted. “All" and “Group" correspond to evaluation on all pedestrians and pedestrians belonging to a group with at least 5 pedestrians respectively.

the effect of local geometry and local dynamic features, summarized in Table 7.6. As showed in the table, the local geometry feature is good at detection and the local dynamic feature is good at trajectory prediction. Geometry feature itself does not work well for trajectory prediction since it ignores dynamics for better detection. By combining both of the features, the benefits in detection and trajectory prediction can be obtained simultaneously.

Effect of history path: Although objects’ geometry and movement are already represented by local geometry dynamic features, taking history path as an extra feature can give another performance gain by improving the DE@3 from 58.95 to 58.04

and the HR@3 from 62.99 to 63.87 (as shown in the first two row of Table 7.7). This suggests the history path, as the easiest and most direct representation of objects' movement, can still help based on the rich representations. However history path itself is far from enough to give accurate trajectory prediction. We conduct an experiment to use only history feature for trajectory prediction, and the results are much worse compared with other baselines which take detailed features for trajectory prediction, as shown in the last row of Table 7.7.

Effect of proposal interaction modeling: To demonstrate the effectiveness of the proposed pedestrian interaction modeling, we measure the performance for all pedestrians as well as pedestrians in a group. Specifically, we design a heuristic rule (based on locations and speeds) to discover pedestrian groups and assign each pedestrian a group label on the evaluation set. We evaluate the trajectory prediction performance on all pedestrians and the pedestrians belonging to a group with at least 5 pedestrians, shown in Table 7.8. The interaction modeling improves trajectory prediction performance on "all pedestrians" and achieve a larger boost for pedestrians that belong to groups (*e.g.*, DE@3 improved from 49.67 to 48.89 by 1.57%).

7.4.5 Model Inference Speed

We measure the inference speed of our proposed model as well as baseline models on context range of 100m by 100m as well as 150m by 150m. All models are implemented in TensorFlow and the inference is executed on a single nVIDIA Tesla V100 GPU. For the context range of 100m by 100m, IntentNet, MF-FRCNN and STINet have inference time of 60.9, 69.4 and 74.6ms respectively. Both two-stage models (MF-FRCNN and STINet) are slower than the single-stage model, and STINet is slightly slower than MF-FRCNN. However, all three models can achieve a real-time inference speed higher than 10Hz. For the maximum range of Waymo Open Dataset, *i.e.*, 150m by 150m, three models have inference time of 122.9, 132.1 and 144.7ms respectively.

