TEIKYO UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

DOCTORAL THESIS

A Framework for Object Detection using Deep Learning

Author:
Tran Duy LINH

Supervisor:
Masayuki ARAI

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Arai Laboratory
Division of Integrated Science and Engineering

February, 2019

Declaration of Authorship

I, Tran Duy LINH, declare that this thesis titled, "A Framework for Object Detection using Deep Learning" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"If you wish to make apple pie from scratch, you must first create the universe."

Carl Sagan

Abstract

Doctor of Philosophy

A Framework for Object Detection using Deep Learning

by Tran Duy LINH

In the present thesis, our aim is to make an object detection framework on images using deep learning. Object detection is one of the most challenging task in computer vision because the detector needs to output not only the location of objects but also predict the object label and/or perform the object segmentation. We focus on developing robust object detection methods that reduce the detection error rate through a two-stage deep network. Since the detection method relied on sliding window scheme, the outputs cover a number of false positive detections and often have low performance on small scale or low resolution. To address these problems, we propose a method that uses a secondary network as classifier to classify these detections again. This approach allows us to perform high level inferences including re-scoring detection confidence and label choosing thus making the better detection results.

For the testing, we perform our approach on several different kinds of objects. We focus on testing key applications of object detection such as pedestrian detection and detecting multi-object in nature scenes. We compare the results with cutting-edge object detection methods. The experimental results show the effectiveness of the proposed method. Some results achieve the state-of-the-art performances for difficult conditions such as small scales and low resolution.

The present thesis makes four main contributions. Firstly, we introduce the two-stage deep neural model for object detection in images. The two-stage model uses a proposal network to extract the set of object candidates which are the proposal bounding boxes, the classes and the scores of objects available in the image. The second stage uses a deep neural network to re-classify proposal detections, then we use a combination function to combine the scores of the first stage and the second stage. We show that the connection between two scores makes the better performance. Secondly, we explore the role of combination function in boosting the performance of both classification network and overall detection model. Thirdly, we show the effectiveness of using two-stage model for small object detection. Lastly, we present the idea to apply the new design for the object detection by segmentation problem.

Acknowledgements

Undertake this Ph.D has been a my life-challenging. I am grateful and thankful to all the people who have give me their support and guidance for completion of my Ph.D study.

Firstly, I would like to express my sincere gratitude to my advisor Prof. Arai Masayuki for the continuous support of my Ph.D study and related research. I feel he has been an ideal advisor. It all started since my first day in graduate school when he offered me a great opportunity to join his pattern recognition lab. Under his supervision, I learned how to define a research problem, review of literature, prepare research design, analyze the data and publish the results. Thank you for his motivation, patience and immense knowledge. He helped me in all the time of research and writing of this thesis. Without his great support, this work would be put off to the next century.

Besides my advisor, I would like to thank the rest of my thesis committee members: Prof. Akimoto Kazuhiro, Prof. Watanabe Hiroyoshi, Dr. Kobayashi Yasuyuki, Dr. Mizutani Kozo and Dr. Yamane Ken for their great support and invaluable advice which encourage me to widen my research from various perspectives. I also show gratitude for Dr. Kondo Naoki, he is an excellent teacher who taught me math and shared his tremendous knowledge in topology geometry as well as life experiences.

I would like to thank all my friends and colleges for their continued support and making the Teikyo University a fun and interesting place to work. I can never forget my friends: Kenneth, Hien, Sang, Hai for introduced me to the Japanese culture and language, even though they are non-Japaneses.

I gratefully acknowledge the funding received towards my Ph.D from Teikyo University. Thanks to Prof. Hirako for his encouragement and valuable advices. Many thanks also to Mrs. Tanaka for her unconditional support even during vacations, her Japanese class was one of the most exciting experiences I had while at TU. I am also very grateful to all those at the Student Support team, especially Ms. Ōkubo Takako and others who were always so helpful and provided me with their assistance throughout my studies at TU.

Last but not least, I would like to express my deepest gratitude to my Parents, my Sisters and my Uncle for their absolute confidence in me. This thesis would not have been possible without their warm love, continued patience, and endless support.

Contents

List of Figures

List of Tables

List of Abbreviations

AI	**A**rtificial **I**ntelligence
AUC	**A**rea **U**nder the **C**urve
CCF	**C**onvolutional **C**hannel **F**eatures
CNN	**C**onvolutional **N**eural **N**etwork
DNN	**D**eep **N**eural **N**etwork
DoG	**D**ifference **o**f **G**aussian
DPM	**D**eformable **P**art **M**odel
DSSD	**D**econvolutional **S**ingle **S**hot **D**etector
FPPI	**F**alse **P**ositives **P**er **I**mage
FPR	**F**alse **P**ositive **R**ate
HOG	**H**istograms of **O**riented **G**radients
ICF	**I**ntegral **C**hannel **F**eatures
ILSVRC	**I**mageNet **L**arge **S**cale **V**isual **R**ecognition **C**hallenge
IoU	**I**ntersection **o**ver **U**nion
mAP	**m**ean **A**verage **P**recision
NMS	**N**on **M**aximum **S**uppression
Pascal VOC	**Pascal V**isual **O**bject **C**lasses
R-CNN	**R**egional **CNN**
R-FCN	**R**egion-based **F**ully **C**onvolutional **N**etworks
ROC	**R**eceiver **O**perating **C**haracteristic
RoI	**R**egion **o**f **I**nterest
RPN	**R**egion **P**roposal **N**etwork
SGD	**S**tochastic **G**radient **D**escent
SVM	**S**upport **V**ector **M**achine
TPR	**T**rue **P**ositive **R**ate
YOLO	**Y**ou **O**nly **L**ook **O**nce

This work is dedicated to my beloved parents

Chapter 1

Introduction

Computer vision has a long history in computer science. The goal of computer vision is to extract useful information from images or videos. Although various applications of computer vision are now in daily use, it remains a challenging task. The computer's performance is still well below a human's performance.

A key problem is how to develop a high-level understanding of the intricate structure of large datasets. Many computer vision algorithms involve pre-programming to solve a particular problem without pre-training the data. This approach has been used across many fundamental computer vision problems, but is limited due to the complexity of image data. A data-driven approach has been proposed to address this problem. In particular, the computer vision method uses training data to try to predict the real world data. We were able to rapidly improve performance by building better datasets and proposing better machine learning algorithms.

To obtain high-level information from images, object classification and detection are vital computer vision problems. These tasks would have many applications, such as in robotics, auto manufacturing, auto driving, and human computer interaction. However, they remain challenging because of a number of object properties, such as viewpoint variance, scale variance, deformation, occlusion, illumination, and background clutter. Thus, a model would need invariance to these variations while retaining sensitivity to inter-class variations. Object detection is also required to localize the object in the image.

This chapter is organized as follows. In Section 1.1, we present the purpose of the thesis. Section 1.2 describes applications of object detection. Section 1.3 discusses why this task is so challenging. Section 1.4 introduces some approaches to resolve the object detection problem. Section 1.5 presents an overview of our proposed method. Section 1.6 details the contributions of this thesis.

1.1 The purpose of this research

In the present thesis, we propose a deep neural model for object detection in images. In particular, we build a two-stage network to predict object location and class in still images. The object categories vary from single category (human) to multiple categories. For a specific problem, we propose several

model designs and use multiple evaluation methods. Our models aim to improve object detection performance. In general, a detector performs two key operations:

- Extracting feature maps from an input image, and
- Classification/regression from those feature maps to output detections.

Recent state-of-the-art object detectors rely on the box-proposal-based method, in which the image is processed through several steps: fixed sliding windows extraction, class-agnostic box prediction, box refinement and class prediction. Our main contribution to design a framework that includes more processing steps, such as re-classifying object proposals by an additional network, to achieve higher inferences from the output of these steps to improve overall performance. However, adding more processing steps leads to an increase in model complexity and optimization difficulty. For example, given an image with the class *cat*, the first model may return the location and the correct class while the second model may return another class, such as *dog*. In this scenario, we do not know which model is correct, so the relationship between the two models impacts the final detection performance. Our overarching model should predict better detections based on the agreement between the first model and the second model. The mismatch of predicted classes requires sophisticated training and decision-making. The purpose is to make a connection between the confidence levels of the first model and the second model. This thesis focuses on general object detectors that are not strong-classification-type. We discuss why this task is so challenging in Section 1.3.

1.2 Application of object detection

Although object detection has formed over five decades, it remains a very challenging task. It has occupied thousands of scientists and countless engineers, but we are still far from achieving computer performance equal to human performance. Despite this, object detection is an attractive and active research task because of its important applications. Our first consideration is the appearance of people in images. "Human" is the most important object and may be the most frequently captured object in images and videos. For example, an autonomous car uses pedestrian detectors to detect the surrounding people in order to improve safety. Auto-driving systems typically include camera, radar, laser light, Global Positioning System (GPS), and computers to control the vehicles and travel between destinations without the need for human input. In this case, via one or multiple cameras, the detectors "view" the surrounding world and identify the locations of people, such as pedestrians walking on the sidewalk or crosswalk, or cyclists riding bicycles. Such detectors can be applied for any level of automated driving system, from driver assistance to full automation.

Figure 1.1 displays examples from the Caltech pedestrian dataset (Dollar et al., 2012). The images show pedestrians captured by camera, and were

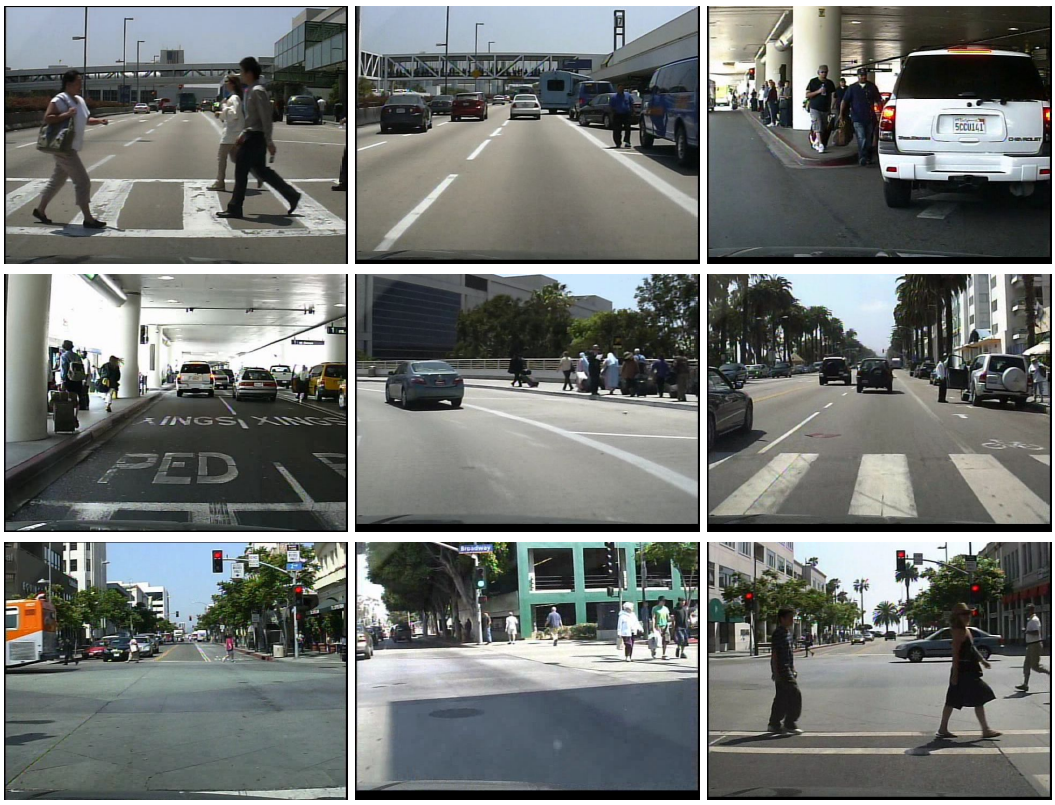


FIGURE 1.1: Pedestrians in an urban scene. These images are part of the Caltech pedestrian dataset, which was captured in an urban environment. People appear alongside other vehicles with low resolution.

collected from a vehicle driving through regular urban traffic. The detectors should be able to recognize people alongside other objects, such as cars, trees, buildings, and traffic signs.

Another application of people detection is video surveillance systems. With the increasing number of cameras in public places such as streets, shopping malls, and train stations, manual observation of video streams is economically infeasible. Thus, there is a crucial role for intelligent surveillance systems that can continuously analyze video streams. In terms of e-health applications, a monitoring system detecting the falls of elderly people would reduce health risks and support elderly people to live longer in their home environment.

Object detection also plays important roles in intelligent digital content management software. In this application, the program automatically adds tags to images, making the user image library searchable and displaying relevant content to users. Figure 1.2 shows examples of objects captured in their natural context. The number of recent worldwide captured images is about one trillion per year[1]. Thus, each person captures an average of one thousand images in 2-3 years. Most images are casual snapshots and it is difficult to organize these huge numbers of images. While devices might already tag images with date and location, intuitive information, such as objects or names

[1]http://blog.infotrends.com/how-long-does-it-take-to-shoot-1-trillion-photos/

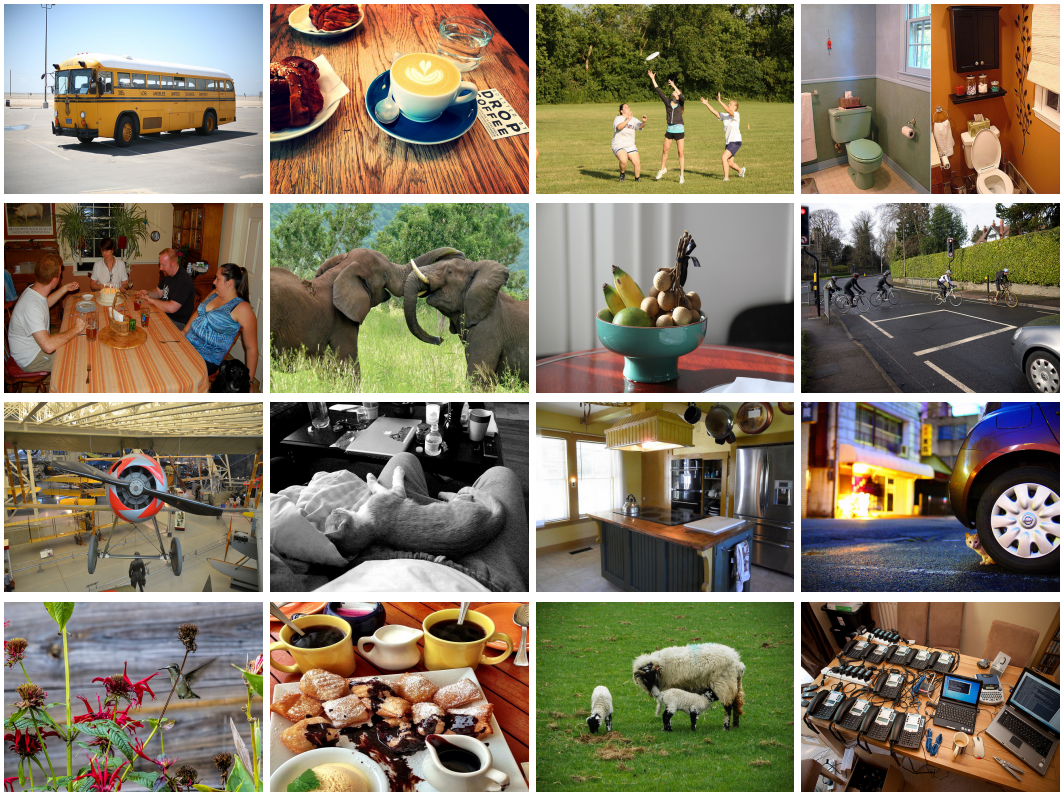


FIGURE 1.2: A collection of multiple categories of objects in their natural context. These images are part of the Common Objects in COntext (COCO) object detection dataset, which is used for evaluating some models in this thesis. The categories cover various sizes, colors, backgrounds, and illuminations.

of people in the image, would support understanding of image contents and associated tools.

1.3 Why this is challenging

Object detection is challenging because the detector needs to simultaneously detect and classify multiple objects. Unlike object classification, the output of object detection is variable in length, and the number of objects may change from image to image. Thus, the biggest difficulty in object detection is the amount of variations in images. We consider the following specific variation types:

- **Viewpoint variation**: since the object is captured in many ways with respect to the camera, a single instance of an object can appear very differently.
- **Scale variation**: the variation in scales occurs not only in the real world (the different sizes of objects within same class), but also in the image. In the image, object size is often defined by the size of the object's bounding box, and the detector needs to handle this large variation.

- **Intra-class variation**: this variation is very common because of the different types of interest categories. For example, consider a "potted plant" object; there are no regulations for the flowerpots nor the plants. The impact of this variation type depends on the object class.

- **Deformation**: many objects can be deformed in various ways. This type of variation often appears in animals. For example, human appearance and pose both change between images.

- **Occlusion**: the occlusion level is defined by the proportion of visible parts. One object detection problem is how to predict the location of the occluded parts of an object. Although the object class can be predicted by some visible object parts, the whole object location is quite difficult to predict. In some cases, the detector needs to predict the whole object position even if the proportion of visible parts is very small.

- **Illumination conditions**: an object's appearance is also affected by illumination. For example, an object will look different in sunlight, in shadow during the day, in dim light at night or under artificial light. Although these conditions seem trivial with the human visual system, the change in illumination causes huge changes at the pixel level, which has a strong impact on detection performance.

- **Background clutter**: this variation is common in natural-context images where the same object may be captured against different backgrounds. For example, images taken in outdoor or indoor environments will have completely distinct backgrounds. In some cases, the object and the background may be very similar, so a robust detector would need to distinguish these changes in background.

Since the above challenges in object detection are not independent (in fact, they often appear together), the detector needs to address them simultaneously, which adds to the complexity of object detection. Moreover, some of these challenges conflict. For example, a detector that handles large intra-class variation may output many false positives (incorrect detections) over a complex background.

In Figure 1.1, people appear with different poses, clothing, and sizes, with variation of illumination conditions and different backgrounds, highlighting difficulties in people detection. Other challenges include the imbalance between positive images and negative images. More than 50% of the images collected contain no pedestrians. The number of pedestrians in positive images is also very low. Training the detector on this dataset thus leads to instability and inaccuracy because conventional detectors are usually designed to improve accuracy by reducing errors. This problem is crucial in some applications, such as safety driving systems, which require the detection of rare events (pedestrian appearances).

Figure 1.2 provides examples of challenges for general object detection using the Common Objects in COntext (COCO) dataset (Lin et al., 2014). In

FIGURE 1.3: Objects with high-level context and background. Some objects can only be detected using image context and more sophisticated techniques, such as high-level inference, to make correct decisions about ambiguous object instances.

these images, apart from the categories of interest, there are many other objects. Some different objects are also very similar. Thus, the challenge is that an image contains multiple objects. The identity of many objects can only be resolved using context due to the small size or ambiguous appearance in the image. The COCO dataset contains more object instances per image (7.7) as compared to the Caltech pedestrian dataset (1.5). Thus, detectors are also required to perform precise object localization.

Figure 1.3 shows object detection in images with high-level background information. To identify false detections, the detector needs to process not only the object's sub-window, but also develop a high-level understanding of the surrounding background. In the left image, by looking at the background (the *bus*), humans can easily eliminate the false detection case because it is clear that the people posing on the side of the bus are not real. Similarly, in the right image, the surrounding context (the *book with text*) indicates that the apparent objects inside the book are not real objects. However, to encode this type of information in the detector is quite difficult. The detector needs to use context information and high-level inference methods to determine correct objects.

Another challenge in object detection is small object detection where objects appear in a small size because of the camera angle and focal length (we only consider small objects by their appearance in an image, not in proportion to the real world). Usually, small objects have low resolution and appear near large objects. The detector tends to focus on large objects and ignore small ones. Thus, the detector needs to handle insufficient resolution instances to detect small objects.

1.4 Object detection background

As mentioned above, the object detection problem has a long history of research, and the detection method involves two main operations: feature extraction and the detection algorithm. The feature extractor transforms the input data into a reduced set of features (or a feature vector). The selected subsets of features are expected to contain the relevant information from the

input data. Then, the detection algorithm uses these selected features to perform the detection.

There is no standard definition of what constitutes features; the features depend on the task at hand. However, it is expected that the feature extractor has the property of being repeatable. Many computer vision algorithms use the feature as a starting point for subsequent algorithms. Many feature extractors have been developed, and they can be divided into two types: handcrafted feature extractors and learned feature extractors.

- **Handcrafted feature extractor**: common examples include those based on the histogram of oriented gradients (HOG) (Dalal and Triggs, 2005a) and SIFT (Lowe, 1999). These methods extract the feature from an image using a predefined algorithm created from human knowledge. The feature extractor can be a very basic algorithm (e.g., edges and corners) and is often inspired by the domain and the particular application.

- **Learned feature extractor**: this type of feature extractor often refers to learned features obtained by training a deep learning network. The deep learning method uses a backpropagation algorithm to update its internal parameters that are used to compute the image representation (the feature). Feature learning can be divided into unsupervised and supervised. For example, AlexNet (Krizhevsky, Sutskever, and Hinton, 2012) is a notable work from the beginning of the artificial intelligence (AI) era. This breakthrough research uses a convolutional neural network (CNN) to drastically reduce the error rate of object recognition, and it was a key catalyst of the subsequent quick adoption of deep learning by the computer vision community.