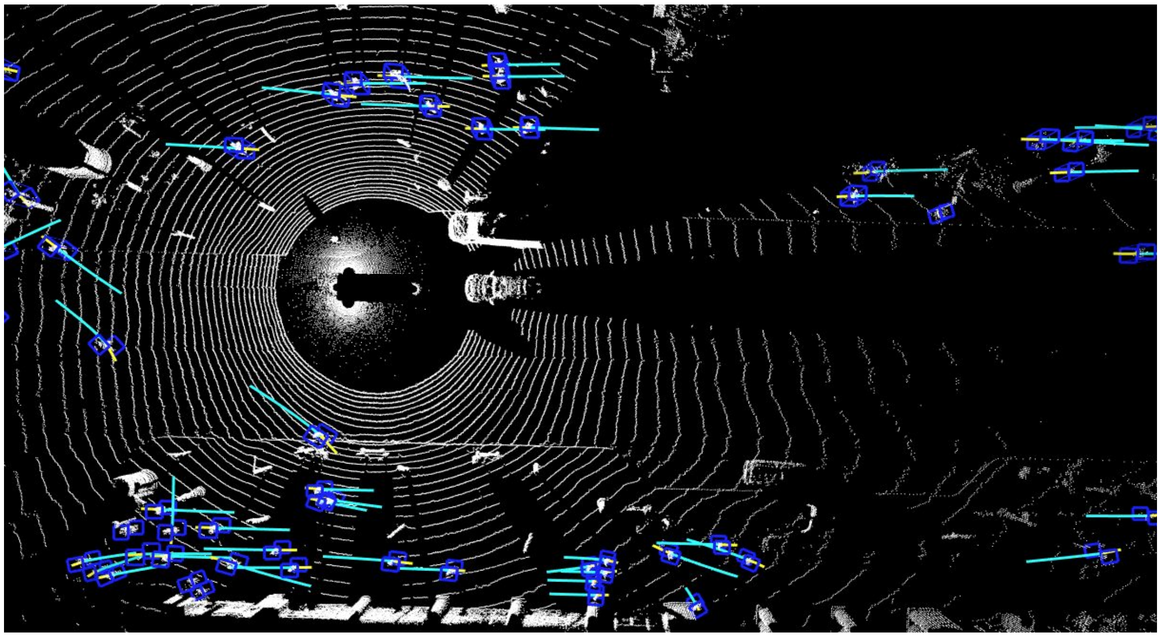
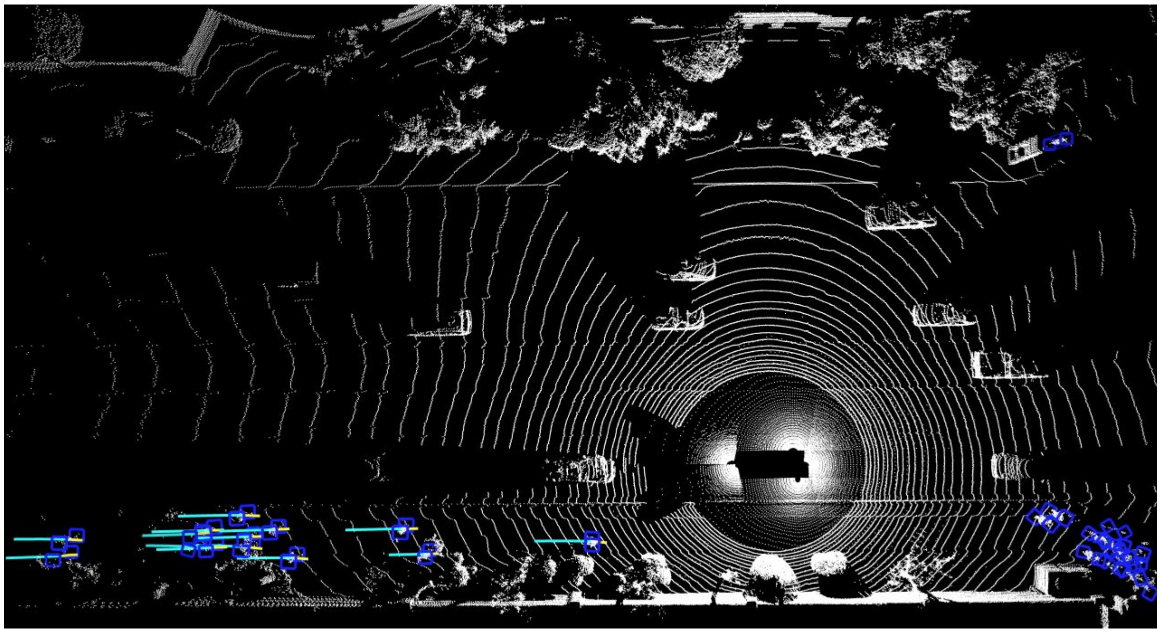

Figure 7.5. Qualitative examples of STINet. The blue box are detected pedestrians. The cyan and yellow lines are predicted future and history trajectories of STINet respectively.

7.4.6 Qualitative Results

The visualization for the predictions of STINet is shown in Figure 7.5. The blue boxes are the detected pedestrians. The cyan and yellow lines are the predicted future and history trajectory for each detected pedestrian respectively. We show two scenarios

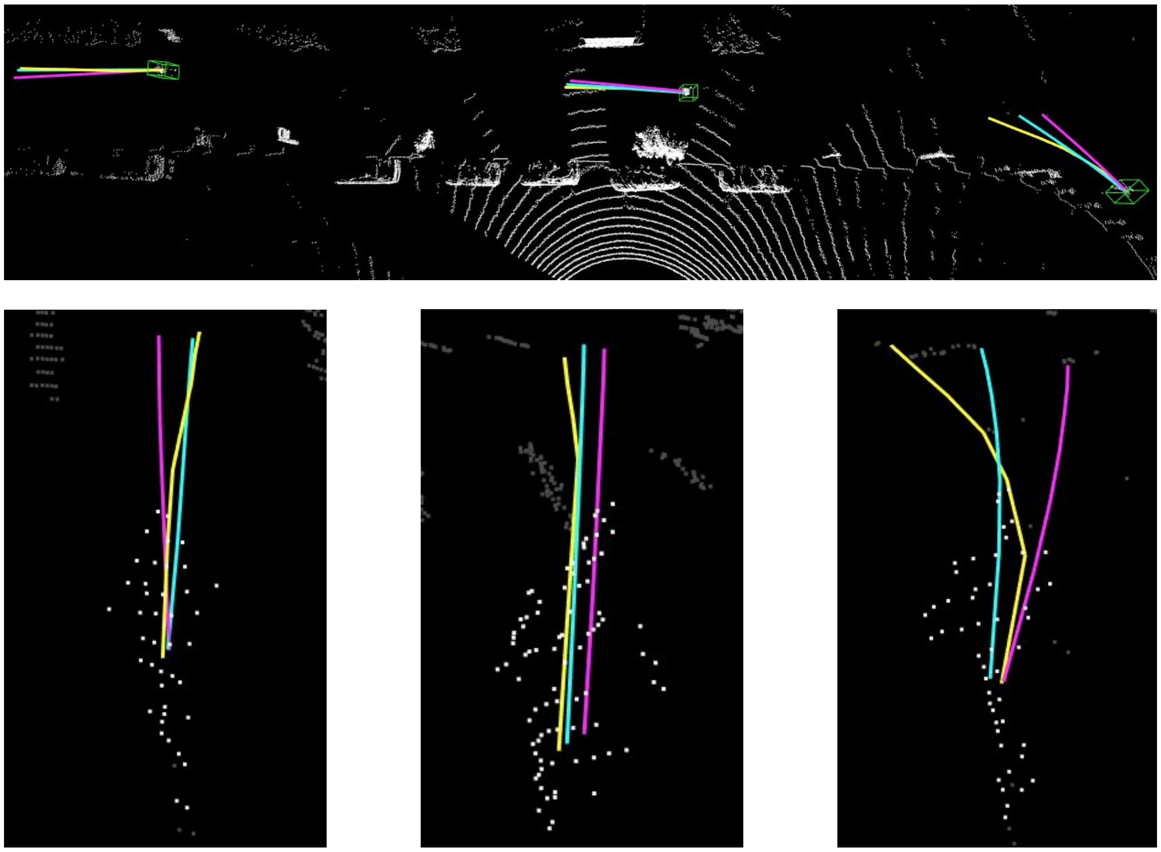

Figure 7.6. Comparison between MF-FRCNN and STINet. The yellow line is the ground-truth future trajectory for pedestrians. The pink and cyan lines are the predicted future trajectory from MF-FRCNN and STINet respectively. It is clear that our proposed method gives a much better prediction compared with the baseline, for all three pedestrians. Upper: the overview of three pedestrians. Lower: zoom-in visualization for three pedestrians.

where the SDC is stationary in the upper sub-figure and the SDC is moving fast in the lower sub-figure. It demonstrates that our model detects and predicts very accurately in both cases.

Figure 7.6 shows a detailed comparison between STINet and MF-FRCNN against the ground-truth for trajectory prediction. Green boxes are the ground-truth boxes. Yellow, pink and cyan lines are the ground-truth future trajectory as well as the predicted future trajectories from MF-FRCNN and STINet respectively. For the left two pedestrians who are walking in a straight line, both MF-FRCNN and STINet predict future trajectory reasonably well but the MF-FRCNN still has a small error compared with the ground-truth; for the right-most pedestrian who is making a slight left turn, MF-FRCNN fails to capture the details of its movement and gives an unsatisfactory prediction, while STINet gives a much better trajectory prediction.

7.5 Conclusion and Future Works

In this chapter, we investigate object detection for autonomous driving, which involves temporal information heavily. The temporal information comes in two directions: 1) we have access to the past frames which provide more details and contexts; 2) we need to detect the objects in the future (*i.e.*, trajectory prediction), which requires stronger modeling of objects. To this end, we propose a novel two-stage end-to-end framework named STINet, to perform joint detection and trajectory prediction with raw lidar point clouds as the input. We propose to build temporal proposals with pedestrians' both current and past boxes and learn a rich representation for each temporal proposal, with local geometry, dynamic movement, history path and interaction features. This design allows us to make full use of the history context and model the objects with better details. We conduct comprehensive experiments on the Waymo Open dataset and the Lyft dataset, and demonstrate that by explicitly modeling the spatio-temporal-interaction features, both detection and trajectory prediction quality can be drastically improved compared with single-stage and two-stage baselines. This also makes us to re-think the importance of introducing second-stage and proposals in detection frameworks, especially for the joint detection and trajectory prediction task. The proposed method is also computationally efficient and achieves the real-time inference speed which makes our model practical to be used in real-world applications.