Compared to handcrafted feature extractors, learned feature extractors have dramatically improved the state-of-the-art visual object recognition, object detection, and many other tasks. Note that the difference between the two types is the approach to obtaining the features. Handcrafted feature extraction does not depend on the data, so the quality of extracted features relies on the algorithm itself. This approach can deal with small-size homogeneous datasets and often runs fast. In contrast, learned feature extraction depends on the dataset. It tends to perform well with heterogeneous and unfamiliar datasets. The quality of the dataset and the learning algorithm strongly impact the quality of extracted features.

The other component of the object detector is the detection algorithm. Numerous models have been studied. The initial idea is sliding window classification, in which the detector tests each sub-window. The sub-window can then be further processed to match with actual object locations. Detection algorithms can be divided into two main approaches:

- **The part-based method** (Felzenszwalb et al., 2010): the object is represented as a set of parts constrained in a possible spatial arrangement. For example, a human can be modeled as a head, two legs, two arms, a torso and the connections of those parts. In this approach, the model uses a star-structure by defining "root" filters plus a set of part filters

and a deformation model. The model then trains the bounding boxes based on the set of part bounding boxes.

- **Region proposals approach** (Girshick et al., 2014): this simultaneously predicts the location and class. Instead of using the sliding window, which is weak for localization, this method uses a set of object candidates to classify the object classes. It then performs regression on the boxes to forecast the offset between predicted boxes and ground-truth boxes. This method is usually based on the rich features extracted by the learned feature extractor.

Both of these approaches rely on the image feature maps and reduce the object detection problem to binary object classification problems with bounding box regression. Other approaches, such as integral channel features (ICF) (Dollár et al., 2009), use multiple registered image channels and the direct sub-window location with smaller step size than HOG to output object detections. The details of these methods will be discussed in Chapter 2.

1.5 Overview of our method

Our approach to object detection focuses on reducing errors (specifically, false positive detections) and improving feature extraction.

First, we adopt the region proposal method to extract the set of object candidates (the object proposals). However, because of the sliding window scheme, in which a rectangular region of fixed width and height "slides" across an image, the output of the detector contains many detection errors. There are two types of errors in the output object detection: poor localization and misclassification. For instance, in the Single Shot Detector (SSD) (Liu et al., 2016) method, more than half of false positive detections are due to misclassification. Misclassification detection occurs because of ambiguousness with background objects, similar objects, or dissimilar objects. This issue motivates us to use an additional classifier to reduce the number of misclassification detections. The results of the first-stage network (the proposal network) and the second-stage network (the classification network) are used to perform the final detection. To provide the final output, a single network output of the two networks is not robust; they must be combined for the final results. We found that aggregation between the first-stage network score and the second-stage score improved the overall performance of the network.

Second, we provide an intensive method to combine the two networks by using the output of the first-stage network as additional input for the second-stage network. We call the new model trained combination, and it is able to discern where the first model performs well and where it performs poorly. Thus, the new trained combination model achieves higher performance than single-stage detectors. Our experiment shows that the new classification model only adds a small computational cost while increasing the speed of convergence.

Third, we adopt a simple architecture (Mask R-CNN) (He et al., 2017), and re-design the feature extraction module for object detection by segmentation

(instance segmentation). We concentrate on one object detection component: the feature pooling module. Our approach is to consider it as a subnetwork. Our pooling module is a quantization-free layer, multi-scale ensemble that preserves exact spatial locations between the Region of Interest (RoI) and the pooled feature map. Although the proposed approach is simple instance segmentation, we show that with appropriate network configuration, it can provide state-of-the-art results.

1.6 Contributions

The main contributions of this thesis are:

- **A proposed hierarchical two-stage network to improve object detection performance**. The image is processed through a proposal network and then a classification network. Based on our study, the two-stage network significantly improves the detection quality by eliminating the detection errors of the given object detector. We prove that training the secondary classifier improves the detection performance without requiring additional training datasets.
- **The connection function between two networks**. We discuss how to combine the two networks by introducing various combination procedures. We also propose an optimized combination method to improve the performance of the final detection scores. Remarkably, the proposed combination method can be used to efficiently train the classification network. The final output has higher accuracy than the stand-alone proposal network as an object detector. The experimental results obtained using the proposed method are comparable to those obtained using state-of-the-art object detection methods.
- **A proposed new network design to extract a rich and multi-scale feature maps for object recognition**. Our experiments show the improvement in the feature extractor helps to improve the quality of instance object segmentation.

The thesis also discusses how to select the detection network for specific object classes of interest as well as the effect of our model on the overall performance. Given a proposal detector, the proposed model improves the performance of many object classes across different settings, especially for small objects. We evaluate our proposed model with extensive experimental tests.

1.7 Outline of thesis

Chapter 1 provided an overview of our research goals and the applications of object detection. We briefly present the object detection challenges, backgrounds of common object detectors, our approach for the deep-learning-based detector, and a summary of our contributions. The remaining chapters are organized as follows:

- Chapter 2 introduces the state-of-the-art methods in object detection. We describe the main approaches to object detection and show their contributions. We also explain our motivation for using deep learning methods for object detection.

- Chapter 3 presents a high level overview of our framework for object detection. In this chapter, we introduce a key idea of our approach and the overall architecture of our model.

- Chapter 4 describes the proposed models in detail. It presents two variations of our proposed models: post-combination and trained combination. Our models are trained and tested using several object detection datasets. The results show that our proposed models help to improve the performance of given proposal object detectors. The two-stage model not only improves the performance of the proposal network, but also achieves state-of-the-art results in small-scale object detection.

- Chapter 5 proposes a new design for a feature extraction network. The new feature extractor is used to detect objects and segmentation inside the detected bounding box (instance segmentation). In this chapter, we also present advantages of the new design and the potential for object detection by a bounding box.

- Chapter 6 discusses the use case of our proposed model. It summarizes the key contributions of our method and discusses limitations. Finally, we suggest directions for future work.

Chapter 2

Related research

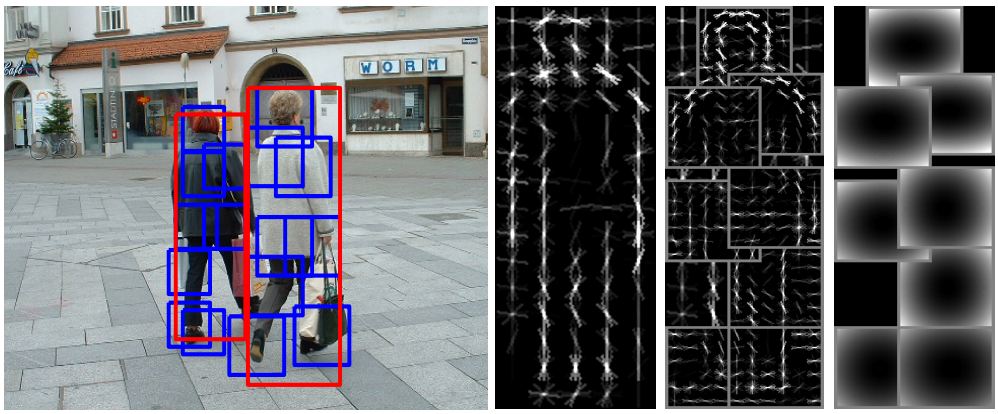
Object detection in images and videos has occupied thousands of experts and computer science communities over several decades. Although it has been a very challenging task, many applications use object detection as a primitive process. Progress in object detection has grown quickly in recent years, with each year's new detectors outperforming the previous ones. This chapter describes the related research in object detection. We review the contributions of each work and focus on the two main detector components:

- The feature extractor, and
- The detection framework that uses the feature extractor

We review the methods related to our research and point out the advantages and disadvantages of different detectors. Subsection 2.1 presents the state-of-the-art handcrafted-feature-based models. Subsection 2.2 covers the learned-feature-based models, which mainly focus on deep learning. Subsection 2.3 shows our motivation based on the reviewed methods.

2.1 Handcrafted-feature-based methods

Handcrafted-feature-based methods rely on the handcrafted feature extractor, which we briefly described earlier. The aim of this approach is to extract the most relevant features needed to predict object classes and locations. A feature can be the simplest descriptor, such as color channels and the gray image, linear filters (e.g., Difference of Gaussian (DoG)), non-linear filters (e.g., gradient), or the gradient histogram. The first notable attempt was by Viola and Jones (Viola, Jones, and Snow, 2003; Viola and Jones, 2004), who applied their (Viola-Jones) VJ detector to the task of pedestrian detection by using the Haar-wavelet-like features descriptor and AdaBoost (Freund, Schapire, et al., 1996) to train cascades of weak classifiers. This method combines both the appearance information and the motion information to detect pedestrians in low resolution and difficult conditions (such as rain and snow). In 2005, Dalal and Triggs introduced the histogram of oriented gradients (HOG) feature descriptor (Dalal and Triggs, 2005a), which significantly improves the performance of people detection as well as general object detection. The method used HOG feature vectors and linear support vector machine (SVM) to output the object/non-object classifications. The deformable part model



(A) (B) (C) (D)

FIGURE 2.1: Deformable part model and detection examples. There are three components in this person detection model: a root filter (2.1b), higher resolution part filters (2.1c) and associated deformation models (2.1d). This visualization shows the positive weights at different orientations. The detection results are shown in image (2.1a).

(DPM) (Felzenszwalb, McAllester, and Ramanan, 2008; Felzenszwalb et al., 2010), used HOG as a building block. Handcrafted-feature-based models have been successful in detecting small-size datasets, and have been critically important for state-of-the-art object detection methods for a long time. These types of models are often referred to as classic detectors.

We review the two main methods that use the handcrafted feature extractor with their two contributions: a multi-part-based detection and box prediction, and a multi-image channel combination for box classification.

2.1.1 Deformable Part Model

The Dalal and Triggs detector (Dalal and Triggs, 2005a) based on the HOG feature extractor and linear SVM helps improve the overall object detection performance compared with previous works. It requires only one model for each object class to calculate the weighted HOG feature for object prediction. The positions of objects in the image depend on the locations of sliding windows. This approach limits some object variants as described in Subsection 1.3. More specifically, nature objects have many variants under non-rigid deformations, intra-class variability, and different viewpoints. These variants require a better representation of objects as a deformable part model.

The deformable part model is based on the pictorial structure frameworks (Fischler and Elschlager, 1973; Felzenszwalb and Huttenlocher, 2005), and represents the deformable components by spring-like connections between pairs of parts. The model utilizes a star-structure, part-based model with three components: a "root" or "body" filter, a set of part filters and a deformation model. The part-based model is based on the idea that objects in the same class share the same structure. The most difficult part of training is the *latent information* since the bounding boxes of parts are not labeled (the weakly labeled setting).

The part-based model is depicted in Figure 2.1[1]. The "root" filter is used to calculate the response of the "root" filter on a HOG feature map. Likewise, the part filters are used to calculate the response of part filters at higher resolution. The final summarization is to maximize the local responses and minimize the global shape deformation. In the work of Felzenszwalb (Felzenszwalb et al., 2010), bounding box prediction is improved by using multiple overlapping detections for each object and eliminating weak detection via non-maximum suppression (NMS). Following the work of the DPM detector, some variant methods have been proposed, such as multi-task (MT)-DPM (Yan et al., 2013). Interestingly, the models that use only a single component, such as Squares Channel Features (Benenson et al., 2014), Roerei (Benenson et al., 2013), can output similar performance for specific object detection tasks (e.g., pedestrian detection).

Regarding the bounding box prediction, DPM uses a linear least-squares regression. The object bounding box containing four coordinates (upper left and lower right corner) is trained by linear functions, and the input data are the coordinates of object parts. Notably, this simple method provides significant improvements in accuracy for predicting locations of some object classes. Since object localization is an important part of object detection, it also improves the overall detection performance.

2.1.2 Integral channel features (ICF)-based methods

Integral channel features (ICF or ChnFtrs) (Dollár et al., 2009) is a popular method for object detection. The idea behind ICF is to use an integral image to extract features such as histograms, the local sum, and Haar-like features from multiple registered image channels. The input images are transformed by linear and non-linear functions to compute the image channels. With a starting point of the handcrafted feature extractor, ICF addresses two problems: analyzing or optimizing the features, and efficient feature computation. The general algorithm can be summarized by the following steps:

- Compute multiple image channels using linear or non-linear transformations of the original input image. For example, the common registered channels are: gray and color, linear filters (Gabor filter, DoG filter) and non-linear (gradient magnitude, Canny edges (Canny, 1986)).
- Extract the features from each channel. The candidate features are selected randomly. For example, with the sum over a rectangular region in a given channel, the detector can choose a random channel index and rectangle.
- Apply the Boosting classifier (Friedman, Hastie, Tibshirani, et al., 2000) with weak classifiers as the depth-two decision trees. The benefit of using the Boosting classifier is fast learning. The method can use other classifiers, such as SVM, to classify features.

[1] We used the implementation from Girshick, Felzenszwalb, and McAllester, 2012. The tested images are from the Institut National de Recherche en Informatique et en Automatique (INRIA) person dataset (Dalal and Triggs, 2005b)

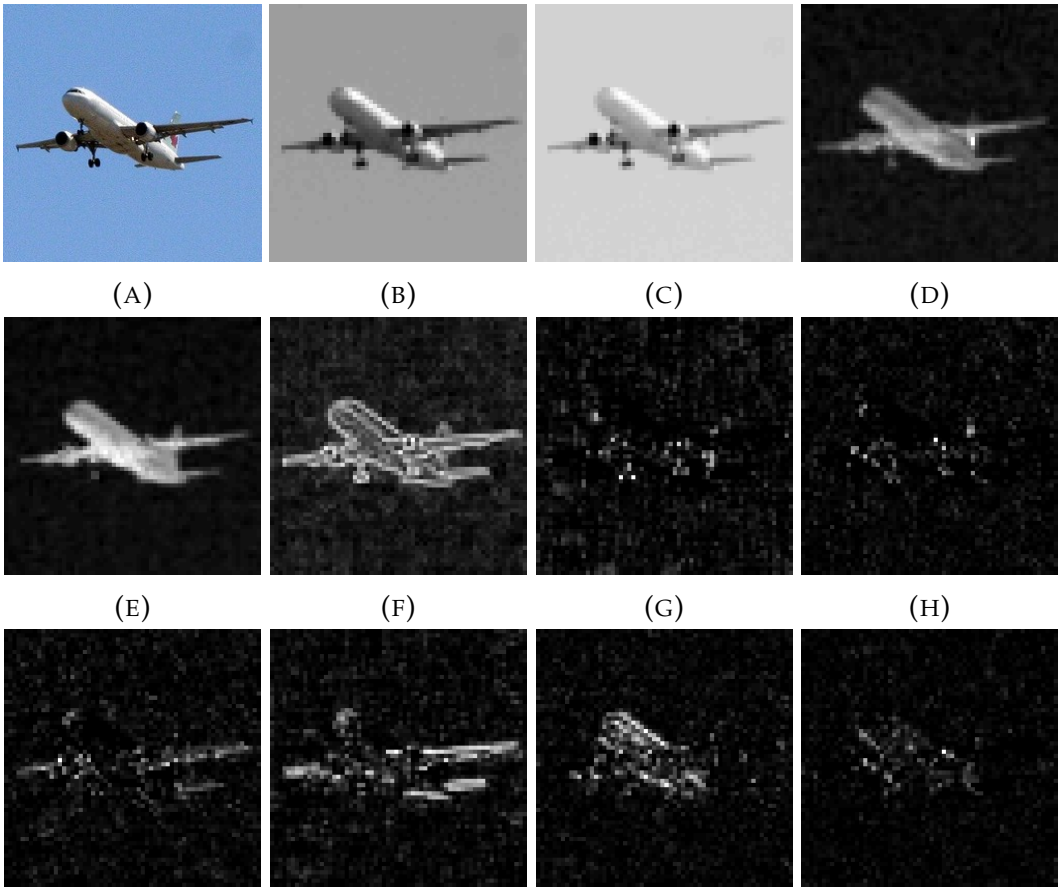


FIGURE 2.2: Example image (A) and computed channels. The simplest image channel is gray image (B). A color image can also be used, such as CIE-LUV channels (C), (D) and (E). Other non-linear transformation channels include: the gradient magnitude (F) and the gradient histogram, which is quantized to the 6 orientations shown in (G) - (L).

- Finally, the classifier is used to classify the object. The ICF detector uses the sliding window method over multiple scales to predict the object location.

Some examples of channel types are shown in Figure 2.2. Typical image channel types include a gray image (2.2b), LUV color channels (2.2c, 2.2d and 2.2e), the gradient magnitude (2.2f), and the gradient histogram (2.2g - 2.2l).

The ICF presents a method to extract features by capturing the richness from different channels. To study the performance of various channel types, the detector separates each channel to evaluate detection performance. Although each channel has a different contribution, the more channels that are combined, the more accurate the detection.

For the bounding box prediction, the ICF method uses a sliding window scheme. Compared to the HOG detector, ICF uses a smaller window step size so that it can predict more precise object locations. However, bounding box regression has not been used with the DPM method.

Some improved versions of ICF include (Benenson et al., 2012; Benenson et al., 2013) for the pedestrian detection task. Another notable variant of ICF is aggregate channel features (ACF) (Yang et al., 2014; Dollár et al., 2014) for general object detection. To increase detection performance, ACF extracts richer image representations by computing multi-resolution image features. ACF addresses computational cost by introducing a fast features approximation. This approximation, in addition to the boosted tree archives, leads to state-of-the-art performance for many object detection tasks.

The variable of extra features as input channels shows improvement over the baseline detectors. It is especially useful for many pedestrian detectors that are based on decision forests. For example, Daniel Costea and Nedevschi, 2016 utilized semantic segmentations as context information for pedestrian detection and integrated them as an extra channel for detecting. The semantic channel detector benefits from the semantic information in a large receptive field. Another example is the ACF+SDt (where SDt is the stabilized Dt-motion feature) method (Park et al., 2013), which embeds the optical flow into channel features and uses a boosted decision forest to detect objects in videos. Motion and optical flow uses are sometimes referred to as *temporal channels*, and are often used to detect objects in videos. The ICF method can also be applied to feature types captured by other sensors, such as depth information. For instance, DispNet (Mayer et al., 2016) provides large-scale datasets for training and evaluating scene flow.

2.2 Deep neural network based methods

Deep neural network based methods (also known as deep learning methods) are a class of *data-driven* methods that use deep networks to train the feature extractor. Deep learning methods allow a machine to automatically discover the representations needed for classification or detection. The network contains multiple levels of representation, starting with the raw input image, which the deep learning network transforms into a higher, more abstract level of representation. For classification tasks, which are a core problem in computer vision, deep learning methods eliminate irrelevant variations and amplify the information for classification. For example, with image data, the input is a three-dimensional array of numbers, and the features in the first layer often represent the absence of edges in the image. The next layer identifies particular arrangements of edges with varying edge positions. These feature layers, however, are not defined by human engineers. The features are learned from data by a training process.

Deep learning has been applied with great success in object detection and classification, making major advances in solving computer vision problems, such as handwritten digit recognition (LeCun et al., 1990), traffic sign recognition (CireşAn et al., 2012), and face recognition (Taigman et al., 2014). Convolution neural networks (CNN) are also able to classify at pixel-level, making them suitable for image segmentation.

Despite these successes, deep learning was largely ignored in favor of other machine-learning methods until the release of ImageNet (Deng et al.,

2009), which used the WordNet Hierarchy (Miller, 1998). AlexNet (Krizhevsky, Sutskever, and Hinton, 2012) has the highest record for image recognition by a large margin. The success of deep learning comes from two factors: an ability to learn very complex functions for rich input representation (feature extraction) and sufficiently-sized diverse training data. Although state-of-the-art deep-learning-based feature extractors are mostly designed for classification tasks and are often trained with large amounts of data (1.2 million images in ImageNet), they can be efficiently applied to other smaller datasets via *transfer learning* and to other related tasks (e.g., detection, segmentation) via *multi-task learning*. We discuss this in Subsection 2.2.1.

2.2.1 Object detection using deep learning

In this section, we use object detection networks to denote object detection using deep learning methods. In practice, most object detection networks are not trained from scratch because the dataset for training is not big enough. For example, the ImageNet training set typically consists of 1.2 million images compared with about 5,000 images in Pascal visual object classes (VOC) train-val set (Everingham et al., 2010). With the same type of data, the deep network is able to store knowledge gained while solving one problem (e.g., ImageNet classification). When applied to other tasks or other datasets, the pre-trained network is used as an initialization or a fixed feature extractor. This use of deep networks is called transfer learning. Applications of transfer learning include the following.

- **Deep network as fixed feature extractor**: the weights of the pre-trained network are fixed except for the last fully-connected layers. The recognition model uses the output of the network and a classifier to train for a new dataset.

- **Fine-tuning deep network**: in this case, the model uses the pre-trained network parameters as initialization and slowly changes them by training with the new dataset. It is possible to keep some layers of the pre-trained network unchanged (the low-level general representation layers) and fine-tune only some high-level layers.