There are a few direction on which our method can be further improved. The first direction is to combine camera/map data which provide extra features/contexts to help understanding the scene. For example, camera inputs can provide detailed facial expressions and gestures which could indicate pedestrians' intent; and the map data can give contexts about where the pedestrians are, which can serve as priors of their possible actions. Another direction is to utilize longer history information with LSTMs, which can provide more context than the current 1-second history. Some actions and movements can take a longer time and the current setting may be ineffective in those cases. Finally, the computation cost may be improved by re-using the feature maps computed in the previous iterations. This enables us to model the objects with great detail and context with trivial computation overhead, as the history information and features are readily available from the previous iterations. This requires the network to be able to offset the ego-motion of the SDC and adjust the previous feature maps accordingly.

Chapter 8

Conclusion

In this dissertation, we focus on topics and projects regarding object detection, which is a fundamental task in computer vision. The challenges of this topic reside in many aspects. In Chapter 3 and Chapter 4 we discuss and propose novel network architectures to detection objects under occlusion and with various sizes. We demonstrate that accumulating mid-level visual cues in a Hough Voting manner can improve the robustness towards occlusion, and enriching the detection feature pyramid with high-level semantically strong segmentation features can help detecting objects with different scales especially for small objects. In Chapter 5 we design a novel training mechanism for long-tailed hard images. We propose to dynamically assign difficulty scores to training images and re-weight the images to address those hard ones on which the network does not work well. Both quantitative and qualitative experiments show our proposed method improves the robustness of hard cases, without introducing any computation overhead. In Chapter 6 and Chapter 7, we investigate how to modify detectors in order to take use of context information. For CT scan images with multiple slices, we propose a hybrid detector to combine 2D backbones and 3D context information for lesion detection and demonstrate large performance gain with comparable speed and GPU memory consumption. In the case of having sequential temporal inputs and making current and future predictions for pedestrians, we propose a spatial-temporal-interactive network with temporal proposals to effectively model moving pedestrians with possibly different speeds and patterns. Comprehensive modeling with local geometry, dynamic movement, history path and interaction features is possible and efficient with our proposed network. Both detection and trajectory prediction performance are improved comparing with strong baselines with the same inputs.

We demonstrate that, with carefully designed architectures and learning mechanisms, object detection generally works well on a board rage of applications with different challenges and opportunities. Further improvements could lie in many directions. Better backbone design with feature pyramids and feature fusions can provide stronger backbones for detection, and this can possibly be achieved via automatic architectures search over a large detection architecture search space. Combining inputs from different sensors is helpful to improve detection. For example, camera and LIDAR have different properties and capture complementary information, thus fusing them may lead to a more robust detection quality. Better training strategies can also benefit the robustness of detectors. Possible strategies include, but are not limit to, introducing uncertainty term in the loss, mining hard examples, overcoming category imbalance for infrequent ones, adaptively sampling positive and negative examples during the training, *etc.*

On the other hand, in addition to improving object detection with normal settings, object detections with scarce training data (few-shot learning), with image-level annotation (weakly supervised learning), with fine-grained annotation (instance segmentation) and with a mixture of labelled and unlabelled data (semi-supervised learning) are also important directions for future research, which have already attracted many research attentions. As in the real-world applications, the amount of data and the level of annotation can vary a lot, these directions make the object detection more versatile and applicable towards more real-world tasks.

References

[1] D. Marr, “Vision: A computational investigation into the human representation and processing of visual information, henry holt and co,” *Inc., New York, NY*, vol. 2, no. 4.2, 1982.

[2] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The kitti dataset,” *International Journal of Robotics Research (IJRR)*, 2013.

[3] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *European Conference on Computer Vision*, Springer, 2014, pp. 740–755.

[4] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet, *Lyft level 5 av dataset 2019*, urlhttps://level5.lyft.com/dataset/, 2019.

[5] S. Yang, P. Luo, C.-C. Loy, and X. Tang, “Wider face: A face detection benchmark,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5525–5533.

[6] J. Wang, C. Xie, Z. Zhang, J. Zhu, L. Xie, and A. Yuille, “Detecting semantic parts on partially occluded objects,” *arXiv preprint arXiv:1707.07819*, 2017.