- **Pre-trained models**: the model uses the pre-trained network directly.

The use of the deep network as a feature extractor for object detection has been studied in several works. For example, convolutional channel features (CCF) (Yang et al., 2015) uses a concatenation of low-level deep network feature extraction and a boosting classifier for face detection and pedestrian detection. The benefits from the rich features of the deep network guarantee high performance for various vision tasks. Similarly, Hosang et al., 2015 and Tian et al., 2015 used the deep learning based method with boosted tree classifiers for pedestrian detection.

Regarding object detection by the deep network, the detection task or segmentation task can be reduced to the classification task and regression task by multi-task learning. The most successful general object detections are based

on the region-CNN (R-CNN) (Girshick et al., 2014) family method. The detector consists of two key components: a score function that maps the input image to the output score, and a loss function that measures the quality of the detection task. The object detection task must predict the object class and object location. In a deep neural network, multi-task learning typically shares parameters of hidden layers while maintaining several task-specific output layers. For example, one of the primary works using multi-task learning for object detection is the Fast R-CNN method (Girshick, 2015), which has the following two outputs for each Region of Interest (RoI): softmax probabilities and per-class bounding-box regression offsets. The multi output network allows end-to-end joint training with multi-task loss. The network is first pre-trained by ImageNet and then fine-tuned with the specific-task dataset (e.g., Pascal VOC). Faster R-CNN (Ren et al., 2015) introduced the Region Proposal Network (RPN), which is used to calculate object proposals.

2.2.2 One-stage models

In this section, we discuss one-stage object detection models, which use only a single-stage network without region proposals. This type of detector trains quickly and can be efficiently deployed. Some examples are SSD (Liu et al., 2016), You Only Look Once (YOLO) (Redmon et al., 2016), deconvolutional SSD (DSSD) (Fu et al., 2017), and Overfeat (Sermanet et al., 2013). Other methods extend this approach to predict boxes, classes, and poses (Poirson et al., 2016).

Single Shot Detector (SSD)

SSD is a single-shot feed forward network based on a pre-trained network for object detection. The SSD model is very similar to the RPN component of the Faster R-CNN model, except that the SSD model directly predicts class-specific and box offsets, and does not require a second object detection network. Instead of using a CNN module to extract the set of RPN, the SSD uses a fixed set of default boxes (anchors) for prediction, and thus can avoid merging the RPN module with Fast R-CNN. The SSD model can generate a large pool of possible box shapes by discrete output into a set of default boxes of different scales and aspect ratios at several feature map locations. This approach allows the SSD model to achieve slightly better speed than Faster R-CNN-like detectors.

You Only Look Once (YOLO)

YOLO (Redmon et al., 2016) is a single neural network for object detection. Since most accurate object detectors are slow, YOLO's goal is to achieve real-time object detection. In this approach, object detection is a regression problem of image pixels to bounding box coordinates and class probabilities. For simplicity, the input image is passed through a deep network that look like a normal CNN. The output is a vector of bounding boxes and class predictions. The image is divided into an $S \times S$ grid of cells. Each cell predicts

B bounding boxes and C class probabilities and confidence levels for those boxes. The output predictions are encoded as an $S \times S \times (B \times 5 + C)$ tensor. There are two changes in the multi-part loss function for better results: 1) the confidence prediction of boxes containing an object and boxes not containing an object are different, and 2) predict the square root of the bounding box width and height instead of directly predicting the width and height. Improved versions of YOLO include YOLO9000 (Redmon and Farhadi, 2017), which has accurate localization while maintaining a high speed. A number of improvements have been applied, such as batch normalization on all convolutional layers, a high-resolution classifier (from 224×224 to 448×448), convolutional with anchor boxes, dimension clusters, fine-grained features, and multi-scale training.

2.2.3 Two-stage models

Although one-stage detectors are fast and have reasonable detection accuracy, the most accurate detection methods are two-stage models. Two-stage detectors rely on two processing steps. The first successful model was R-CNN, which was followed by a number of variants, such as Fast/Faster R-CNN, R-FCN (Dai et al., 2016), Zagoruyko et al., 2016; Bell et al., 2016, PVANET (Kim et al., 2016), Shrivastava and Gupta, 2016, Shrivastava, Gupta, and Girshick, 2016, Yang et al., 2016, Zhai et al., 2017.

Fast/Faster R-CNN

Fast R-CNN (Girshick, 2015) is a deep learning object detector that combines an object proposal method (e.g., Selective Search (Uijlings et al., 2013)) and a CNN classifier. Fast R-CNN introduces a Region of Interest (RoI) pooling mechanism and multi-task losses by minimizing the loss functions of both the class confidences and the bounding box regression.

The improved version of Fast R-CNN, i.e., Faster R-CNN (Ren et al., 2015), replaces the region proposal component with a deep network. Faster R-CNN has two components: an RPN and a Fast R-CNN (Girshick, 2015) object detection network. The first component, the RPN, is a CNN that predicts class-agnostic box proposals (object and non-object). These networks can be trained separately or end-to-end, and they share the extracted image features with the object detection network.

The shared features are fed to the remaining layers of the feature extractor. The second component uses the box proposals to crop features and then outputs the class-specific and box offsets for each proposal.

Region-based Fully Convolutional Networks (R-FCN)

In contrast to previous region-based detectors, such as Fast/Faster R-CNN, that apply the region-specific component several hundred times per image, the R-FCN method (Dai et al., 2016) is a fully convolutional detector that shares the computation on the entire image. The method adopts the region

proposal and region classification of Fast/Faster R-CNN. The feature cropping is taken from the last layer of features for detection in contrast to Faster R-CNN, which crops features from the same layer where the RPN predicts the object proposals. Because the cropping occurs at the last layer, the per-region computation is minimized.

The R-FCN model has comparable accuracy to Faster R-CNN and often runs faster. R-FCN has been influential in a number of follow-up works, such as Deformable Convolutional Networks (Dai et al., 2017), FCIS (Li et al., 2016).

Mask R-CNN

Mask R-CNN (He et al., 2017) is a method for instance segmentation. Whereas Faster R-CNN predicts object locations and class labels, Mask R-CNN extends this architecture by adding a third branch that outputs the object mask. The object mask is used to predict the pixel-level inside the object location (instance segmentation). The mask branch predicts a fixed-size mask $m \times m$ for every class, resulting in an N binary mask, where N is the number of classes. Although the concept behind Mask R-CNN is simple, the pixel-level prediction requires finer feature extraction. The method introduces RoIAlign, which replaces the RoIPooling layer of Faster R-CNN, and leads to large performance improvements.

2.3 Motivation of this research

2.3.1 From handcrafted feature to deep learning

Handcrafted feature extraction based models appear inferior to deep neural network based models. The accuracy gap between the two approaches is more significant for object detection with a large-scale dataset. Although some methods that use handcraft feature extraction, such as HOG (Dalal and Triggs, 2005a), ICF (Dollár et al., 2009), and its variants, can achieve highly accurate detection for specific tasks, the diversity of extracted features is limited. The HOG feature descriptor performs remarkably well in DPM (Felzenszwalb, McAllester, and Ramanan, 2008; Felzenszwalb et al., 2010) in terms of addressing the wide variance in nature objects. However, there are some methods which outperform DPM (Benenson et al., 2013) by using a single "root" filter without parts, indicating the necessity of components and parts. In the early history of object detection with the sliding window classification approach, the type of classifier was also important to detection performance. Boosting methods like AdaBoost strongly impacted detection results. With limited image representation by handcrafted feature descriptors, some methods, such as CCF (Yang et al., 2015) and RPN+BF (Zhang et al., 2016), use deep networks for feature extraction followed by a boosting classifier. With the richer representative capacity of CNN, these methods improve performance in various vision tasks. The role of classifier types (e.g., the boosting decision tree and linear SVM) was argued in the works of MultiFtrs (Wojek

and Schiele, 2008) and Ohn-Bar and Trivedi, 2016. This thesis uses learned feature descriptors for image feature extraction. CNN features are able to encode high-level concepts of the object while lower-level feature maps detect simple concepts like edges and shapes.

In computer vision, CNNs achieved a large decrease in error rate for object classification and detection. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) shows the highest-ranked methods using CNN-based frameworks. In 2012, AlexNet (Krizhevsky, Sutskever, and Hinton, 2012) significantly outperformed other methods in the ILSVRC, and was the first work to popularize CNN in computer vision. The ILSVRC evaluated algorithms designed for large-scale object detection and image classification. The performance of AlexNet proved a hypothesis: the right algorithm and a large amount of data (ImageNet contains 1.2 million images for training) could be the key to successful AI. Compared to LeNet (LeCun et al., 1998), AlexNet had similar architecture, but was much bigger and deeper. Following the success of AlexNet, a number of CNN methods, such as visual geometry group (VGG)Net (Simonyan and Zisserman, 2014), GoogleNet (Szegedy et al., 2015), and ResNet (He et al., 2016), were the top-performing methods in the ILSVRC. Inside state-of-the-art CNNs, there are some technical improvements: AlexNet and VGGNet show that the depth of the network is a critical component for good performance. However, the drawback of these networks is that they use a lot of memory with huge numbers of parameters (about 140M for VGGNet and about 60M for AlexNet). GoogleNet uses the Inception module, which significantly reduces the number of parameters (about 4M). ResNet proposes the skip connection and the use of batch normalization. The success of deep-network-based methods inspires us to utilize CNN as a classifier and detector for our thesis. However, there is much room for improvement, such as multi-scale feature encoding, optimization, data augmentation, context embedding, and stabilized training.

The potential of CNN for object recognition task has recently increased even further. The object detection performance has strong positive correlation with classification performance (Huang et al., 2017). In addition, the detection algorithm is an important component in the overall detection framework. Our approach is based on a two-stage object detector, by which the image is processed by a proposal network and a classification network. In the two-stage network, the connection between the two stages is important for higher-level reference, better optimization, and error reduction.

2.3.2 One-stage and two-stage object detectors

We consider two approaches for object detection modeling. In the first approach, one-stage detectors (unified detectors), such as SSD (Liu et al., 2016) or YOLO (Redmon et al., 2016; Redmon and Farhadi, 2017), use a single CNN to predict object location in an entire image. Since one-stage detectors require only single-network computation, they are quicker than other CNN-based detectors. However, the limitation of YOLO is that detecting small

objects is difficult and it does not work well with unusual object aspect ratios. Although SSD provides better object localization than YOLO, because of location sharing for multiple categories, the SSD method involves increasing confusion with similar object categories. Moreover, SSD and multi-scale (MS)-CNN (Cai et al., 2016) independently predict objects at multi-feature map locations because there are no combinations of features or scores.

In the second approach, two-stage detectors, such as Fast/Faster R-CNN (Girshick, 2015; Ren et al., 2015) and R-FCN (Dai et al., 2016), require the first stage to extract object proposals. By refining the object proposals twice (once when refining the anchors to class-agnostic box proposals and once when refining the RPN output to class-specific boxes), Fast/Faster R-CNN-like methods can obtain better detection results than one-stage detectors. However, training these detectors requires significant effort because it is difficult to optimize each network component.

The proposed model uses a two-stage approach in which the proposals are class-specific boxes. Without combining the first and second stages, the proposed model is equivalent to a non-box-regression Fast R-CNN detector with a pre-computed RPN. The proposed method is also different from Fast/Faster R-CNN in terms of sampling. The extracted RPN proposals of Faster R-CNN are in the same image or the same image batch, whereas the second stage can freely shuffle all the training samples in the training dataset. From Bengio, 2012, it is efficient to optimize the classifier in which the order of samples is changed for each epoch, and each sample is sampled independently.

Chapter 3

Object detection methodology

Our aim is to propose a deep learning framework to improve object detection performance. The process includes reducing detection errors and improving the accuracy of detections. To reduce detection errors, we first analyze the impact of different error types. In particular, the performance of object detection is summarized over classes or groups of classes. This summary does not tell us why one method outperforms another in detail. Detection error analysis allows us to determine which aspect of object detection could be improved. We then improve detection performance by proposing a better detection framework. We focus on two approaches:

- Finding a better image feature extractor, and
- Other techniques such as data preprocessing and robust optimization.

Here, we introduce our detection framework, give an overview of our design considerations, and describe our training and inference processes. We also summarize our key findings. The chapter is organized as follows: Subsection 3.1 presents the object detection error analysis and the contribution of each error type to the overall detection performance. Subsection 3.2 introduces our overall detection architecture, which is separated into two processes: training network design and inference network design. It also presents the impact of using the first-stage score for the second-stage training and model inference. Subsection 3.3 introduces the optimization process that involves the proposal detection scores.

3.1 Detection errors reduction

The output of the detector includes a set of detections where each detection contains at least the following: the rectangle that covers the object, the object class, and the confidence level (see Appendix B for more details). The detections are assigned to ground-truth objects and are judged to be true or false positives by measuring the bounding box Jaccard index, also known as Intersection over Union (IoU). In particular, one detection with a predicted bounding box B_p is assigned to a ground-truth bounding box B_{gt}. The IoU is calculated by the formula

$$IoU(B_p, B_{gt}) = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \tag{3.1}$$

where *area* denotes the area of the region, $B_p \cup B_{gt}$ is the union of the two boxes, and $B_p \cap B_{gt}$ is their intersection. For a specific evaluation task, the true positives detection must exceed a threshold (e.g., 50%). Hence, the detection results depend on the location of the predicted bounding box, the size of the box, and the predicted class. One image can contain several objects and each object corresponds with one ground-truth. The detector often uses a sliding window scheme or a set of box proposals to predict the object location, which leads to multiple detections for a single instance of an object. However, only one detection (that with the biggest IoU) is chosen as the correct detection; the others are considered false detections. In this case, the detector often sorts the detection by confidence in decreasing order. Unfortunately, the confidence and the IoU do not always correspond. For example, correct detections with low confidence levels do not contribute to the detection performance. Correct detections with high confidence but low IoU are often considered false detections and, thus, reduce the overall detection performance. Details of evaluation metrics for object detection are given in Appendix A.

In the object detection task, the detection results usually contain numerous false positive detections. There are two kinds of false positive errors: poor localization (an object from the target category is detected with a misaligned bounding box) and misclassification (the detected object matches with other categories or background). For instance, in the SSD method (Liu et al., 2016), more than half of false positive detections are due to misclassification. Figure 3.1 shows examples of misclassification detection due to ambiguity with background objects, similar objects or dissimilar objects. This issue motivates the use of an additional classifier to reduce the number of misclassification detections. In other words, we re-order the proposed detections. Note that recall (also known as the true positive rate) for each false positive detection in Figure 3.1 is calculated by the number of true positives with higher confidence than that detection divided by the number of real positives in the dataset. Therefore, (1-recall) is the fraction of correct examples that are ranked lower than the given false positive.

The idea of using two stages to detect objects has been considered in numerous studies (Girshick, 2015; Ren et al., 2015; Lin et al., 2017). These models rely on an external or internal region proposal generator (as the first stage) to predict class-independent box proposals, allowing the box proposals to be separated into objects and non-objects (the background). The second stage is used to predict the class and offset the shifting proposal location to fit the ground-truth bounding box.

In the proposed approach, we similarly use an object detector as a proposal generator. However, the proposals extracted using this detector contain not only box locations, but also class labels and scores. We call this detector the proposal network. We then add a convolutional neural network, called the classification network, to re-classify the extracted proposals. The outputs of the two networks allow us to choose either the first-stage scores or the second-stage scores as the final scores. However, we observed that

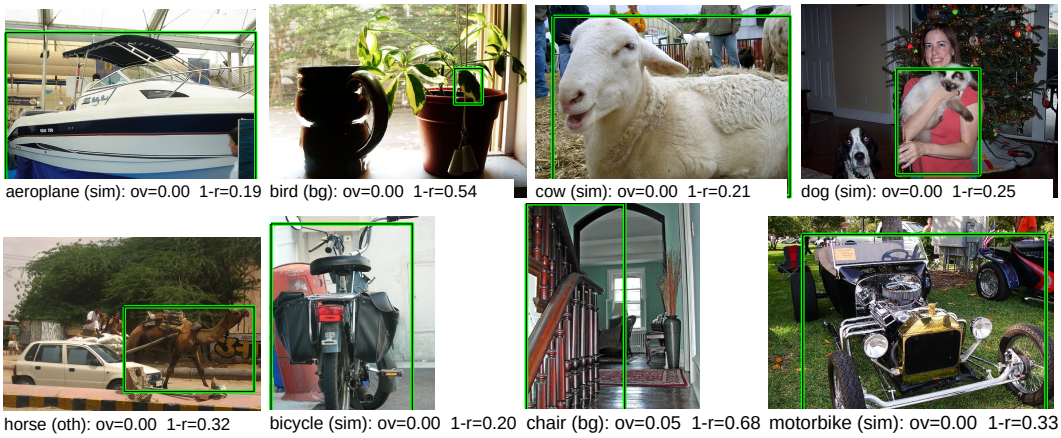


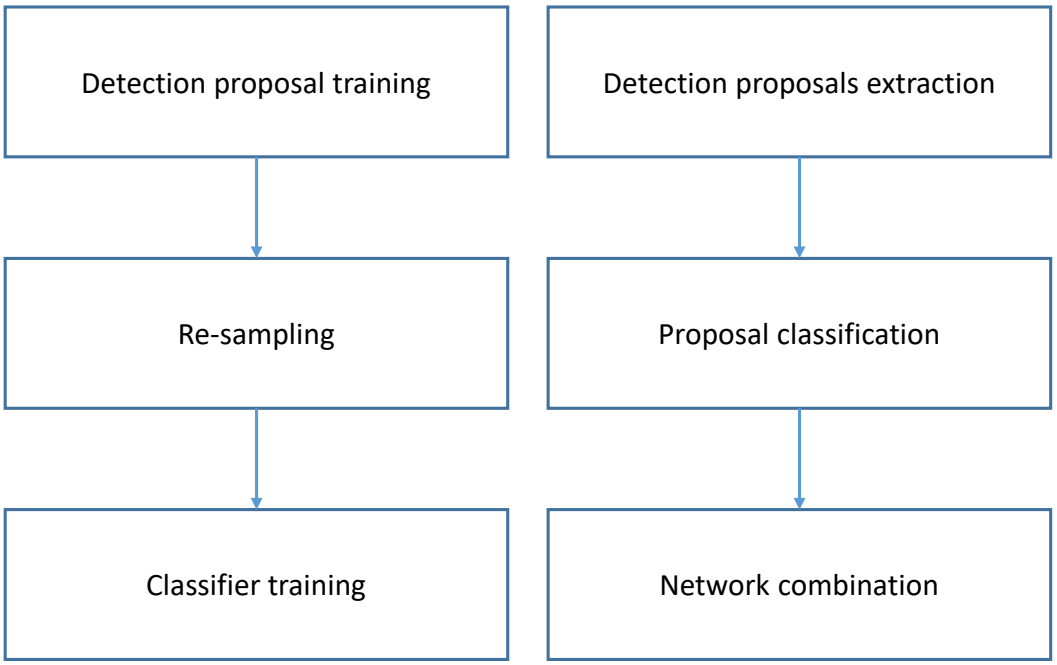
FIGURE 3.1: False positive detections on the Pascal VOC dataset by the Faster R-CNN object detection model. The green bounding boxes depict the object location inside an image. The text under the image indicates the detected class, and the type of misclassification (sim: confusion with similar object, oth: confusion with dissimilar object, bg: confusion with background), the overlap with the ground-truth bounding box (ov), and the 1-recall value (1-r).

the second-stage scores after re-classification generally do not boost detection performance. We suggest two reasons for this unsatisfactory result: (i) the classification network focuses on classifying objects based on their similarities and differences, but lacks localization support, and (ii) there are no connections between proposal confidence and classification confidence.

3.2 Overall architecture

The overall detection architecture is built on a two-stage deep neural network. There are two distinct phases (see Figure 3.2): the training phase and the testing phase. The training phase uses input images to fit the parameters of model. In the object detection, the training dataset consists of input images, object labels, and object bounding boxes. The testing phase is used to evaluate the final model fit on the training dataset. Each phase has three main steps. This simple architecture is complicated in practice because increasing model complexity often leads to over-fitting. The overall performance of the model also depends on the accuracy of both model stages, the reliability of the combination procedure, and the effectiveness of the training.

The first stage of the training phase is detection proposal training. Essentially, this model component works as a region proposal network in two-stage detectors. The first stage is called the *proposal network* and is trained separately. The difference in our approach is that we utilize the output confidences for the next stage of training or to test phase inferencing. The proposal extraction of the first stage has some advantages. First, it can perform detection directly from the proposal network, thus allowing changes to the overall model component. It is easier to test the impact of different network proposals on the total detection performance. Second, in the next step, the extracted proposals can be re-sampled with different sampling strategies and different



(A) Model training phase (B) Model testing phase

FIGURE 3.2: Overall object detection architecture. (A) The training phase includes training the detection proposal, re-sampling the extracted proposals, and training the second-stage classifier. (B) The testing phase includes object proposal prediction, re-classifying the proposals, and combining the two stages.

datasets. For example, since the detection proposal training is more complex than the classifier training, it could require an additional training dataset whereas the next stage classifier needs to focus on the specific dataset. The training samples for the classification network are also independent with a selected proposal network. Finally, the proposal network works with a large number of input windows (e.g., sliding windows, anchor windows, or RPN windows), which can produce many false positives. The overall performance of the model is very sensitive to the false positive rate of the classification network. To address this problem, a soft rejection that re-orders the output detections could help improve the detection results. In contrast to some classic object detection models, such as the VJ detector (Viola and Jones, 2001), which used a detection cascade to reject sub-windows by a sequence of classifiers, our model does not eliminate any region proposals.

The increasing network complexity can cause a computational problem, but the increasing cost is reasonable. In practice, for example, the proposal network with the Pascal VOC dataset could extract thousands of proposals for training, but needs fewer than 100 predictions for testing and can be effectively processed by simultaneously forwarding these proposals to a deep network.

The object proposals after re-sampling are fed to the classification network for the classifier training. At this step, we can incorporate the confidences of object proposals with classifier training samples (which only require the bounding boxes and classes) to create a better classifier. This combination process is a major topic of this thesis and is discussed more concretely below.

Recent classifiers by CNN generally contain two components: one feature extractor by multiple convolutional layers (and other layers) and a classifier, such as linear SVM or softmax as the last layer. The details are presented in the Appendix B. However, with sufficient training data, the SVM and softmax classifiers perform similarly. In our classifier model, we mainly use the softmax classifier as it was proved to be the more accurate and reliable classifier in our experiments.

During the testing phase, the input image is fed to the proposal network. The outputs of the proposal network are pre-processed and forwarded to the classification network. Because the sizes of proposal boxes are different, the boxes are resized to the classifier input size. The detection confidences are computed by the combination functions using the first-stage and second-stage confidences. The details of combination functions are described in Chapter 4.

3.3 Overview of the training process

3.3.1 Multi-step training and end-to-end training

We train the first stage and the second stage separately. In the first stage, both one-stage (such as SSD) and two-stage (such as Faster R-CNN) networks could be used. The network requires pre-computed *anchors* that cover the input windows with different scales and aspect ratios. An anchor is a region proposal centered at the sliding window that is associated with a scale and aspect ratio. For example, in RPN, the anchors are located in image grids and are annotated with a binary class label (of being an object or not). The anchor labels differ from box prediction as one ground-truth may assign positive labels to multiply anchors. The IoU threshold of the first stage could be higher than that of object prediction. For example, with the Faster R-CNN model (Ren et al., 2015), the RPN threshold is 0.7 and the detection threshold is 0.5. Some anchors are not trained because the IoU is "ambiguous" between positive and negative labels (e.g, the anchors with IoU from 0.3 to 0.5).

There are two approaches for training: training each component of the model alternately or end-to-end training the whole model. In the first approach, we train the RPN and use the proposals to train the second component (e.g., Fast R-CNN). The network fine-tuned by the second component is then used to initialize the first component, and the process is repeated. Although this approach is complicated, the performance is better than the model that pre-computes RPN proposals. To perform the second training approach, we need to define the joint training loss function, which is the sum

of two multi-task loss functions. End-to-end training is easier to implement and optimize.

3.3.2 Classifier training

Hard negative mining

After the proposal extraction step, the output contains many negative proposals, especially when the number of anchors is large. To improve the stabilization of the second-stage training, it is better to maintain the ratio between the number of positive and negative samples. This sample preparation can be done by choosing the top confidence level instead of using all negative proposals. Another consideration a low number of positive proposals.

Training with confidence

The confidence score of the proposal network can be used to train the classification network. A new design with fully-connected layers of the classification network concatenates the confidence of the first stage to form the output of the network.

Chapter 4

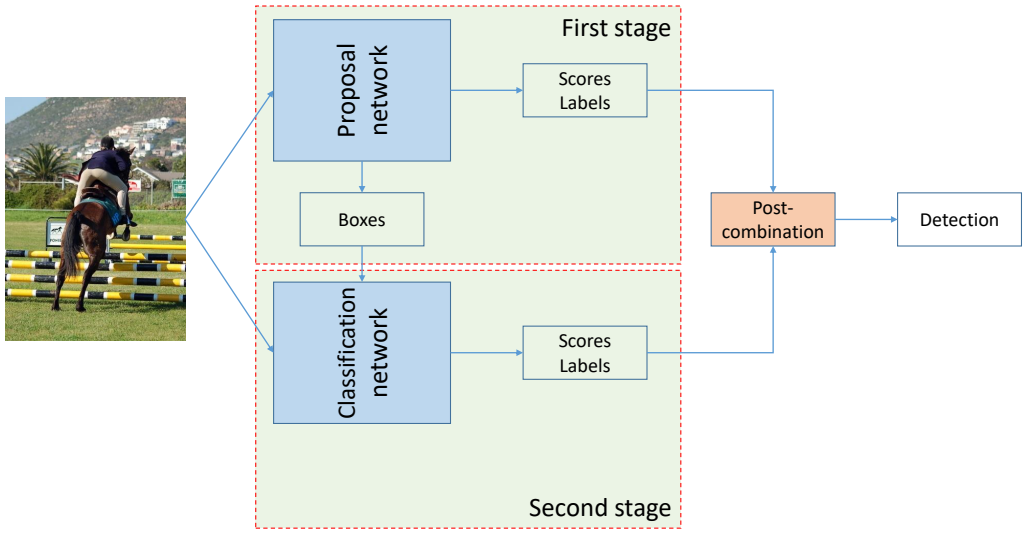
Two-stage object detection model

This chapter describes the two-stage object detection deep network. We introduce the details of the proposed framework with each model component, the network architecture, combination functions, and the effect on model performance. The two-stage network not only improves detection performance compared with the single-stage network, but can also be applied to various datasets. We first define the network architecture in Subsection 4.1, where we present two different designs. Subsection 4.2 presents details of the first stage (proposal network). We consider several architectures for specific tasks. In Subsection 4.3, we describe the second stage (classification network) and the effect of this classifier on overall performance. Subsection 4.4 presents details of combination functions that are used to combine the two network stages. Experimental results are given in Subsection 4.5. To validate the effectiveness of our proposed model, we performed detection on several datasets with different model settings and evaluation metrics. We present extensive result analysis and methods to optimize our model. Finally, we discuss these results in Subsection 4.6.

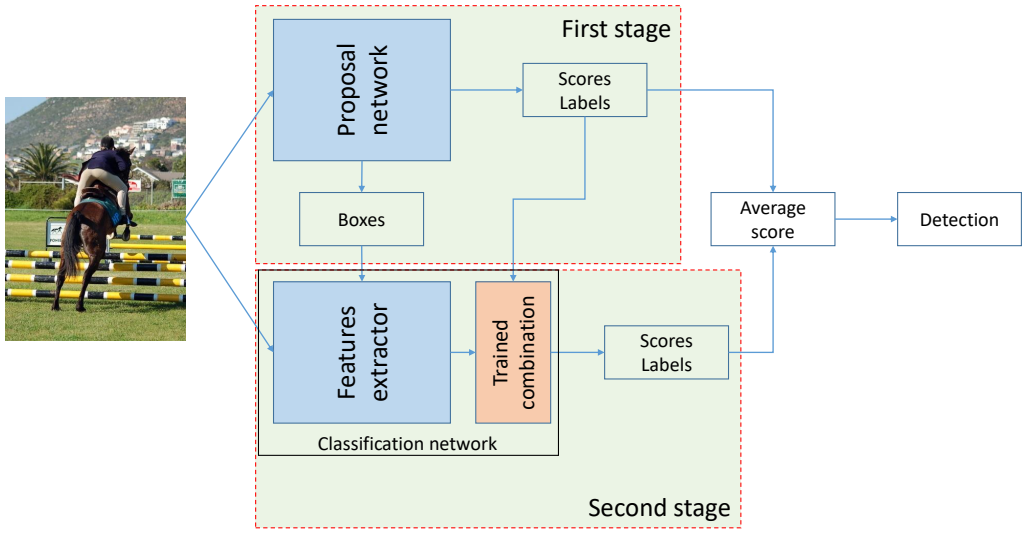
4.1 Two-stage architectures

The two-stage network includes two main components: the proposal network and the classification network. This section describes two proposed model variants. For each, we show the key model design and the use. As mentioned in Subsection 3.2, the two variants use the same basic design. However, the presence of the first-stage confidence score in the second stage is different.

Figure 4.1 shows the proposed architectures. Each model contains two stages: the proposal network and the classification network. An input image is fed to the proposal network (e.g., Faster R-CNN) to generate a set of object proposals inside the image. The outputs of the first stage are the candidate boxes and the corresponding scores for each category of these boxes. The boxes are later used to sample images to fine-tune the classification network (e.g., Inception-V3 Szegedy et al., 2016). In the second stage, we introduce two combination strategies: post-combination and during-training combination (referred to hereinafter as trained combination). The details of combination methods are discussed in Subsection 4.4.



(A) Post-combination network.



(B) Trained combination network.

FIGURE 4.1: Post-combination network and trained combination network. (A) The post-combination network uses the first-stage confidence scores for combination after the second-stage training. (B) The trained combination network uses the first-stage confidence scores for in both the second-stage training and after training.

These designs are inspired by the box-proposal-based object detectors. The "sliding-window-based" method is a well-known approach that requires classification for many sample windows, which are sampled by searching for all possible rectangle insides a given image. For example, a single-scale object detector requires about $10^4 - 10^5$ windows per image for classifying. Thus, the classifier needs to process a large number of samples. Moreover, for

multi-scale detection, the number of windows grows by an order of magnitude. Thus, this strategy leads to inefficient computation. On the other hand, since the appearance of an object in an image is rare, the ratio of positive windows to negative windows is small. This unbalancing not only causes an unstable training process, but also produces many false positive detections. The box-proposal-based method has been widely studied to address this problem. The basic idea is that all objects of interest share common visual properties that the detector can distinguish from the background. A proposal extractor outputs a set of proposal regions in the given image that are likely to contain objects. Comparing to the sliding-window-based method, the number of output rectangles is significantly reduced. This allows the object detector to speed up the window classification and apply more sophisticated classification methods.

Another benefit of the box-proposal-based method is that it enables the use of *number proposal controlling*. The output of the proposal extractor contains both the rectangles containing an object of interest and the scores of those rectangles (also known as regions). These scores can be used for sorting and selecting the proposals. In our proposed method, the output scores are used for subsequent tasks such as training classifier and detection.

The detector is trained with a modified appearance distribution of both positive and negative windows. The missed objects cannot be recovered in the subsequent classification stage. Thus, when using the proposals for object detection, the proposals should cover the whole object of interest in the image. It is common to choose a high recall over the number of proposal windows (or IoU) of ground-truth annotations.

4.1.1 Post-combination network

Figure 4.1a illustrates the architecture of the post-combination network. In this design, the confidence scores of the proposal network are used after training the classification network. We define a post-combination function to connect the confidence scores of the first stage network and the second-stage network. For some proposal networks, such as Faster R-CNN (Ren et al., 2015), the scores of RPN are used to perform NMS on the proposal regions. However, because RPN produces class-agnostic box proposals, the scores are not used for further prediction after training the second stage. Compared to the RPN box proposals, the outputs of the proposal network are significantly reduced (typically, RPN extracts about 300 proposals per image to crop features from a features map, whereas the proposal network extracts about 100 proposals per image to train the classification network.). Because the number of training data is small for the classification network, we fine-tune the data using the pre-trained network.

Regarding object location prediction, because the classification network does not modify object locations, the detected object locations are maintained with proposal network outputs. However, since we re-order the confidence scores, there may be some detections that are eliminated, the detected rectangles may be changed through NMS processing.

4.1.2 Trained combination network

Figure 4.1b shows the trained combination network design. Similar to the post-combination network, it contains two main components: the proposal network and the classification network. The major difference is the appearance of class-specific confidence scores in the classification training process. To perform the during-training combination, we define a trained combination module that inputs confidence scores of the proposal network and concatenates these with the last layers of the classification network.

Classification results often rely on the properties of object appearance in cropped windows. The CNN classifier uses those windows and classes to discover the information needed for classification during training. However, we found that the first-stage confidence scores improved the classification performance by training. For example, given an input window with a high confidence score, there is a high possibility that the classification network predicts the same class as the proposal network. In this case, the combination function is modeled as a trainable function.

Because the input of classification has been changed, the design of the classification network must also be changed. We describe these details in Subsection 4.4. The idea of this design is inspired by a property of confidence scores. We consider the confidence scores vector as a high-level image feature so it can be concatenated with the last layers of the classification network to form a new classifier.

Following the trained combination, a post-combination is performed after detection by the classification network. Although classification scores could be used directly for detection results, our experiment suggested that, despite improving classification performance, the connection between the first stage and the second stage after training is needed to boost overall detection performance. The process of post-combination is similar to the above post-combination model (Subsection 4.1.1).

4.2 Proposal network

This section describes the details of the proposal network. The proposal networks are chosen according to a specific detection task of interest. In the first stage of the proposed model, we choose a proposal network to extract the set of object candidates. We primarily focus on Fast/Faster R-CNN and SSD because many recent methods (He et al., 2017; Cai et al., 2016; He et al., 2016; Lin et al., 2017) are based on these architectures.

4.2.1 Meta architecture

Over the past few years, the deep network has become the top performing method for object detection. The first successful CNN models for object detection were R-CNN (Girshick et al., 2014) and its successor Fast R-CNN (Girshick, 2015). Both R-CNN and Fast R-CNN use a pre-computed proposal

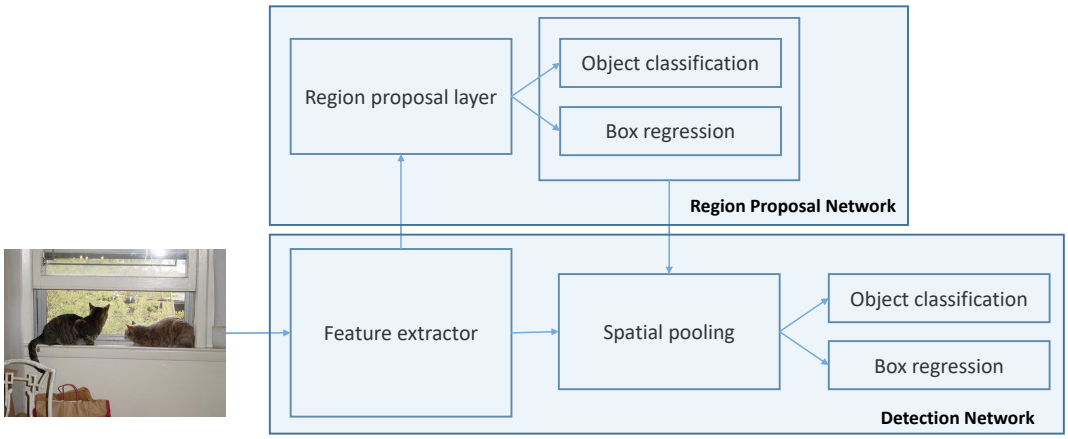


FIGURE 4.2: Proposal network with Faster R-CNN pipeline. The region proposal network and detection network share the extracted image features. The output of RPN is a set of class-agnostic boxes and the output of the detection network is a set of class-specific boxes.

generator. However, modern methods show that it is possible to extract proposals using a deep network such as in the works of Erhan et al., 2014; Ren et al., 2015. These methods typically use a set of boxes in an image with different locations, scales and aspect ratios. These boxes are called "anchors" or "default boxes". Each anchor is used to train the detection network, which predicts two things: a class-specific for that anchor and the offset to shift the anchor to fit the ground-truth. To train these detectors, the network minimizes a loss function which combines a classification loss and bounding box regression loss. An anchor is needed to encode for training by a transform function. For example, in Faster R-CNN, the transform function is modeled as a linear function of cropped features from the image. Such a function can then be learned. The details of anchor box encoding are described in the Appendix C.

The choice of anchor boxes strongly impacts detection performance. In recent works, such as SSD, the anchors (the default boxes in the paper) are regularly sampled across the image at different scales and aspect ratios. The position of each box relative to its corresponding feature map cell is fixed. This approach resembles the sliding window method, but in contrast, it allows the detector to share parameters between different object viewpoint predictions.

In the next sections, we describe the parameter choices for each meta-architecture and adapt the designs according to specific tasks.

4.2.2 Faster R-CNN

In Figure 4.2, the image is processed through two components, RPN and the detection network. The output of RPN (box proposals) is used to crop the features from the intermediate feature map. These cropped features are fed to the remainder of the feature extractor for class and box-refinement prediction.

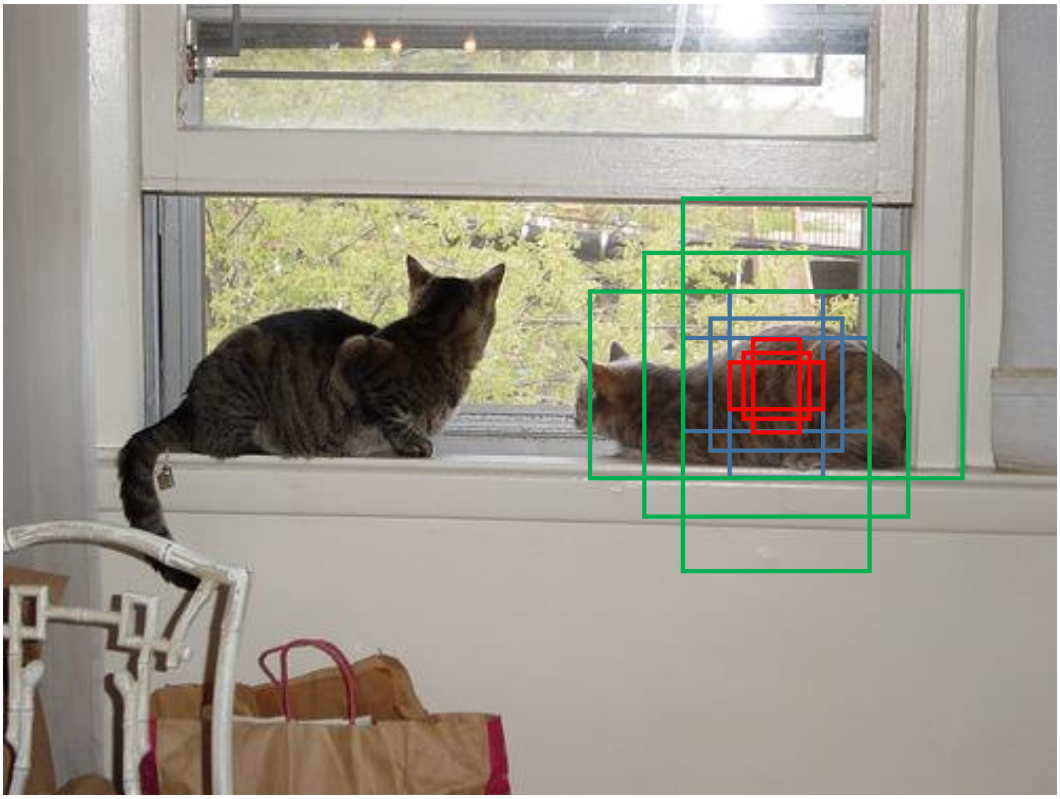


FIGURE 4.3: Anchor box settings for each sliding window. The anchor is located at the center of the window. We use 3 scales ($128 \times 128, 256 \times 256$ and 512×512) and 3 aspect ratios ($1:1, 1:2$ and $2:1$) to create 9 anchor boxes for each sliding window.

Anchor boxes

The anchor boxes are taken at each sliding window location across the image. We keep the default configuration of the anchor as in Faster R-CNN. There are 9 anchors at a position in an image. Figure 4.3 gives an example of the anchor boxes. In our setting:

- Each color of box represents a box size: $128 \times 128, 256 \times 256$ and 512×512.
- For each colored box, the ratio of height and width is one of the following $1:1, 1:2$ or $2:1$.

The number of the anchor position (the center point of anchor) depends on the sliding window stride. The selection of aspect ratio depends on the task of interest. For example, with people detection, the object rarely appears within a very short bounding box. The above aspect ratios and sizes work well with the Pascal VOC and COCO datasets. The anchor design has the important property of *translation invariance*. That is, the detector is able to predict the object proposal independent of the object location in a given image. Another advantage of the anchor design is that it presents a scheme for multi-scale object detection. Compared with image/feature pyramids object detection, the *pyramid of anchors* is more cost-efficient. The *pyramid of anchors*

is built on a single scale of images and feature maps, and it uses a single size of filters.

Baselines

In this thesis, a network *baseline* (or network *backbone*) is a part of the network that is reused to construct the feature extractors. We use a high-quality feature extractor, such as Resnet-101 (He et al., 2016) or Inception-Resnet-V2 (Szegedy et al., 2017), which are used in state-of-the-art ImageNet ILSVRC 2012 classification and detection tasks, as a baseline.

In the Resnet-101 network, we extract features from the last layer of the conv4 block. The Region of Interest (RoI) is cropped to 14×14, and a max-pooling layer is then used to reduce the feature size to 7×7. Intuitively, using a smaller window stride should give better results; thus, the output stride of Resnet-101 network is modified to 16 (the original output stride of Resnet-101 is 32). This modification is done by changing the stride of conv5_1 layer from 2 to 1.

Inception-Resnet-V2 mixes the residual design with Inception networks (Szegedy et al., 2015). We implement Inception-Resnet-V2 on TensorFlow (Abadi et al., 2016) and extract features from the Mixed_6a layer. The features are cropped by the RoI, resized to 17×17 and maxpooled with a stride of 1.