[7] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (voc) challenge,” *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[8] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[9] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

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

[12] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[13] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[14] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[15] K. Yan, X. Wang, L. Lu, L. Zhang, A. P. Harrison, M. Bagheri, and R. M. Summers, "Deep lesion graphs in the wild: Relationship learning and organization of significant radiology image findings in a diverse large-scale lesion database," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9261–9270.

[16] *Waymo open dataset: An autonomous driving dataset*, 2019.

[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European Conference on Computer Vision*, 2016, pp. 21–37.

[18] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Object detectors emerge in deep scene cnns," *arXiv preprint arXiv:1412.6856*, 2014.

[19] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[20] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1440–1448.

[21] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[22] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 734–750.

[23] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1627–1645, 2009.

[24] S. Fidler, R. Mottaghi, A. Yuille, and R. Urtasun, "Bottom-up segmentation for top-down detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3294–3301.

[25] X. Wang, M. Yang, S. Zhu, and Y. Lin, "Regionlets for generic object detection," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 17–24.

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

[27] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, “Overfeat: Integrated recognition, localization and detection using convolutional networks,” *arXiv preprint arXiv:1312.6229*, 2013.

[28] J. Dai, Y. Li, K. He, and J. Sun, “R-fcn: Object detection via region-based fully convolutional networks,” in *Advances in Neural Information Processing Systems*, 2016, pp. 379–387.

[29] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, “Selective search for object recognition,” *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[30] C. L. Zitnick and P. Dollár, “Edge boxes: Locating object proposals from edges,” in *European Conference on Computer Vision*, Springer, 2014, pp. 391–405.

[31] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, “Centernet: Keypoint triplets for object detection,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 6569–6578.

[32] M. Najibi, P. Samangouei, R. Chellappa, and L. S. Davis, “Ssh: Single stage headless face detector,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4875–4884.

[33] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li, “S3fd: Single shot scale-invariant face detector,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 192–201.

[34] X. Tang, D. K. Du, Z. He, and J. Liu, “Pyramidbox: A context-assisted single shot face detector,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 797–813.

[35] C. Chi, S. Zhang, J. Xing, Z. Lei, S. Z. Li, and X. Zou, “Selective refinement network for high performance face detection,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 8231–8238.

[36] K. Yan, M. Bagheri, and R. M. Summers, “3d context enhanced region-based convolutional neural network for end-to-end lesion detection,” in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer, 2018, pp. 511–519.

[37] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3d classification and segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[38] C. R. Qi, O. Litany, K. He, and L. J. Guibas, “Deep hough voting for 3d object detection in point clouds,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 9277–9286.

[39] C. R. Qi, X. Chen, O. Litany, and L. J. Guibas, “Imvotenet: Boosting 3d object detection in point clouds with image votes,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4404–4413.

[40] J. Ngiam, B. Caine, W. Han, B. Yang, Y. Chai, P. Sun, Y. Zhou, X. Yi, O. Alsharif, P. Nguyen, *et al.*, “Starnet: Targeted computation for object detection in point clouds,” *arXiv preprint arXiv:1908.11069*, 2019.

[41] Y. Zhou and O. Tuzel, “Voxelnet: End-to-end learning for point cloud based 3d object detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4490–4499.

[42] Y. Yan, Y. Mao, and B. Li, “Second: Sparsely embedded convolutional detection,” *Sensors*, vol. 18, no. 10, p. 3337, 2018.

[43] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, “Pointpillars: Fast encoders for object detection from point clouds,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 697–12 705.

[44] Y. Zhou, P. Sun, Y. Zhang, D. Anguelov, J. Gao, T. Ouyang, J. Guo, J. Ngiam, and V. Vasudevan, “End-to-end multi-view fusion for 3d object detection in lidar point clouds,” *arXiv preprint arXiv:1910.06528*, 2019.

[45] W. Luo, B. Yang, and R. Urtasun, “Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 3569–3577.

[46] S. Casas, W. Luo, and R. Urtasun, “Intentnet: Learning to predict intention from raw sensor data,” in *Conference on Robot Learning*, 2018, pp. 947–956.

[47] J. Wang, Z. Zhang, C. Xie, Y. Zhou, V. Premachandran, J. Zhu, L. Xie, and A. Yuille, “Visual concepts and compositional voting,” *arXiv preprint arXiv:1711.04451*, 2017.

[48] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *International Conference on Learning Representations*, 2015.

[49] S. Qiao, W. Shen, W. Qiu, C. Liu, and A. Yuille, “Scalenet: Guiding object proposal generation in supermarkets and beyond,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1791–1800.