Box regression

The object detection network needs to simultaneously predict the object classes and object locations. Since the object classes are classified by predicting a discrete class label though a classifier (e.g.,CNN-softmax, CNN-SVM), the box locations are predicted by estimating a continuous quantity. The box locations are encoded to bounding box targets before training. We describe the box encoding method in Appendix C.

Typically, one training bounding box (x, y, w, h) is encoded into a bounding box target t as follows:

$$\begin{aligned} t_x = (x - x_a)/w_a, \quad t_y = (y - y_a)/h_a \\ t_w = log(w/w_a), \quad t_h = log(h/h_a) \end{aligned} \tag{4.1}$$

Similarly, a ground-truth box (x_a, y_a, w_a, h_a) is encoded as follows:

$$\begin{aligned} t_x^* = (x^* - x_a)/w_a, \quad t_y^* = (y^* - y_a)/h_a \\ t_w^* = log(w^*/w_a), \quad t_h^* = log(h^*/h_a) \end{aligned} \tag{4.2}$$

The box regressor minimizes the loss function

$$L_{box}(t, t^*) = Smooth_{L_1}(t - t^*) \tag{4.3}$$

where

$$Smooth_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \tag{4.4}$$

Region of Interest Pooling

After RPN, the proposal boxes are different sizes, resulting in different-sized feature maps. In order to train the next stage, the RoI pooling (or spatial pooling) layer is used to extract feature maps as the same size. The layer has two inputs:

- Fixed size feature maps obtained by the feature extractor from previous layers. Note that the feature maps are shared between RPN and the detection network instead of using another separate feature extractor. The shared feature maps not only significantly reduce the computation cost, but also enable joint training between the model components.
- A matrix of size $N \times 5$, where N is the number of RoI, and four coordinates, which represent the proposal regions and an image index.

The dimension of the RoI pooling output does not depend on the size of region proposals nor the size of input feature maps. The RoI pooling layer can improve the processing speed by using same input-computed feature maps for all proposals in an image while maintaining high detection accuracy.

There are several implementations of RoI pooling. The first idea is the work of R-CNN, which pools the portion of the feature map inside the proposal into a fixed-size feature. However, the network needs to forward each proposal and write the output feature to the disk for further use without sharing the feature maps. SPPNet (He et al., 2014) addresses this problem by sharing computations, after which, multiple output features are concatenated as in spatial pyramid pooling. Fast R-CNN uses a simpler but higher detection quality approach where the method uses only a one-level pyramid level. Other RoI pooling methods include Spatial Transformer Networks (Jaderberg, Simonyan, Zisserman, et al., 2015), differentiable cropping (Dai, He, and Sun, 2016), and the attention model (Gregor et al., 2015). In this thesis, we use the "crop and resize" layer in Tensorflow (Abadi et al., 2016), which uses bilinear interpolation to crop an image feature map into a fixed-size feature. Unlike the Fast/Faster R-CNN method, we disable the back-propagation process with respect to box coordinates due to instability during training.

Training details

Ren et al., 2015 proposed four-step alternating training that first trains RPN and uses the proposals to train Fast R-CNN. The Fast R-CNN network is then used to initialize RPN, and the process is repeated. Instead of using four-step alternating training, we adopt an end-to-end joint training of the RPN and Fast R-CNN components for convenience. The multi-task loss on each RoI is the sum of the cross-entropy loss of classification and the box regression loss $L(p, u, t, t^*) = L_{cls}(p, u) + \lambda[u \geq 1]L_{box}(t, t^*)$, where $L_{cls}(p, u) = -log(p_u)$ is the cross-entropy loss for true class u. The tuple $t = (t_x, t_y, t_w, t_h)$ is the predicted bounding box regression offsets, and $t^* = (t_x^*, t_y^*, t_w^*, t_h^*)$ represents the ground-truth. The indicator function $[u \geq 1]$ is equal to 1 when $u \geq 1$,

otherwise 0. In our experiment, we set the balance loss weight $\lambda = 1$. We use the NMS with a threshold of 0.7 IoU and 0.6 IoU for RPN and Fast R-CNN, respectively.

4.2.3 Single Shot MultiBox Detector (SSD)

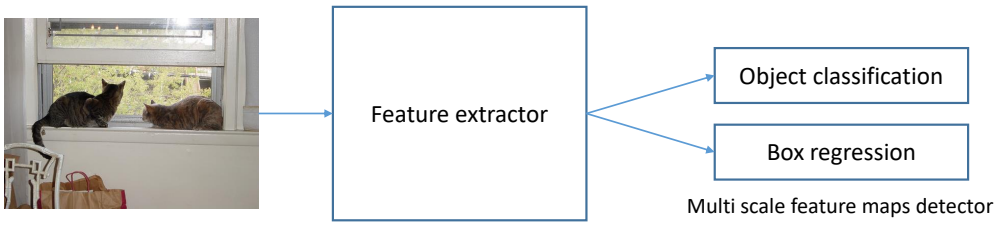


FIGURE 4.4: Proposal network with SSD pipeline.

For training, the SSD model only needs an input image and available ground-truth bounding boxes (manually labeled bounding boxes that specify objects in the image). The pipeline of SSD is depicted in Figure 4.4. The image is fed to a feature extractor network (e.g., VGG16) and the detection network predicts the object classes and bounding boxes directly from the generated anchors (default bounding boxes). The detection network adds more convolutional layers with the spatial resolution reducing by a factor of 2. It performs the prediction at each additional layer.

Compared with two-stage detectors, the model presents a different approach to detecting objects using a single-stage network that directly predicts classes and anchor offsets without requiring second-stage object class prediction. Advantages of this design include:

- Multi-scale feature maps for detection: the additional layers at different feature map locations and the different sizes of these layers allow the SSD to predict an object at multiple scales.
- SSD runs faster than two-stage networks because it does not require RPN for box proposals.

Details of design

With the SSD model, we use VGG16 (Simonyan and Zisserman, 2014) pretrained by the ImageNet (Deng et al., 2009) dataset as the baseline network, which is a simple and widely-used design. Figure 4.5 depicts the details of the SSD model using the VGG16 baseline. The VGG16 baseline network is modified by converting the fc6 and fc7 layers to convolutional layers, which also reduces the number of outputs to 1,024. All dropout layers (Srivastava et al., 2014) and the fc8 layer are removed from the original VGG16 network. We add five additional output layers (conv6_2, conv7_2, conv8_2, conv9_2, and conv10_2) to predict the locations and scores of objects. The model is then finely tuned using the COCO dataset (Lin et al., 2014) for the COCO test-dev detection task.

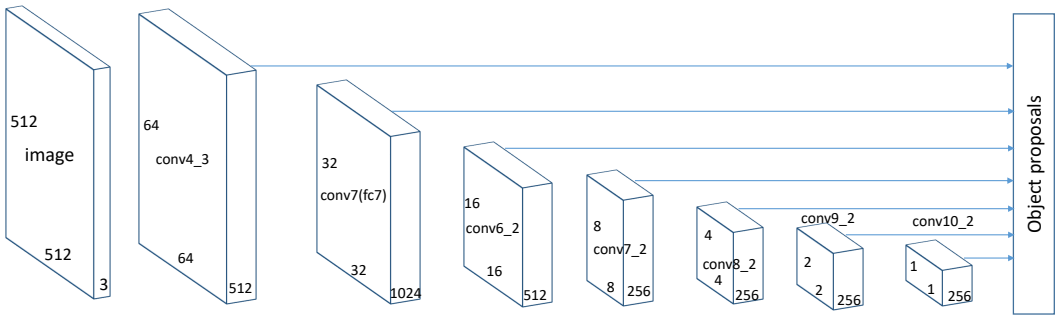


FIGURE 4.5: VGG.

TABLE 4.1: Output layer and anchor settings for the VGG16 baseline. Aspect ratio is the ratio of the width to the height of an anchor.

Output layer	Aspect ratio (w/h)	min	max
conv4_3	1/2, 1, 2	20.5	51.2
conv7	1/3, 1/2, 1, 2, 3	51.2	133.1
conv6_2	1/3, 1/2, 1, 2, 3	133.1	215.0
conv7_2	1/3, 1/2, 1, 2, 3	215.0	297.0
conv8_2	1/3, 1/2, 1, 2, 3	297.0	378.9
conv9_2	1/3, 1/2, 1, 2, 3	378.9	460.8
conv10_2	1/2, 1, 2	460.8	542.7

Anchor boxes

Generating anchor boxes in SSD is similar as in Faster R-CNN. However, the anchor boxes are generated at every output of the detection network. This multiple-level anchor generation allows the network to be set up at different scales and aspect ratios to achieve different outputs. Thus, the anchor size does not need to correspond to the receptive field size of each layer. Table 4.1 shows the details of the aspect ratio setting for each output layer where *min* and *max* are the values used to compute anchor box size. Following SSD, the scale of anchor boxes for each output layer is computed as

$$s_k = s_{min} + \frac{s_{max} - s_{min}}{m - 1}(k - 1), \quad k \in \{1, ..., m\} \tag{4.5}$$

where m is number of output layers, $s_{min} = 0.2, s_{max} = 0.9$. An anchor with scale $s' = \sqrt{s_k s_{k+1}}$ is also added to list of anchor boxes.

The anchor boxes cover a large number of object sizes and shapes by combining the predictions of all output layers.

Training details

Similar to the Faster R-CNN model, in this SSD proposal network detection, we train the network by minimizing the following loss function, which is the

sum of the cross-entropy loss of classification and the box regression loss:

$$L(p, u, t, t^*) = \frac{1}{N}(L_{cls}(p, u) + \lambda[u \geq 1]L_{box}(t, t^*)) \tag{4.6}$$

where N is the number of matched anchor boxes. In the experiment, the weight λ is set to 1.

4.3 Classification network

4.3.1 Baseline networks

After extracting the set of object proposals from an image, the number of false positive detections is still large because of the sliding window scheme. We use these proposals to train the second stage as a classifier. In the experiment, we use several classifiers, such as Resnet-101 (He et al., 2016), Inception-V1 (Szegedy et al., 2015) and its successor Inception-V3 (Szegedy et al., 2016). All of these classifiers are pre-trained on ImageNet, which contains 1.2 million training images. The details for each baseline network are described as follows:

- **Inception-V1**: We remove the final fully-connected layers (the final *logits* (classifier) layers) of the original Inception-V1 network. The input size of Inception-V1 is 224×224. We add three fully-connected layers at three network outputs (one for prediction and two for auxiliary classifiers at intermediate layers). We do not use auxiliary branches of Inception-V1 to fine-tune the classification network.

- **Resnet-101**: We faithfully follow the design of Resnet-101, which applies batch normalization after every convolutional layer. We use the feature extraction after the network's *block4*. The input size is 224×224.

- **Inception-V3**: Since Inception-V3 is an update of Inception-V1, we continue to use the auxiliary logits after the *Mixed_6e* layer. The network input size is 229×229.

The pre-trained model weights only affect the initialization of the model. For small datasets, the model typically trains a subset of layers to avoid overfitting. In the experiments, by sampling multiple proposals per ground-truth, the classification network is able to fine-tune all layers.

4.3.2 Data sampling

Although the second stage is trained using the same dataset as the proposal network except for the Caltech pedestrian dataset (Dollar et al., 2012), the method of sampling data is different from the first stage. We first eliminate some "bad" proposals (boxes with very low confidence scores or boxes that are too small). Each proposal is then matched with ground-truth boxes. We

calculate the IoU of these object proposals with the available ground-truth in an image. A proposal p is a positive sample if $\alpha^* \geq 0.5$, where

$$\alpha^* = \max_{g_i \in G} IoU(p, g_i) \tag{4.7}$$

and $G = \{(g^i, l^i)\}_{i=1}^{M}$ is the set of ground-truth bounding boxes where $g^i = (g^i_x, g^i_y, g^i_w, g^i_h)$ specifies the top left coordinates of the ground-truth box together with its width and height, l^i is the ground-truth class, and M is the number of ground-truth boxes in the given image. Otherwise p is a negative sample if $\alpha^* < 0.5$. In the case of a positive sample, the class of proposal p is l^{i^*} where

$$i^* = \underset{i \in \{1,\ldots,M\}}{\operatorname{argmax}} IoU(p, g^i) \tag{4.8}$$

In order to train a more stable classification network, we maintain a constant ratio between the number of positive samples and the number of negative samples. (In our experiments, the ratio of positive to negative samples is 1:3.)

4.3.3 Preprocessing

The classification network is fine-tuned from a pre-trained model. The proposal regions from the first stage are used to crop the input images, and then resized using bilinear interpolation to the size of 256×256 for Inception-V1 and Resnet-101, and 340×340 for Inception-V3. We then randomly crop to network input size for training (at the test time, we use center cropping). The training samples are randomly flipped horizontally[1] with 0.5 probability, and are subtracted from the dataset image means.

In the experiment, we preserve the aspect ratio image resizing before randomly cropping as in AlexNet (Krizhevsky, Sutskever, and Hinton, 2012). However, we observed that training the classification network was unstable. We think this might be due to the variables of aspect ratios from extracted proposals.

4.4 Combination functions

As shown in Figure 4.1, in the proposed model architectures, our combination module uses post-combination and trained combination. These combination procedures are key techniques to strengthen detection by using an object detector as a proposal network.

[1]The intuition behind flipping an image is that an object should be equally recognizable as its mirror image (in the left/right direction). Columns are preserved, but appear in a different order than before.

4.4.1 Post-combination

The post-combination network is depicted in the Figure 4.1a. In this approach, the combination occurs after the training of two stages. The classification network outputs the scores of object proposals. The final detections can use either the proposal scores or the classification scores. However, we observed that standalone classification scores do not improve detection performance. For large and clear objects, the proposals are usually well-detected, and the detection scores are high. On the other hand, deep-learning-based detectors tend to be weak for small objects due to insufficient resolution of feature maps for detecting small instances (Zhang et al., 2016). In this case, the output detection scores are usually low. To address this problem, we propose a method to maintain good detections and enhance weak detections by combining two detection scores. For each output of the proposal network $(\mathcal{L}, s_p)$, $\mathcal{L} = (x_1, x_2, y_1, y_2)$ is the location of the box, $s_p = (s_{p_0}, ..., s_{p_N})$ is the score of the proposal network, and $s_c = (s_{c_0}, ..., s_{c_N})$ is the score of the classification network. Further, $i^* = \text{argmax}_{i \in \{0,...,N\}} s_{p_i}$ and $j^* = \text{argmax}_{j \in \{0,...,N\}} s_{c_j}$ are the detected classes of the proposal network and the classification network, respectively. The final detection of the combined network is $(\mathcal{L}, s_{i^*}, i^*)$ with

$$s_{i^*} = f(s_p, s_c) \tag{4.9}$$

where f is a combination function. We evaluate several combination functions. The combinations are based on the value of two scores and the agreement of two detection classes. We consider a mean function (f_1) and a multiplication function (f_2), defined as follows:

$$f_1(s_p, s_c) = \frac{(s_{p_{i^*}} + s_{c_{i^*}})}{2} \tag{4.10}$$

and

$$f_2(s_p, s_c) = (s_{p_{i^*}} * s_{c_{i^*}}) \tag{4.11}$$

and we define f_3 as follows:

$$f_3(s_p, s_c) = \begin{cases} f_1(s_p, s_c * c) & \text{if } i^* = j^* \text{ and } s_{p_{i^*}} < d \\ f_1(s_p, s_c) & \text{otherwise} \end{cases} \tag{4.12}$$

where $c > 1$ is the boosting weight and $d < 1$ is the high-score threshold.

The key idea behind f_3 is to encourage detections with the same detected classes that have higher confidence scores between the proposal network and the classification network. However, the scores of the proposal network for these cases are not good enough (i.e., they are lower than the threshold score d). Note that, to increase the final score, it is possible to boost the first-stage score (s_p), but we found that the experimental results of boosting the first-stage score are worse than that of boosting the second-stage score (s_c).

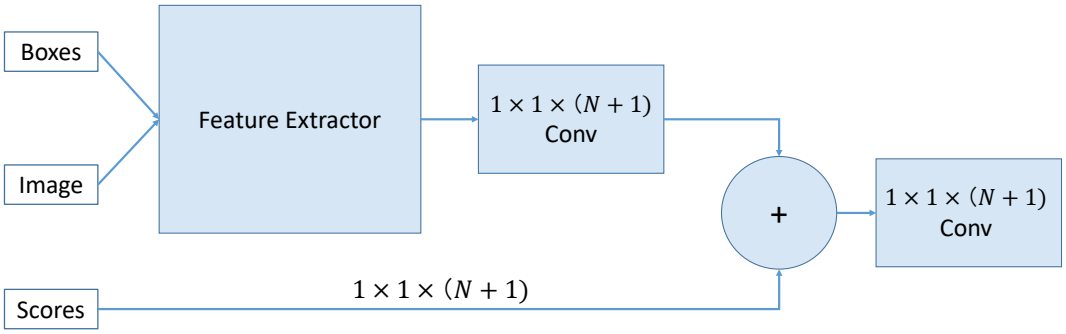


FIGURE 4.6: Classification network with trained combination. The output is an $(N+1)$-way classification layer where N is the number of interest categories and plus 1 indicates the background.

4.4.2 Trained combination

In this approach, the combination occurs both after training the classification network and during the training classifier process. Given a proposal $P = (P_x^1, P_y^1, P_x^2, P_y^2)$, the output of the last layer in the classification network is denoted by $\phi_l(P)$. By adding the new combination module in Figure 4.1b, the output of the classification network becomes $\hat{\mathbf{w}}_\star(\phi_l(P), s_p)$, where s_p is the score vector, and $\hat{\mathbf{w}}_\star$ are trainable parameters. In this case, for a true class u, the training loss is

$$L_{cls}(p, u) = -log(p_u) \tag{4.13}$$

where

$$p = (p_0, ..., .p_N) = f_{cls}(\hat{\mathbf{w}}_\star(\phi_l(P^i), s_p)) \tag{4.14}$$

is computed as softmax (f_{cls}) over the $(N+1)$ outputs of the last fully-connected layer. After the second-stage training, we apply the average score to generate the final score.

The implementation of the combination module is depicted in Figure 4.6. We construct a $1 \times 1 \times (N+1)$ vector from the score of the proposal network, where N is the number of interest categories and plus 1 indicates the background. This vector is concatenated with extracted features from the last layer of the feature extractor network, and is then connected to a fully-connected layer. The computation of this new trained combination module is nearly cost-free in comparison to training the original classification network.

4.4.3 Implementation details

The new fully-connected layers are initialized with Xavier initializer (Glorot and Bengio, 2010), which assigns weights by a Gaussian distribution and ensures the same variance for inputs and outputs. We continue using the stochastic gradient descent (SGD) to train CNN parameters. The base learning rate and learning rate decay policy are set according to the specific task, as will be detailed in Subsection 4.5.

We keep a maximum of 100 samples per image for training, where samples are selected base on their detection scores by the proposal network.

TABLE 4.2: Evaluation settings for the Caltech benchmark.

	Height (pixels)	Occlusion level
All	$20 - inf$	$0 - 80\%$
Reasonable	$50 - inf$	$0 - 35\%$
Near scale	$80 - inf$	0%
Large scale	$100 - inf$	0%
Medium scale	$30 - 80$	0%
Far scale	$20 - 30$	0%
No occlusion	$50 - inf$	0%
Partial occlusion	$50 - inf$	$1 - 35\%$
Heavy occlusion	$50 - inf$	$35 - 80\%$
Small objects	$30 - 50$	$0 - 35\%$

Some samples highly overlap with each other. To avoid redundancy, we adopt NMS on the sample regions based on their scores. An IoU threshold of 0.7 is fixed for NMS.

4.5 Experimental results

The proposed method should be tested carefully on appropriate datasets to prove its potential for real-world applications. In this section, we present experimental results for three object detection datasets: Caltech pedestrian (Dollar et al., 2012), Pascal VOC (Everingham et al., 2010), and COCO (Lin et al., 2014).

While the experiments mainly showed the benefit of using a two-stage network detector, we also obtained results that were comparable to those obtained using other cutting edge detection methods. We denote the two-stage model used in various experiments as follows: SSD+Inception-V1+M indicates that the proposal network is SSD, the baseline of the classification network is Inception-V1, and the method of combination is multiplication score f_2 (M). Other notation of the combination method includes the mean function f_1 (A), the combination with threshold function f_3 (T), and the trained combination (TC).

4.5.1 Caltech pedestrian detection

We first apply the proposed models to a single-category dataset, namely, the Caltech pedestrian dataset. The dataset consists of approximately 1,900 individual pedestrians, which are annotated with approximately 350,000 ground-truth annotations. The number of images in the test set is 4,024. The Caltech evaluation benchmark uses the log-average miss rate to summarize the detector performance. The average miss rate is computed by averaging the miss rate for nine false positives per image (FPPI) in log space from 10^{-2} to 10^{0}. Evaluation is performed using several different settings, listed in Table 4.2, based on the height and occlusion level of pedestrians.

TABLE 4.3: Miss rates of detection using the proposal network (SSD 512) and the classification network, and combination scores for the Caltech test set.

	All	Reasonable	Near	Medium	Far	No occ.	Partial occ.	Heavy occ.	Small objects
Proposal network (SSD Liu et al., 2016)	57.74	17.56	1.47	43.11	79.61	15.59	27.89	69.02	48.86
Classification network (Inception-V1)	55.38	17.63	1.59	38.27	77.71	15.53	30.34	70.08	45.07
SSD+Inception-V1+A	52.02	14.59	1.44	34.77	77.26	12.74	26.01	68.34	42.05
SSD+Inception-V1+M	52.30	14.97	1.44	35.33	77.59	13.11	**25.66**	67.85	42.21
SSD+Inception-V1+T	**51.70**	**13.89**	**1.15**	**34.30**	**74.92**	**11.75**	27.29	**66.59**	**41.56**
SSD+Inception-V1+TC	55.11	15.28	1.33	38.64	79.67	13.08	28.19	68.62	47.06

In the work of Zhang et al., 2016, it is argued that, despite particularly successful general object detection, Faster R-CNN (as a stand-alone detector) has limited success for pedestrian detection. For this reason, we use an SSD with a VGG16 baseline as the proposal network for this experiment. In the network settings, the aspect ratios of the default bounding boxes (the anchors) are $\{1/3, 1/2, 1, 2, 3\}$. For output from the conv4_3 layer, because of the different feature scales, an $L2$ normalization layer (Liu, Rabinovich, and Berg, 2015) is added to scale down the feature norm to 20.

To fine-tune the proposal network, we used a pre-trained SSD from the COCO dataset because the COCO dataset is more similar to the Caltech dataset than ImageNet (Deng et al., 2009). Since Caltech is a relatively small dataset, we adopt a data augmentation strategy by adding more training images from other pedestrian datasets: the KITTI dataset (Geiger, Lenz, and Urtasun, 2012), the TUD-Brussels dataset (Wojek, Walk, and Schiele, 2009), and the ETH pedestrian dataset (Ess et al., 2008). This addition of data increases the number of images by 26% compared to the Caltech dataset alone. We trained a model with a base learning rate of 10^{-4}, a momentum of 0.9, and a weight decay of 0.0005. The total number of training iterations is 240,000. We used Inception-V1 (Szegedy et al., 2015) to classify pedestrian candidates extracted from the proposal network, where the initial learning rate was 10^{-4}.

We compare the performance of the proposal network (Liu et al., 2016), the classification network (Szegedy et al., 2015), and combinations of the two networks. Table 4.3 shows the experimental results. The classification network performs better than the proposal network for the "all", "medium", and "small objects" settings by 2.36%, 4.84%, and 3.79%, respectively. However, the performance of the classification network is inferior for "near" objects. This indicates that Inception-V1 is more robust than SSD for small pedestrian classification. Moreover, the performance of the classification network for occluded pedestrians ("partial occ." and "heavy occ.") is worse than the proposal network. This might be because the SSD model's classification task is supported by the box prediction task, which allows the model to better predict the objects of unusual aspect ratio (occluded objects). The results in Table 4.3 show that the miss rates for every setting are reduced for all combination functions compared with the proposal network. With combination function f_3, the performance is slightly improved (0.32%) for the near scale and significantly improved for smaller objects. The miss rate reductions for the "small objects" and "medium" settings are 7.30% and 8.81%, respectively. Overall, the mean combination f_1 performs slightly better than the

TABLE 4.4: Miss rates of detection using the proposed method and state-of-the-art pedestrian detection methods on the Caltech test set. The proposed model outperforms the other methods for "small", "far","near" and "large" pedestrians.

	All	Reasonable	None	Partial	Heavy	Near	Medium	Far	Medium	Small	Large
LDCF++	67.24	14.98	12.76	33.11	75.74	5.25	58.46	100	58.46	83.22	2.15
RPN+BF	64.66	9.58	7.68	24.23	74.36	2.26	53.93	100	53.93	79.83	1.18
MS-CNN	60.95	9.95	8.15	19.24	59.94	2.6	49.13	97.23	49.13	70.34	1.99
F-DNN	**50.55**	**8.65**	7.10	**15.41**	**55.13**	2.96	**33.27**	77.47	**33.27**	44.86	1.70
Our method (T)	51.73	14.07	11.89	27.36	66.56	**1.15**	34.44	**74.95**	34.44	**41.43**	**0.00**

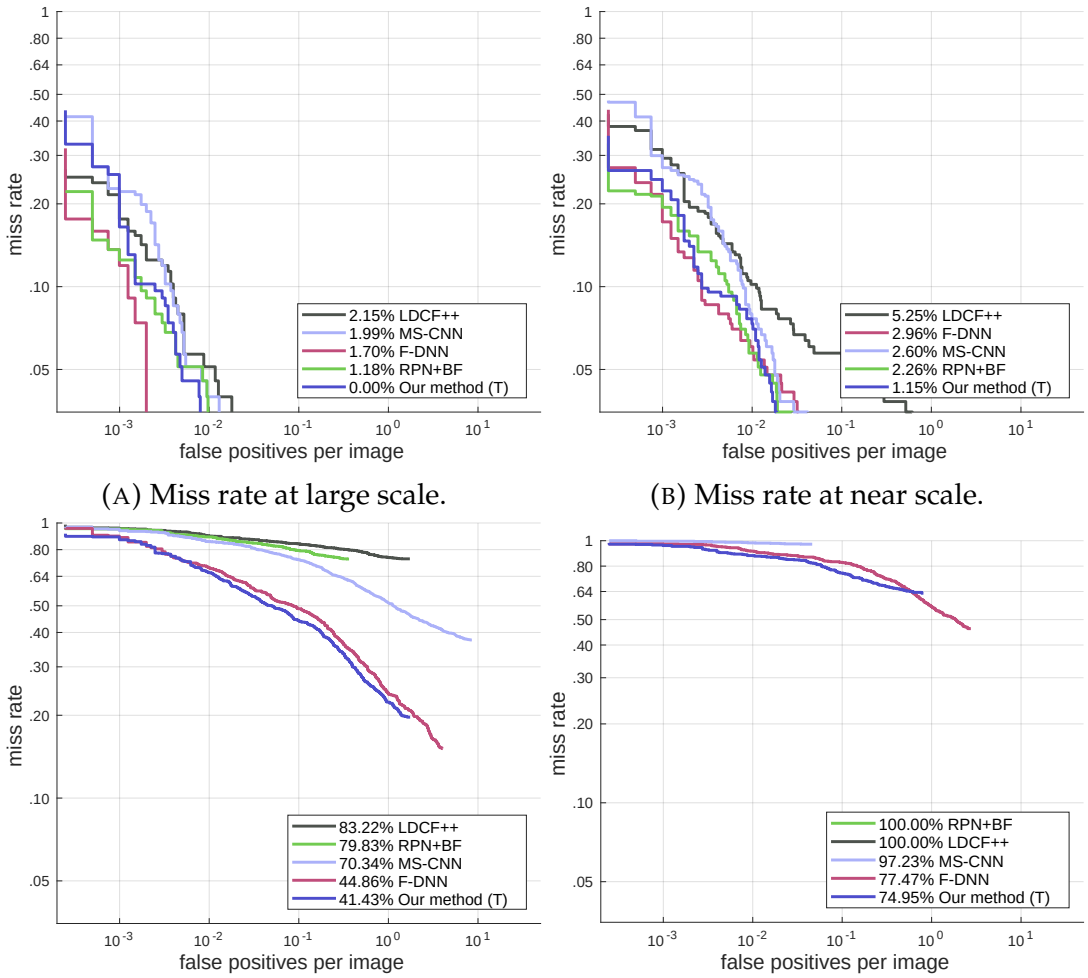
multiplication combination f_2.

We also compared the proposed model with state-of-the-art pedestrian detection methods in Table 4.4. For the overall configuration ("all" setting), our proposed method outperforms MS-CNN (Cai et al., 2016) method and is close to the result of F-DNN (Du et al., 2017), which uses pixel-wise semantic segmentation for reinforced detection. However, our method is simpler and faster than F-DNN, which requires 2.48 sec per image (Du et al., 2017). For large-size object detection ("large" and "near" settings), the two-stage network is the most accurate with respective miss rates of 0.00% and 1.15%. For small-size object detection ("small" and "far" settings), the proposed method also outperforms all the state-of-the-art methods by a large margin. For instance, the miss rate at "small" setting is 41.43%, which is 3.43% better than the next most accurate method (F-DNN). At the "far" setting, some methods, such as RPN+BF (Zhang et al., 2016) and LDCF++ (Ohn-Bar and Trivedi, 2016), were unsuccessful in detecting pedestrians (miss rate of 100%). Our method had the smallest miss rate (74.95%), which indicates the effectiveness of the proposed method in detecting small-size pedestrians.

Note that a miss rate of 0.00% does not mean that the method is perfect. In Figure 4.7, we show the receiver operating characteristic (ROC) curves on the Caltech test set at different settings. The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. However, performance is assessed as the average miss rate at nine false positives per image (FPPI). The ROC curve is used rather than the precision-recall curve because in certain applications, such as auto-driving, there is an upper limit on the acceptable FPPI rate independent of pedestrian density (Dollar et al., 2012).

Detection examples

Figure 4.8 shows some detection examples using the Caltech pedestrian detection test set. We used the combination with threshold to train data using several pedestrian datasets, including KITTI, TUD-Brussels, ETH pedestrian and Caltech itself. Images show the bounding boxes and detection scores (normalized). We select images with pedestrians in various scales. Some images contain crossing pedestrians for easy viewing, people riding bicycles are annotated as pedestrians. Note that in the Caltech pedestrian dataset, most frames contain very few people.



(A) Miss rate at large scale.

(B) Miss rate at near scale.

(C) Miss rate at small scale.

(D) Miss rate at far scale.

FIGURE 4.7: Caltech test ROC. We test the model with different evaluation settings. The figure shows the average miss rate of the proposed model and state-of-the-art pedestrian methods.

4.5.2 Pascal VOC object detection

We used the Pascal VOC (Everingham et al., 2010) dataset to evaluate the proposed method. The Pascal VOC 2007 test set has 4,952 images belonging to 20 categories for the object detection task. We performed data augmentation by adding Pascal VOC 2007 trainval set and Pascal VOC 2012 trainval set (referred to herein as Pascal VOC 2007+2012), resulting in approximately 16k images for training. We chose the Faster R-CNN architecture to extract the set of object proposals. We first fine-tuned the proposal network on the COCO dataset (Lin et al., 2014) for 80 categories of the object detection task. The detector was then fine-tuned on Pascal VOC 2007+2012 and tested on the Pascal VOC 2007 test set.

During training, we used an SGD (Krizhevsky, Sutskever, and Hinton, 2012) optimizer with a batch size of 1, a momentum of 0.9, and a weight decay of 0.0005. The images were resized to $600 \times 1,024$, and other preprocessing steps, such as means subtraction and random flipping, were also applied.

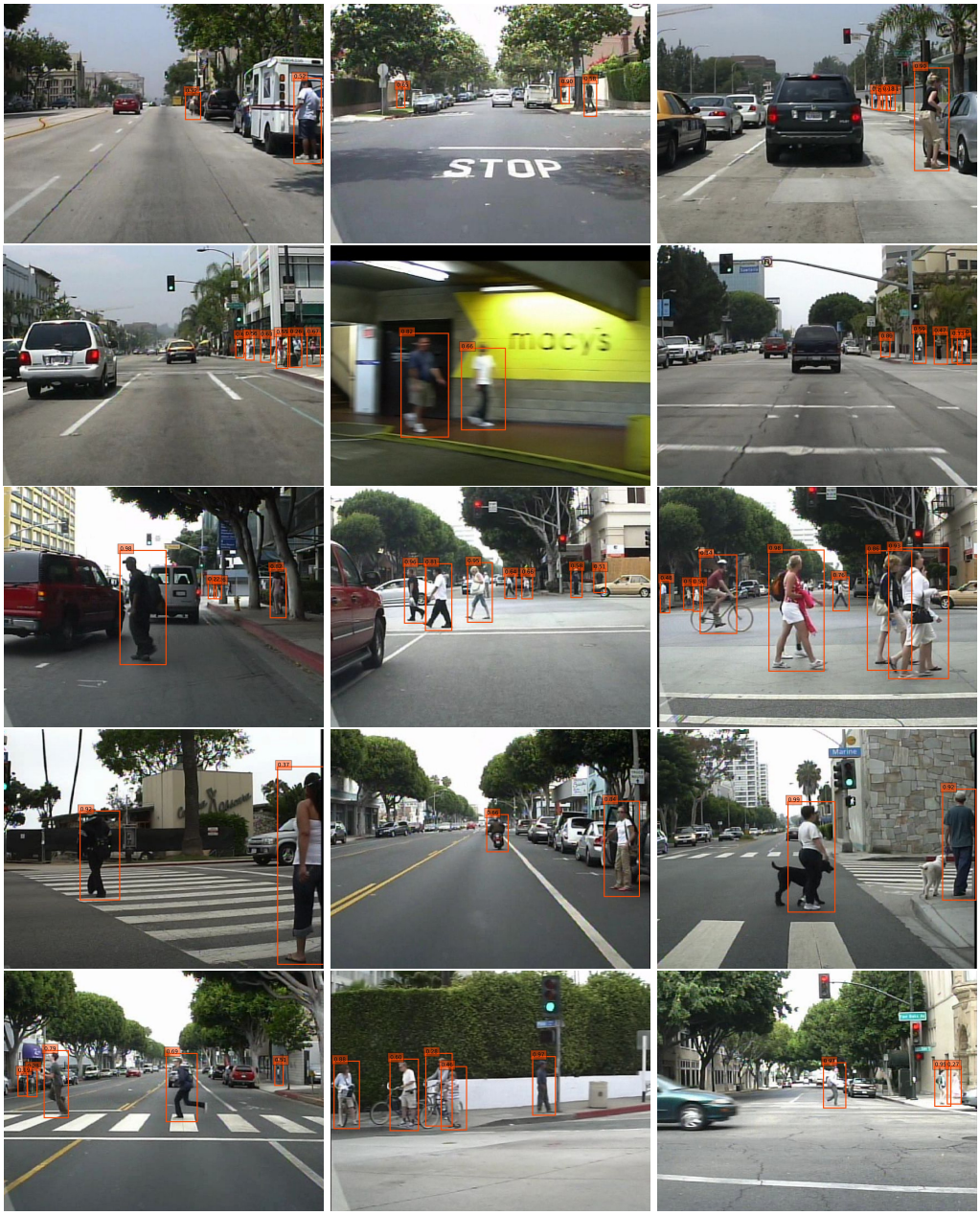


FIGURE 4.8: Detection examples on the Caltech test set. Detected pedestrians with bounding boxes and score (normalized). Note that our approach can detect pedestrians in some difficult conditions, such as small scale (first row) or blurred pedestrians (2^{nd} row, 2^{nd} column).

With a base learning rate of 0.003, we trained the proposed model for 200,000 iterations.

Table 4.5 shows the results for the Pascal VOC 2007 test set. We use Faster R-CNN as the proposal network. In this test, the baseline is Resnet-101 (He et al., 2016) (pre-trained on ImageNet), and the second stage uses Resnet-101 as the classification network. We observed that the combination methods improved detection performance for all classes. The two-stage model with trained combination has a better mean average precision (mAP) than the

TABLE 4.5: Individual average precision (%) on the Pascal VOC 2007 test set. All models were trained with the COCO dataset and then fine-tuned with the Pascal VOC 2007+2012 training set. We then retrain Faster R-CNN[1] (Ren et al., 2015) (with Resnet-101 as the baseline) to fine-tune the classification network.

Group	Class	Faster R-CNN[1]	A	M	T	TC
vehicles	aero	80.0	81.0	81.0	81.0	**81.3**
	bike	80.2	80.2	80.2	80.3	**80.6**
	boat	67.7	70.5	70.6	70.7	**72.8**
	bus	79.7	81.1	81.2	**81.3**	81.2
	car	86.6	**88.7**	**88.7**	88.6	88.6
	mbike	79.3	79.9	79.8	**80.2**	80.1
	train	78.6	78.7	78.7	78.7	**79.2**
animals	bird	77.2	78.3	77.9	78.3	**78.8**
	cat	87.0	87.5	**88.4**	87.4	87.0
	cow	83.5	86.3	85.8	86.2	**86.7**
	dog	84.6	86.0	86.3	86.0	**86.9**
	horse	86.8	87.3	87.6	87.6	**88.3**
	sheep	75.7	**78.9**	78.6	78.8	**78.9**
	person	77.1	78.6	78.7	78.8	**79.9**
furniture	bottle	65.4	68.3	68.2	67.9	**69.1**
	chair	62.4	63.5	63.9	63.9	**64.7**
	table	73.3	74.0	73.9	74.2	**74.8**
	plant	40.3	43.5	43.5	43.7	**46.5**
	sofa	77.9	80.8	80.5	81.3	**82.6**
	tv	69.6	71.0	71.2	71.1	**71.4**
	mAP	75.6	77.2	77.2	77.3	**78.0**

proposal network (Faster R-CNN) by 2.4%. Comparing different combination methods, the mean (A) and multiplication (M) combinations are almost equal with mAP of 77.2%. The combination with threshold function (T) performs slightly better than the mean and multiplication functions. The trained combination method has better results than other combination methods for most classes and its overall result is a mAP of 78.0%. The proposed method shows large improvements for difficult object classes, such as plant, with a maximum mAP gap of 6.2%. This indicates that training the classifier with the additional output from the proposal network (the detection scores) improves the performance for small objects by a good margin.

Table 4.6 shows the results of the second test with Faster R-CNN, which has a better baseline Inception-Resnet-V2, and uses Inception-V3 as the second-stage network. Because the Inception-Resnet-V2 baseline is more accurate than the Resnet-101 baseline in ImageNet ILSVRC, the performance of the proposal network using Inception-Resnet-V2 is also better than that using Resnet-101 as the baseline. The combination with threshold and trained combination achieved the highest performance for most classes. With trained combination, the mAP surpasses Faster R-CNN by 1.2%. Some categories are greatly improved (e.g., plant: 2.4%, bottle: 2.5%, boat: 3.4%). As in the

TABLE 4.6: Individual average precision (%) on the Pascal VOC 2007 test set. All models are trained with the COCO dataset and then fine-tuned with the Pascal VOC 2007+2012 training set. We then retrain Faster R-CNN[2] (Ren et al., 2015) (with Inception-Resnet-V2 as the baseline) to fine-tune the classification network.

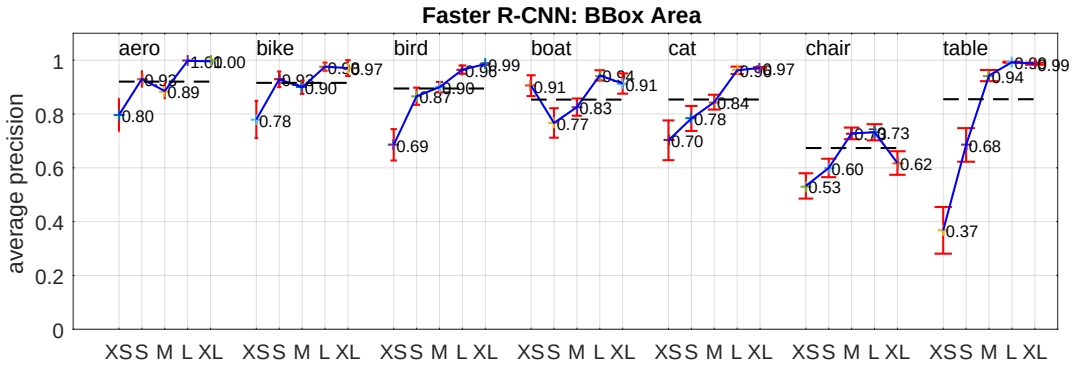
Group	Class	Faster R-CNN[2]	A	M	T	TC
vehicles	aero	87.1	87.5	87.4	87.7	**88.3**
	bike	88.6	88.9	88.9	88.8	**89.1**
	boat	74.4	76.5	76.4	76.8	**77.8**
	bus	87.5	88.5	88.4	88.7	**88.8**
	car	88.7	89.2	89.2	**89.3**	**89.3**
	mbike	86.8	87.4	87.4	87.4	**87.5**
	train	87.3	87.1	87.1	87.2	**87.5**
animals	bird	86.4	86.6	86.7	86.7	**87.2**
	cat	78.2	**79.6**	**79.6**	79.5	79.0
	cow	87.6	88.5	88.5	**88.9**	88.7
	dog	88.0	86.4	86.4	87.0	**88.4**
	horse	89.3	89.6	89.6	**89.9**	89.8
	sheep	78.5	79.6	79.6	**79.9**	79.3
	person	86.6	86.9	86.9	86.9	**87.2**
furniture	bottle	74.7	**77.3**	77.2	77.2	77.2
	chair	65.3	66.9	66.9	**67.1**	**67.1**
	table	**76.1**	75.5	75.6	75.7	**76.1**
	plant	57.9	57.4	57.2	58.1	**60.3**
	sofa	78.6	78.5	78.4	79.0	**80.5**
	tv	78.6	79.6	79.7	79.9	**80.3**
	mAP	81.3	81.9	81.8	82.1	**82.5**

previous experiment, the mean combination and multiplication combination are almost equal with only 0.1% difference in mAP

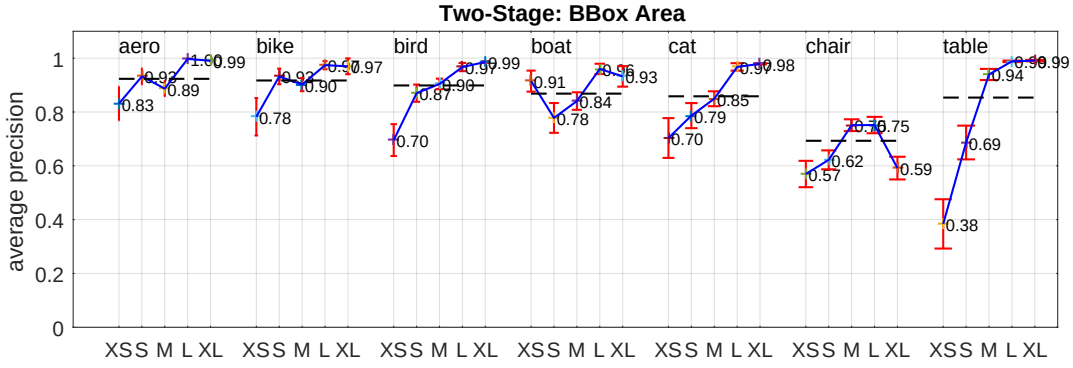
To understand our network performance in detail, we used an error diagnosing tool from the work of Hoiem, Chodpathumwan, and Dai, 2012. In Figure 4.9 and Figure 4.10, we show the sensitivity and impact of different object characteristics on the Pascal VOC 2007 test set. The black dashed lines indicate overall APs. The impact is the difference between best and overall, and the difference between best and worst indicates sensitivity. In general, the two-stage model is more robust than Faster R-CNN to different sizes and aspect ratios.