[50] X. Chen, R. Mottaghi, X. Liu, S. Fidler, R. Urtasun, and A. Yuille, “Detect what you can: Detecting and representing objects using holistic models and body parts,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[51] J. Zhu, X. Chen, and A. L. Yuille, “Deepm: A deep part-based model for object detection and semantic part localization,” *arXiv preprint arXiv:1511.07131*, 2015.

[52] N. Zhang, J. Donahue, R. Girshick, and T. Darrell, “Part-based r-cnns for fine-grained category detection,” in *European Conference on Computer Vision*, Springer, 2014.

[53] H. Zhang, T. Xu, M. Elhoseiny, X. Huang, S. Zhang, A. Elgammal, and D. Metaxas, “Spda-cnn: Unifying semantic part detection and abstraction for fine-grained recognition,” in *Computer Vision and Pattern Recognition*, IEEE, 2016.

[54] S. Huang, Z. Xu, D. Tao, and Y. Zhang, “Part-stacked cnn for fine-grained visual categorization,” in *Computer Vision and Pattern Recognition*, IEEE, 2016.

[55] G. Gkioxari, R. Girshick, and J. Malik, “Actions and attributes from wholes and parts,” in *International Conference on Computer Vision*, IEEE, 2015.

[56] D. Novotný, D. Larlus, and A. Vedaldi, “I have seen enough: Transferring parts across categories,” in *British Machine Vision Conference*, 2016.

[57] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *International Conference on Machine Learning*, 2010.

[58] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting.," *Journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[59] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *International Conference on 3D Vision*, IEEE, 2016.

[60] A. Kortylewski, Q. Liu, H. Wang, Z. Zhang, and A. Yuille, "Combining compositional models and deep networks for robust object classification under occlusion," in *The IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 1333–1341.

[61] Y. Wang, L. Xie, C. Liu, S. Qiao, Y. Zhang, W. Zhang, Q. Tian, and A. Yuille, "Sort: Second-order response transform for visual recognition," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017.

[62] P. Tang, X. Wang, X. Bai, and W. Liu, "Multiple instance detection network with online instance classifier refinement," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2843–2851.

[63] G. Papandreou, L.-C. Chen, K. P. Murphy, and A. L. Yuille, "Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1742–1750.

[64] S. Gidaris and N. Komodakis, "Object detection via a multi-region and semantic segmentation-aware cnn model," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1134–1142.

[65] A. Shrivastava and A. Gupta, "Contextual priming and feedback for faster r-cnn," in *European Conference on Computer Vision*, 2016, pp. 330–348.

[66] T. Kong, F. Sun, A. Yao, H. Liu, M. Lu, and Y. Chen, "Ron: Reverse connection with objectness prior networks for object detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 2017, p. 2.

[67] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," *arXiv preprint arXiv:1709.01507*, 2017.

[68] S. Bell, C. Lawrence Zitnick, K. Bala, and R. Girshick, "Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2874–2883.

[69] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[70] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "Dssd: Deconvolutional single shot detector," *arXiv preprint arXiv:1701.06659*, 2017.

[71] M. M. Derakhshani, S. Masoudnia, A. H. Shaker, O. Mersa, M. A. Sadeghi, M. Rastegari, and B. N. Araabi, "Assisted excitation of activations: A learning technique to improve object detectors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9201–9210.

[72] J. Cao, Y. Pang, and X. Li, "Triply supervised decoder networks for joint detection and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7392–7401.

[73] V. Jain and E. Learned-Miller, "Fddb: A benchmark for face detection in unconstrained settings," Tech. Rep.

[74] J. Yan, X. Zhang, Z. Lei, and S. Z. Li, "Face detection by structural models," *Image and Vision Computing*, vol. 32, no. 10, pp. 790–799, 2014.

[75] X. Zhu and D. Ramanan, "Face detection, pose estimation, and landmark localization in the wild," in *2012 IEEE conference on computer vision and pattern recognition*, IEEE, 2012, pp. 2879–2886.

[76] X. Cao, Y. Wei, F. Wen, and J. Sun, "Face alignment by explicit shape regression," *International Journal of Computer Vision*, vol. 107, no. 2, pp. 177–190, 2014.

[77] X. Xiong and F. De la Torre, "Supervised descent method and its applications to face alignment," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 532–539.

[78] M. Kim, S. Kumar, V. Pavlovic, and H. Rowley, "Face tracking and recognition with visual constraints in real-world videos," in *CVPR*, 2008, pp. 1–8.

[79] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *BMVC*, 2015, pp. 41.1–41.12.

[80] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *CVPR*, 2015, pp. 815–823.

[81] P. Hu and D. Ramanan, "Finding tiny faces," in *CVPR*, 2017, pp. 1522–1530.

[82] X. Tang, D. K. Du, Z. He, and J. Liu, "Pyramidbox: A context-assisted single shot face detector," in *ECCV*, 2018, pp. 812–828.

[83] J. Yu, Y. Jiang, Z. Wang, Z. Cao, and T. Huang, "Unitbox: An advanced object detection network," in *ACMMM*, 2016, pp. 516–520.

[84] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li, "S$\hat{3}$fd: Single shot scale-invariant face detector," in *ICCV*, 2017, pp. 192–201.

[85] C. Zhu, R. Tao, K. Luu, and M. Savvides, "Seeing small faces from robust anchor's perspective," in *CVPR*, 2018, pp. 5127–5136.

[86] Y. Bai, Y. Zhang, M. Ding, and B. Ghanem, "Finding tiny faces in the wild with generative adversarial network," in *CVPR*, 2018, pp. 21–30.

[87] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *CVPR*, 2016, pp. 761–769.

[88] I. Loshchilov and F. Hutter, "Online batch selection for faster training of neural networks," *arXiv preprint arXiv:1511.06343*, 2015.

[89] T.-Y. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *ICCV*, 2017, pp. 2999–3007.

[90] H. A. Rowley, S. Baluja, and T. Kanade, “Neural network-based face detection,” *PAMI*, pp. 23–38, 1998.

[91] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *CVPR*, 2001, pp. 511–518.

[92] B. Singh and L. S. Davis, “An analysis of scale invariance in object detection–snip,” in *CVPR*, 2018, pp. 3578–3587.

[93] V. Jain and E. Learned-Miller, “Fddb: A benchmark for face detection in unconstrained settings,” Tech. Rep., 2010.

[94] N. Qian, “On the momentum term in gradient descent learning algorithms,” *Neural networks*, pp. 145–151, 1999.

[95] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, “A unified multi-scale deep convolutional neural network for fast object detection,” in *ECCV*, 2016, pp. 354–370.

[96] E. Ohn-Bar and M. M. Trivedi, “To boost or not to boost? on the limits of boosted trees for object detection,” in *ICPR*, 2016.

[97] H. Wang, Z. Li, X. Ji, and Y. Wang, “Face r-cnn,” *arXiv preprint arXiv:1706.01061*, 2017.

[98] J. Wang, Y. Yuan, and G. Yu, “Face attention network: An effective face detector for the occluded faces,” *arXiv preprint arXiv:1711.07246*, 2017.

[99] Y. Wang, X. Ji, Z. Zhou, H. Wang, and Z. Li, “Detecting faces using region-based fully convolutional networks,” *arXiv preprint arXiv:1709.05256*, 2017.

[100] B. Yang, J. Yan, Z. Lei, and S. Z. Li, “Aggregate channel features for multi-view face detection,” in *IJCB*, 2014, pp. 1–8.

[101] S. Yang, P. Luo, C.-C. Loy, and X. Tang, “From facial parts responses to face detection: A deep learning approach,” in *ICCV*, 2015, pp. 3676–3684.

[102] S. Yang, Y. Xiong, C. C. Loy, and X. Tang, “Face detection through scale-friendly deep convolutional networks,” *arXiv preprint arXiv:1706.02863*, 2017.

[103] C. Zhang, X. Xu, and D. Tu, “Face detection using improved faster rcnn,” *arXiv preprint arXiv:1802.02142*, 2018.

[104] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, “Joint face detection and alignment using multitask cascaded convolutional networks,” *IEEE Signal Processing Letters*, pp. 1499–1503, 2016.

[105] C. Zhu, Y. Zheng, K. Luu, and M. Savvides, “Cms-rcnn: Contextual multi-scale region-based cnn for unconstrained face detection,” in *Deep Learning for Biometrics*, 2017, pp. 57–79.

[106] J. Li and Y. Zhang, “Learning surf cascade for fast and accurate object detection,” in *CVPR*, 2013, pp. 3468–3475.

[107] M. Mathias, R. Benenson, M. Pedersoli, and L. Van Gool, “Face detection without bells and whistles,” in *ECCV*, 2014, pp. 720–735.