Error analysis

Figure 4.11 shows the distribution of the top-ranked false positive types. We classify objects into three groups: animals (all animals + person), furniture, and vehicles. Compared with the proposal network results shown in Figure 4.11a, a higher proportion of the error in Figure 4.11b results from poor localization (a duplicate detection or a detection with IoU overlapping the correct



(A) Faster R-CNN results



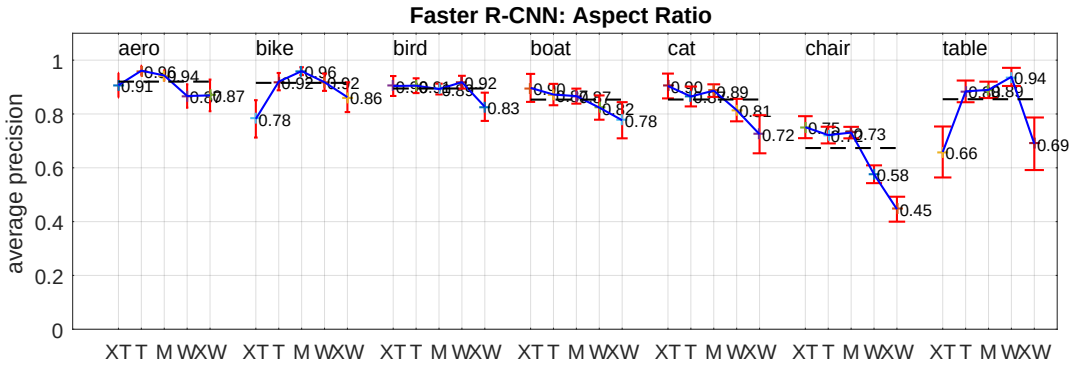
(B) Two-stage model results

FIGURE 4.9: Sensitivity and impact of different object characteristics on the Pascal VOC 2007 test set. The black dashed lines indicate overall APs. Some abbreviations: XS=extra-small; S=small; M=medium; L=large; XL =extra-large.

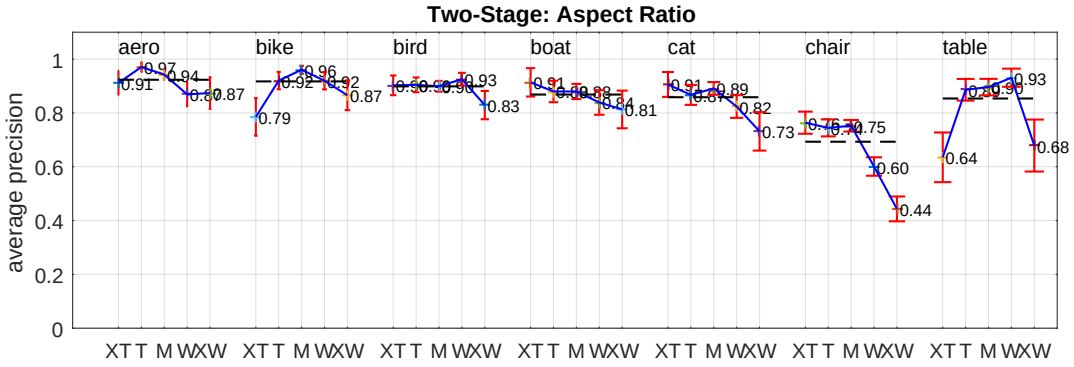
class between 0.1 and 0.5) of the two-stage detector. Since we do not modify the proposal localization, this indicates that the combination results are better than the proposal network results for reducing misclassification (Sim, Oth, and BG). The vehicles group shows significant improvement with the combination method in terms of reducing misclassification errors. On the other hand, the animals group shows only a slight improvement.

Further analysis

As discussed in Section 4.4, we proposed Equation 4.12 to encourage detections that have matched labels between the proposal network and the classification network. In our experiment, we set $c = 1.4$ as a constant boosting weight. The parameter d denotes the high-score threshold. Figure 4.12 shows the effect of choosing the threshold. We try different high-score thresholds d from 0.4 to 0.9 and find that the performance peaks at 0.6, indicating that beyond that value (0.6), the first stage confidence is sufficiently good and the final confidence does not require further boosting. On the other side, if the high-score threshold is below that value, the matching labels between two networks are not reliable enough to correctly predict labels. Note that the performance of two-stage network drops quickly with high score threshold values above that value 0.6. This result also indicates that boosting just



(A) Faster R-CNN results



(B) Two-stage model results

FIGURE 4.10: Sensitivity and impact of different object characteristics on the Pascal VOC 2007 test set. The black dashed lines indicate overall APs. Some abbreviations: XT=extra-tall/narrow; T=tall; M=medium; W=wide; XW =extra-wide.

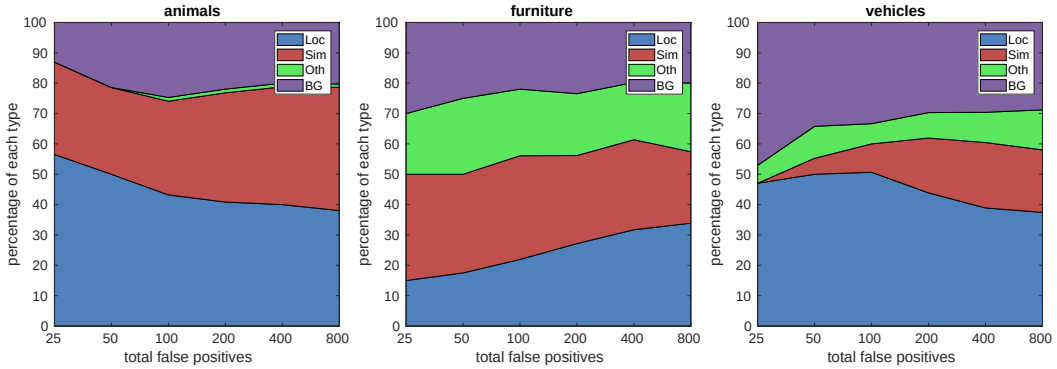
only classification score is harmful to the final combination detection performance.

Detection examples

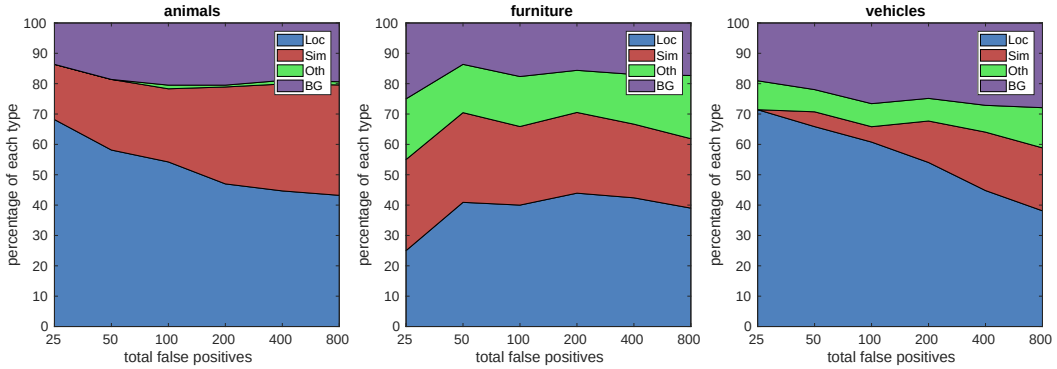
Figure 4.13 shows some detection examples using the Pascal VOC 2007 test set. We randomly selected images and tried to cover all categories in the dataset. We used the trained combination method, which achieved 82.5% mAP, to detect objects.

4.5.3 MS COCO object detection

In this section, we performed experiments on the COCO dataset. COCO is a large-scale object detection dataset with 80 object categories. The training dataset contains 118,287 images, including all training images and a subset of valuation images (coco_2014_train and coco_2014_valminusminival). We use COCO API (Lin and Dollar, 2016) to evaluate our results, which are measured by mAP over IoU in various thresholds. The model results are tested on the mini-valuation test set (coco_2014_minival), which contains 5,000 images. Compared with Pascal VOC, objects in the COCO dataset tend to be smaller, and the number of objects is higher. Thus, we fine-tuned the COCO



(A) Proposal network (Faster R-CNN).



(B) Two-stage detector using trained combination.

FIGURE 4.11: Distribution of top-ranked false positive (FP) types for three groups, i.e., animals, furniture, and vehicles, from the Pascal VOC 2007 test set. The top row shows the distribution of FPs on the proposal network results, and the bottom row shows the distribution of FPs on the trained combination results. The FPs are divided into four error types: poor localization (Loc), confusion with a similar category (Sim), confusion with a dissimilar object category (Oth), and confusion with the background (BG). False positives are sorted in descending order by confidence score.

dataset with more iterations. We trained the proposal network with 1.2M iterations and a base learning rate of 0.0003. We reduced the learning rate by a factor of 10 at 900,000 iterations with a momentum of 0.9. We fine-tuned the classification network for 800,000 iterations with a base learning rate of 0.001.

The results are summarized in Table 4.7. The tested proposal networks were Faster R-CNN with Resnet-101 and Inception-Resnet-V2 as baselines. We set up the training parameters similar to the training with the Pascal VOC 2007 dataset, but with the network initialized by pre-training on ImageNet and the model trained with more iterations. From Table 4.7, the detector (Faster R-CNN and our proposed two-stage model) performance on the COCO dataset was much lower than for the Pascal VOC dataset. Because the COCO dataset contains more object classes than Pascal VOC, the categories are no longer easily separated. For comparison, the average precision of the two-stage network on COCO at IoU of 0.5 was about 60%, whereas the average precision on Pascal VOC was over 80%. We compare our model with

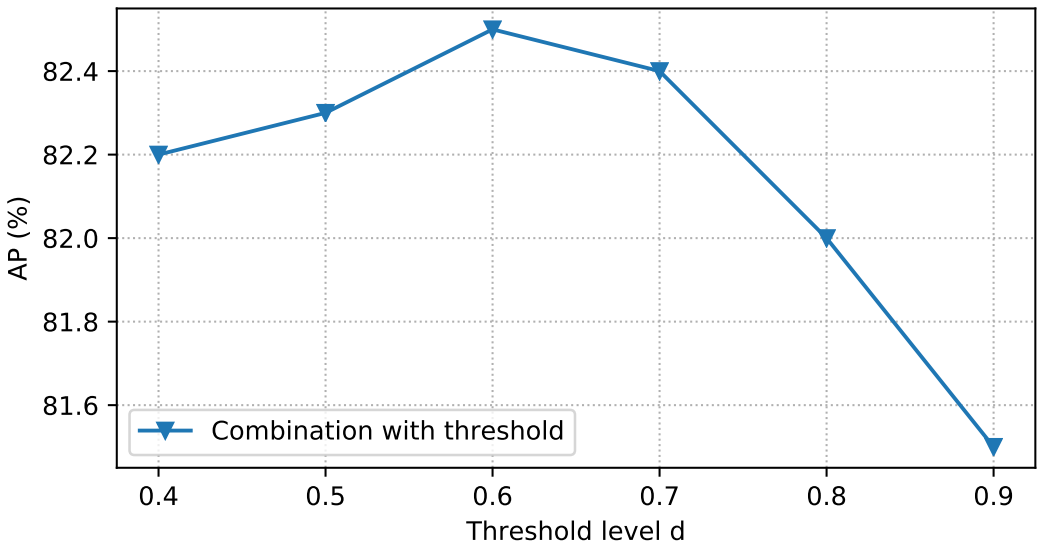


FIGURE 4.12: Experimental results for different high score threshold settings. The results are evaluated on the Pascal VOC 2007 test set.

TABLE 4.7: COCO results on the coco_2014_minival set (bounding box AP). The subscript number indicates the IoU value, and AP without a subscript number indicates the AP (%) at IoU from 50% to 95%. Moreover, S, M, and L indicate small, medium, and large objects, respectively.

	Baseline	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Mask R-CNN	Resnet-101-FPN	37.5	60.6	39.9	17.7	41	55.4
Faster R-CNN	Resnet-101	37.8	55.7	42.5	15.5	43.5	57.6
Faster R-CNN+Inception-V3+A	Resnet-101	37.9	55.8	42.6	15.6	43.6	57.7
Faster R-CNN	Inception-Resnet-V2	43	60.7	48.4	19.1	49.2	64.7
Faster R-CNN+Inception-V3+A	Inception-Resnet-V2	**43.1**	60.8	**48.6**	**19.2**	49.2	64.7
Faster R-CNN+Inception-V3+T	Inception-Resnet-V2	**43.1**	**60.9**	**48.6**	19.1	**49.4**	**64.8**
Faster R-CNN+Inception-V3+TC	Inception-Resnet-V2	43	60.8	**48.6**	**19.2**	49.3	64.7

a state-of-the-art model, namely, Mask R-CNN (He et al., 2017). Note that the Mask R-CNN model in Table 4.7 uses a better feature extractor, Resnet-101-FPN (Lin et al., 2017) (which uses the Feature Pyramid Networks module to detect objects at different scales), than the Resnet-101 feature extractor. Moreover, Mask R-CNN uses the RoIAlign pooling layer, which has better proposal and pooled feature alignment than the RoIPool layer used in the two-stage proposal network. Again, the performance of the model with a high-quality feature extractor (Inception-Resnet-V2) as the baseline performs better than model with a lower quality feature extractor (Resnet-101). Although the performance of the two-stage model depends on which proposal network is used, we observe that the performance is slightly improved in all settings. This indicates that the two-stage model still improves the proposal network performance on the COCO dataset. We also observe that the performance of the trained combination network almost equals the performance of the post-combination network.

Detection examples

Qualitative results are presented in Figure 4.14. Each image was randomly sampled from the COCO dataset (coco_2014_minival). Compared to Pascal VOC, the COCO dataset has more categories of interest and more objects in an image, making it is more difficult to predict. Some images contain both false positive and false negative detections.

4.6 Discussion and Conclusion

4.6.1 How to choose between the trained combination model and the post-combination model

The differences between the models are the method of training the second stage and the combination method. We propose several combination methods and observe that the mean combination (f_1) and the multiplication combination (f_2) are approximately equal. The combination with the threshold function (f_3) is better than f_1 or f_2 alone. However, the f_3 function depends on the high-score threshold and the boosting weight values, and thus is more difficult to optimize. The trained combination is better than f_3 on the Pascal VOC dataset, but is almost equal with f_3 on the COCO dataset because the scores of the first stage affect the classification results of the second stage. The trained combination only works with "high-accuracy" proposals. For instance, at $IoU = 0.5$, the mAP on Pascal VOC, which is approximately 84% is much higher than the mAP on COCO, which is about 60%. (For reasons of comparison, we evaluated the Pascal VOC test set using the COCO evaluation tool.).

In the trained combination model, the second stage uses the output scores of the first stage as additional inputs. We conclude that it needs a reliable proposal (high mAP) network to outperform the post-combination model. The selection between the models depends on the specific dataset. As shown in the experiment section, both proposed models are better than given proposal networks. For a high-mAP proposal model (e.g., $mAP > 75\%$) the trained combination model should be chosen. For an insufficiently high-mAP model (e.g., $mAP < 60\%$), the post-combination model should be chosen. Otherwise, further experiments may help us choose between the models.

4.6.2 How to choose the networks for the first and second stage

Although the first stage has a strong impact on the final performance, choosing the second stage is also important. We analyze the method for choosing networks for the first and second stage of the proposed model.

First, following Huang et al., 2017, the baseline of the first stage is chosen based on its classification performance. However, the SSD meta-architecture appears to be less affected by its baseline's classification accuracy, whereas it

TABLE 4.8: Feature extractor properties used in the second stage. The top-1 and top-5 accuracy (%) are the classification accuracies on ImageNet. The bounding box mAP is evaluated on Pascal VOC 2007 using the same proposal network (Faster R-CNN) and the same combination method (f_3).

Feature extractor	Num. Params.	Top-1 Acc.	Top-5 Acc.	mAP
Resnet-101	43M	76.4	92.9	81.9
Inception-V3	22M	78.0	95.2	82.4
Inception-Resnet-V2	54M	80.4	95.3	82.2

significantly affects the Faster R-CNN meta-architecture performance. Second, because the second stage of the proposed model does not change the proposal location, we should use strong-localization-type detectors, rather than strong-classification-type detectors. For general object detection tasks, Faster R-CNN is a considerably better choice than SSD.

Finally, the second stage is chosen by balancing performance and model complexity. Table 4.8 shows the properties of the feature extractors used in the second stage. We explore the relationship between the performance (mAP) on Pascal VOC 2007 and image classification accuracies on ImageNet, and the number of parameters of the feature extractors used to initialize the second stage. Remarkably, although Inception-Resnet-V2 is the most accurate model on ImageNet, the two-stage model using Inception-Resnet-V2 has a lower performance than the two-stage model using Inception-V3. The Inception-V3 model also has lower complexity than the Inception-Resnet-V2 model. Thus, Inception-V3 appears to be the most appropriate model for use in the second stage.

4.6.3 Effectiveness of the trained combination

Since we analyzed how to use the trained combination above, if the first stage is "good" enough, the trained combination is better than other combination methods. For this reason, we continue to explore the roles of the trained combination module in the second stage as a classifier on the Pascal VOC 2007 dataset. We trained the Inception-V3 and Inception-V3+trained combination module with the same training set and learning rate, and then evaluated both models on the Pascal VOC 2007 test set. In Figure 4.15, we compare the training errors of the two classification models. The error drops more quickly for the Inception-V3+trained combination network, which indicates that the first-stage confidence helps speed up the training classification network process.

4.6.4 Effectiveness for small object detection

Deep learning performance suffers in the case of small object detection. We believe that this is because low-resolution feature maps are used to handle small objects. The proposed model works well for some individual object categories on the small scale.

TABLE 4.9: Miss rate comparison with state-of-the-art pedestrian detection methods on the Caltech test set for the detection of small objects.

Method	Small objects (30-50)	Far scale (20-30)
RPN+BF (Zhang et al., 2016)	79.83	100
CompACT-Deep (Cai, Saberian, and Vasconcelos, 2015)	75.86	100
SA-FastRCNN (Li et al., 2018)	74.49	100
MS-CNN (Cai et al., 2016)	70.34	97.23
F-DNN (Du et al., 2017)	44.86	77.47
F-DNN+SS (Du et al., 2017)	45.14	77.37
SmallDeep (Linh and Arai, 2017)	42.05	77.26
Proposed model	**41.56**	**74.92**

TABLE 4.10: Top improvement performance (AP) for small object detection. The results are evaluated on COCO minival using Faster R-CNN as the proposal network and Inception-V3 as the classification network.

Category	Faster R-CNN (%)	Faster R-CNN+Inception-V3+A (%)	Gap (%)
suitcase	11.5	13.0	1.5
stop sign	23.5	25.2	1.6
carrot	10.3	12.0	1.7
fire hydrant	25.2	27.4	2.2
sheep	21.1	23.5	2.4
toilet	8.6	11.5	2.9

TABLE 4.11: Detection speed (s) and memory usage (GB) of the proposed models. Note that the time and memory usage are comparable within same dataset (same row).