[108] X. Shen, Z. Lin, J. Brandt, and Y. Wu, “Detecting and aligning faces by image retrieval,” in *CVPR*, 2013, pp. 3460–3467.

[109] P. Viola and M. J. Jones, “Robust real-time face detection,” *IJCV*, pp. 137–154, 2004.

[110] Y. Bai, Y. Zhang, M. Ding, and B. Ghanem, “Finding tiny faces in the wild with generative adversarial network,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 21–30.

[111] J. Ding, A. Li, Z. Hu, and L. Wang, “Accurate pulmonary nodule detection in computed tomography images using deep convolutional neural networks,” in *MICCAI*, Springer, 2017, pp. 559–567.

[112] F. Liao, M. Liang, Z. Li, X. Hu, and S. Song, “Evaluate the malignancy of pulmonary nodules using the 3d deep leaky noisy-or network,” *arXiv preprint arXiv:1711.08324*, 2017.

[113] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, “Tensorflow: A system for large-scale machine learning,” in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283.

[114] A. A. A. Setio, A. Traverso, T. De Bel, M. S. Berens, C. van den Bogaard, P. Cerello, H. Chen, Q. Dou, M. E. Fantacci, B. Geurts, *et al.*, “Validation, comparison, and combination of algorithms for automatic detection of pulmonary nodules in computed tomography images: The luna16 challenge,” *Medical image analysis*, vol. 42, pp. 1–13, 2017.

[115] Z. Zhu, C. Liu, D. Yang, A. Yuille, and D. Xu, “V-nas: Neural architecture search for volumetric medical image segmentation,” in *2019 International Conference on 3D Vision (3DV)*, IEEE, 2019, pp. 240–248.

[116] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, “Social gan: Socially acceptable trajectories with generative adversarial networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2255–2264.

[117] A. Milan, S. H. Rezatofighi, A. Dick, I. Reid, and K. Schindler, “Online multi-target tracking using recurrent neural networks,” in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[118] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, “Social lstm: Human trajectory prediction in crowded spaces,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 961–971.

[119] J. Hong, B. Sapp, and J. Philbin, “Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8454–8462.

[120] *Waymo open dataset: An autonomous driving dataset*, 2019.

[121] R. Hou, C. Chen, and M. Shah, “Tube convolutional neural network (t-cnn) for action detection in videos,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5822–5831.

[122] V. Kalogeiton, P. Weinzaepfel, V. Ferrari, and C. Schmid, “Action tubelet detector for spatio-temporal action localization,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4405–4413.

[123] P. Tang, C. Wang, X. Wang, W. Liu, W. Zeng, and J. Wang, "Object detection in videos by high quality object linking," *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[124] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

[125] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap, "A simple neural network module for relational reasoning," in *Advances in neural information processing systems*, 2017, pp. 4967–4976.

[126] X. Wang and A. Gupta, "Videos as space-time region graphs," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 399–417.

[127] C. Sun, A. Shrivastava, C. Vondrick, R. Sukthankar, K. Murphy, and C. Schmid, "Relational action forecasting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 273–283.

[128] J. Yang, J. Lu, S. Lee, D. Batra, and D. Parikh, "Graph r-cnn for scene graph generation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 670–685.

[129] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, *et al.*, "Argoverse: 3d tracking and forecasting with rich maps," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8748–8757.

[130] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker, "Desire: Distant future prediction in dynamic scenes with interacting agents," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 336–345.

[131] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, Springer, 2015, pp. 234–241.

Vita

Zhishuai Zhang is completing his Ph.D. degree of Computer Science at the Johns Hopkins University, under the supervision of Bloomberg Distinguished Professor Alan L. Yuille. He has been working on computer vision, machine learning and object recognition during his Ph.D. study. Before that, he obtained the Bachelor of Engineering degree of Computer Science and Engineering from the University of Science and Technology of China in 2016. During his undergraduate study, he became interested in computer vision and visited the Johns Hopkins University to get involved in research on that area. As a Ph.D. student, he mainly works on object recognition and detection on large-scale datasets with deep learning based methods. Particularly, he is interested in how to detect object robustly, with possible challenges like occlusion or scale variance, and incorporating 3D or temporal contexts. He also worked on adversarial attack and defense, content recommendation and personalization. He interned in Facebook and Waymo and obtained industrial experience and the knowledge of combining academic works with industry problems and requirements.