	Proposal network				Classification network			Total	
Dataset	model	baseline	time	mem.	baseline	time	mem.	time	mem.
Caltech ped.	SSD	VGG16	0.087	0.9	Inception-V1	0.014	0.9	0.10	1.8
Pascal VOC	Faster R-CNN	Resnet-101	0.106	2.2	Resnet-101	0.018	4.0	0.12	6.2
Pascal VOC	Faster R-CNN	Inception-Resnet-V2	0.602	10.6	Inception-V3	0.036	3.1	0.64	13.7
COCO	Faster R-CNN	Resnet-101	0.115	2.0	Inception-V3	0.036	3.1	0.15	5.1
COCO	Faster R-CNN	Inception-Resnet-V2	0.602	10.7	Inception-V3	0.036	3.1	0.64	13.8

For the Caltech pedestrian task, we consider the "small objects" setting (or small pedestrian setting), because the miss rate increases drastically in the range of 30 to 50 pixels for most pedestrian detectors (Dollar et al., 2012). The proposed method achieves state-of-the-art performance for small pedestrian detection, as shown in Table 4.9.

Likewise, in the COCO dataset, the largest gaps between the single model detector and the proposed two-stage network are in small object detection. The top six improvement categories for small object evaluation are listed in Table 4.10.

4.6.5 Time and memory analysis

We trained and tested the proposed model on a machine with 32 GB of RAM and a single graphical processing unit (GPU) TITAN X. The models were implemented on a TensorFlow framework (Abadi et al., 2016). Because our proposed method comprises two stages, the running time of the proposed model

is the total running time of the two stages. Table 4.11 summarizes the speed of the proposed model. The running time of the SSD model is faster than that of Faster R-CNN. The running time of the trained combination method is the same as that of the post-combination method with the same baseline. Since the number of proposals is large, we perform non-maximum suppression (NMS) to ensure a maximum of 100 suppression proposals per image. These proposals then are efficiently forwarded to the classification network with batch sizes of 32. The second stage running time is less than 24% of the total running time. Thus, most of the running time is expended on object proposal extraction. Note that the slowest second-stage running time is just 0.036 s per image, but the slowest proposed model (with running time is 0.64s) is still far from real-time (30 frames per second or better).

Since the proposed model requires two networks, the memory usage of the model is the total memory usage of the proposal network and the classification network. Overall, the memory of each stage depends on the size of the feature extractors and whether meta-architectures are used. In Table 4.11, we also report the memory usage for each stage of the proposed model.

4.6.6 Conclusion

This chapter proposed a two-stage deep neural network detector for general objects in images. The detector contains two networks that are respectively used for object proposal extraction and object classification. Each network contributes to the final detection by a combination function. Both the proposal score and classification score are used to predict object appearances. We studied several combination functions that combine two networks after training and/or during second-stage training. The performance varied for different combination functions and combination methods, and the overall performance of the framework depended on the performance of the baseline used. However, we showed that the performance of proposal networks were improved, with some experiments showing state-of-the-art results. As described in the experiment section, the mean combination (A) and the multiplication combination (M) were almost equal. Moreover, we proposed a combination function that boosts proposal detection scores when two predicted classes (proposal class and classification class) are matched. The proposed two-stage network detection proves the connection between the two stage is essential. We also provided a new method (the trained combination method) to aggregate the two networks during the training process by utilizing the first-stage output as additional input for the second stage. Unlike the post-combination method, the trained combination method achieves better performance when combined with a reliable proposal network. In the case of Caltech pedestrian detection, the performance of the trained combination network was worse than the post-combination network. This may be because of low performance of the proposal network. Nevertheless, we introduced a simple, fast and effective two-stage deep network to predict object appearance in images. This chapter confirmed the connection between detection stages is essential for better and high-level inference for object detection.

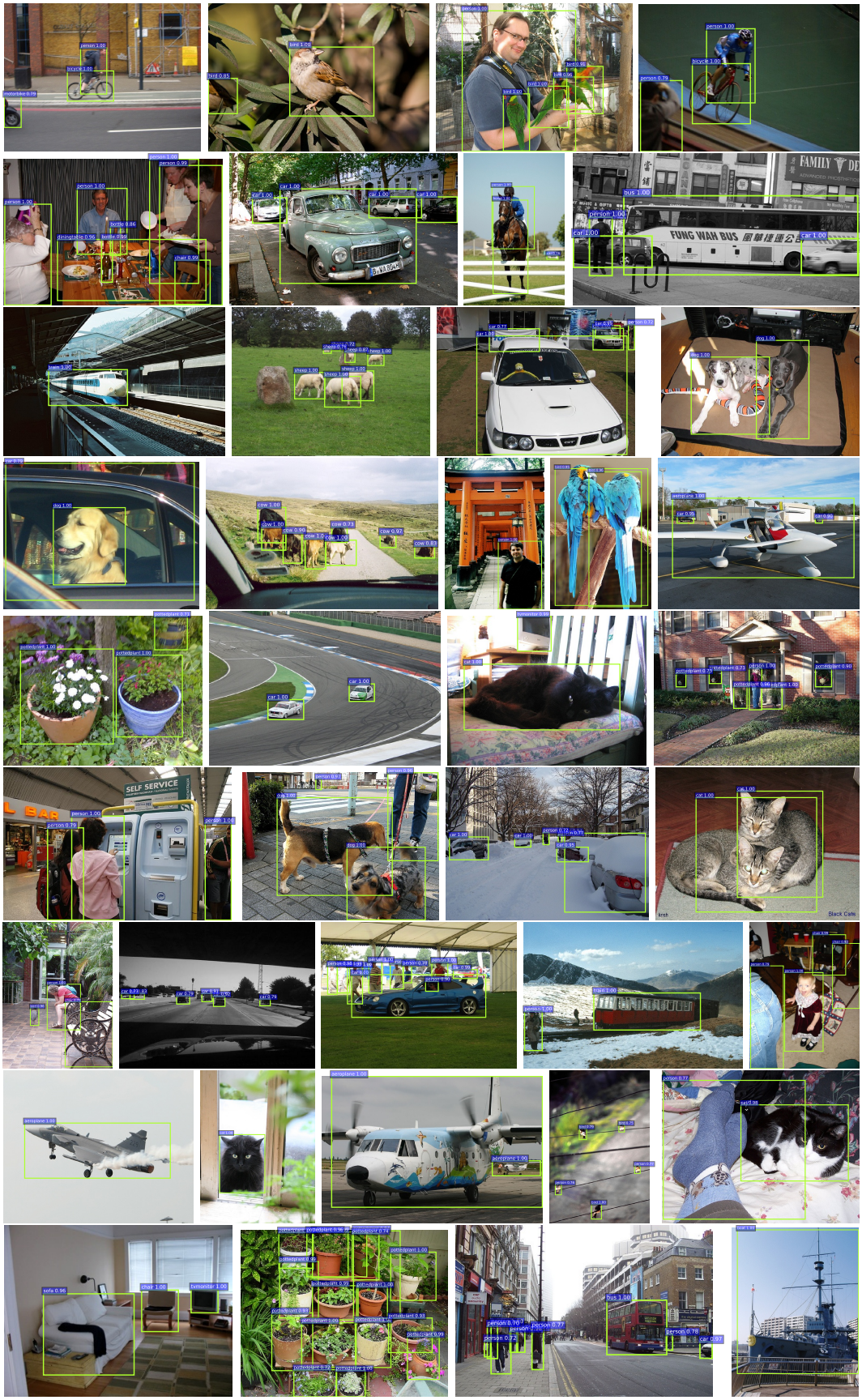


FIGURE 4.13: Detection examples on Pascal VOC 2007 test set. Each image was selected randomly, the detection are selected from the proposed method (trained combination) which achieved 82.5% mAP. All detections are labels with predicted classed, the bounding boxes and detection scores.

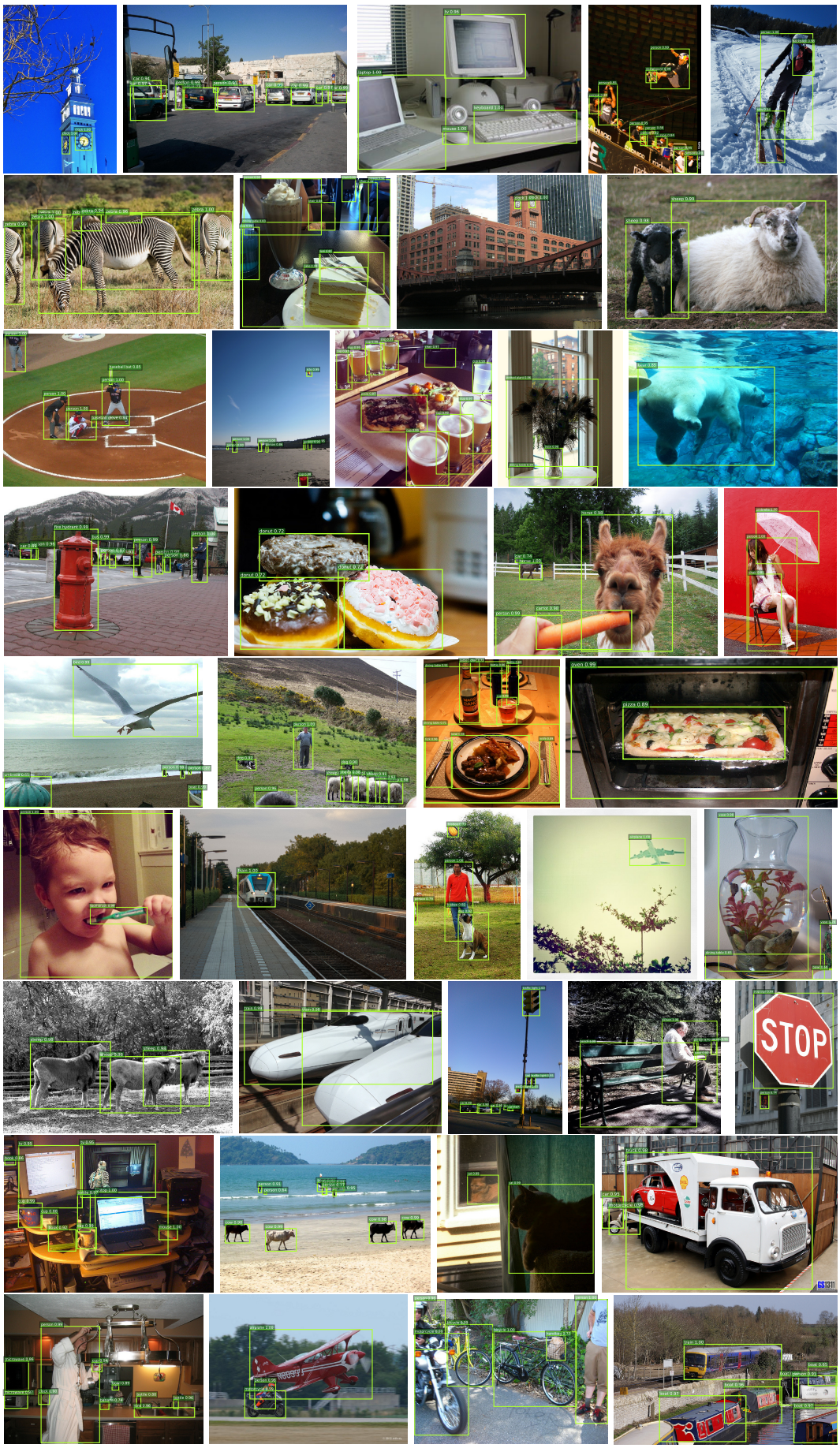


FIGURE 4.14: Detection examples on COCO dataset. Each image was selected randomly.

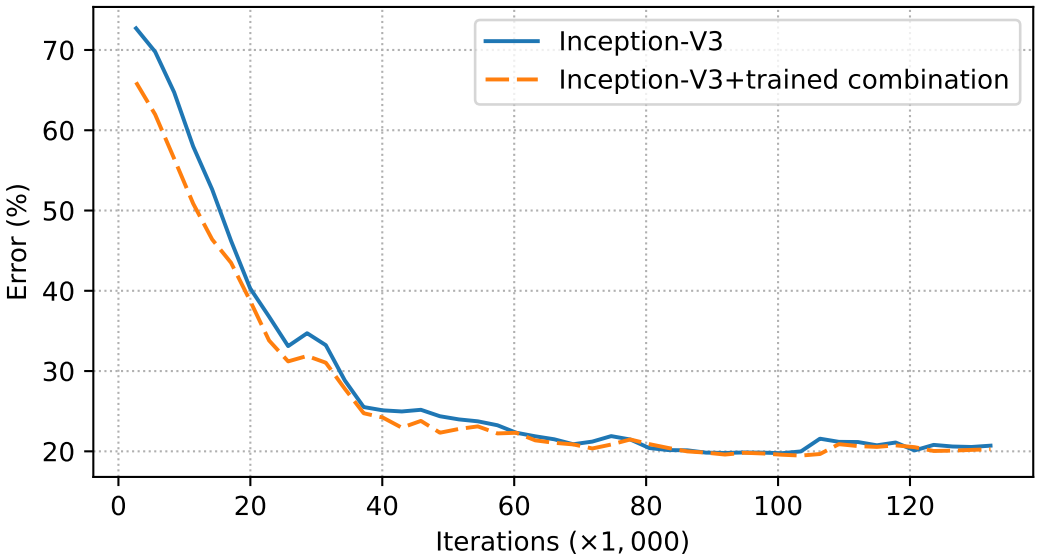


FIGURE 4.15: Top-1 error evolution during training of Inception-V3 vs. an Inception-V3+trained combination module. The computational costs of the two models are approximately equal. The evaluation set is extracted from the Pascal VOC 2007 test set.

Chapter 5

Feature extraction for Instance Segmentation

In this chapter, we present an improvement to the second component in object detection: the feature extractor described in Chapter 3 and Chapter 4. Approaches to improving the performance of object detection include: strong classifier (strong feature extractor), training sample selection, hard negative mining and hard positive generation, multi-scale detection, context modeling, and segmentation. In this chapter, we focus on enriching the feature extractor based on the pre-design baseline.

The goal of this chapter is to exploit the region pooling module to improve the performance of object detection by segmentation (instance segmentation). Detecting the object by segmentation introduces a new problem: segmentation must be performed inside the predicted object locations in an image. This requires a better feature map than object detection by bounding box that guarantees better pixel-to-pixel alignment between input and output. To segment the object inside an image or a bounding box, the network needs to classify the image at pixel level. Thus, the segmentation task can be modeled as predicting a binary mask for each object class in an image. Although the size of objects in an image varies, the detector predicts the mask of an object in a simpler form: a fixed-size mask. The fixed-size mask is then bilinearly resized to the actual object bounding box size. Thus, mask prediction requires a fixed-size pooling module to pool arbitrary regions from feature maps to fixed-size feature maps.

Our inspiration is the RoIAlign pooling layer of Mask R-CNN (He et al., 2017). RoIAlign is a quantization-free layer that preserves the spatial localizations by using a bilinear interpolation (Jaderberg, Simonyan, Zisserman, et al., 2015) to compute the input features at four regular sample locations at each continuous RoI bin, followed by max pooling. In this chapter, we replace the RoIAlign layer with a small network module and ensemble the extracted multi-scale features in a feature map.

Subsection 5.1 discusses multi-scale feature ensembling in deep neural networks. Subsection 5.2 presents the RoI pooling layer, which can typically be divided into two approaches: quantization and non-quantization of the RoI boundaries or bins. This section also describes the effect of RoI pooling layers on the object detection task and instance segmentation task. We present a new subnetwork concept to replace the RoI pooling layer described

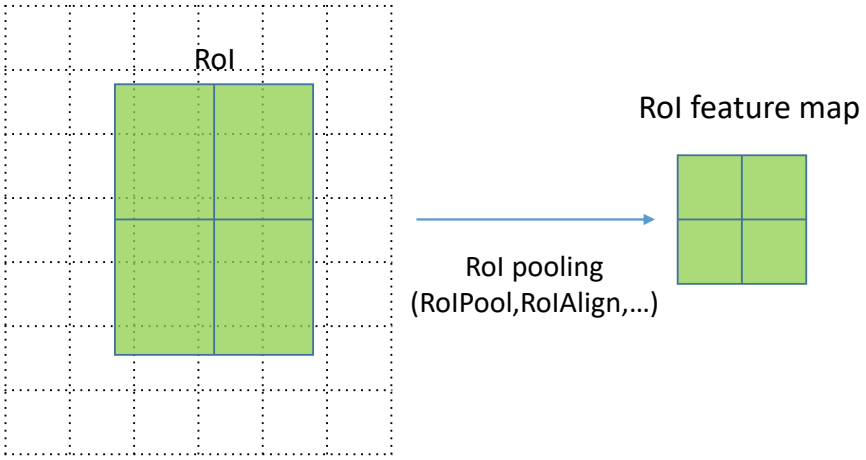


FIGURE 5.1: RoI feature extraction. Fixed-size RoI feature maps are extracted from an RoI of arbitrary size through an RoI pooling layer (e.g., RoIPool, RoIAlign).

in Section 5.3. In this subsection, we present the concrete design and training details of our proposed trainable submodule. Subsection 5.4 present our experimental results on the COCO dataset. Our experiments show that using the subnetwork to extract rich and multi-scale RoI feature maps results in good prediction performance on the object's mask. We also compare the proposed module with other instance segmentation methods and study the effect of the new module on the overall network performance. We present the advantages of our subnetwork, including the training convergence of the detection network. We summarize our conclusions in Subsection 5.5.

5.1 Multi-scale feature ensembling

In a deep network, combining multi-scale features can improve performance. Some methods use segmentation tasks (including semantic segmentation and instance segmentation) by computing the partial scores for each class over multiple scales, such as FCN (Long, Shelhamer, and Darrell, 2015) and Hypercolumns (Hariharan et al., 2015). Numerous methods use a similar strategy for object detection tasks, such as FPN (Lin et al., 2017) and HyperNet (Kong et al., 2016).

Unlike the above methods, the proposed method is based on the concept of incorporating multi-scale features and precisely maintaining the RoI alignment. We replace the RoIAlign layer in the Mask R-CNN method with a subnetwork that ensembles the features within each RoI at multiple scales. This approach enables us to train the RoI pooling module and achieve better RoI feature representation.

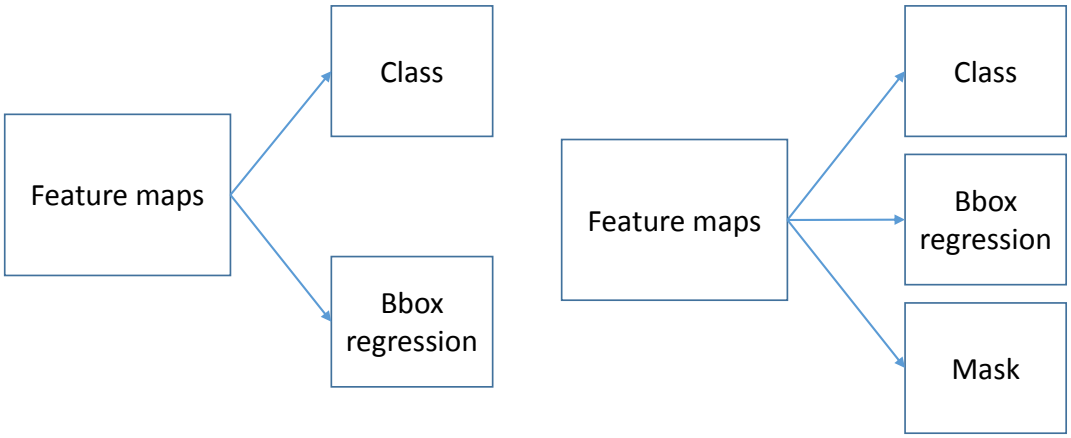


FIGURE 5.2: Faster R-CNN (left) and Mask R-CNN (right). The main difference between these two methods is the number of outputs. Bbox refers to bounding box.

5.2 RoI pooling layer

5.2.1 RoI pooling layer in the detection network

Current state-of-the-art detection networks, such as Mask R-CNN (He et al., 2017), Faster R-CNN (Ren et al., 2015), and Fast R-CNN (Girshick, 2015), use a two-stage design for networks. These methods use a precomputation method (e.g., SelectiveSearch (Uijlings et al., 2013)) or deep networks to extract the set of object region proposals. These proposal boxes are used to extract features within the boxes (called RoI feature maps) by an RoI pooling layer, which are then fed to the next stage (CNN). Figure 5.1 illustrates the concept of using an RoI pooling layer to output fixed-size feature maps. The choice of the RoI pooling layer depends on the detection task. For example, the RoIPool (Girshick, 2015) layer is widely used in object detection (predicting boxes and classes). RoIAlign (He et al., 2017) is an improved version of RoIPool and is used for pixel-prediction tasks. The quality of detection depends on the quality of the pooled RoI feature maps. The RoIAlign layer removes quantizations of RoIPool. For example, it uses $x/16$ instead of $[x/16]$ to perform pooling from coordinate x of RoI to the discrete granularity of the feature map or in other word, a quantization-free layer.

A natural question to be asked is whether a subnetwork module, which replaces the RoI pooling layer, can achieve a similar quality. In the present study, we propose a module that replaces the RoI pooling layer in Figure 5.1 with a subnetwork. We replace the pre-computed RoI with a deep network called the region proposal network (RPN), which is based on the Faster R-CNN method. However, in the instance segmentation task, the subnetwork must ensure pixel-to-pixel alignment between the RoI and the extracted features. The proposed subnetwork can be considered a trainable version of the RoI pooling layer.

5.2.2 From object detection to instance segmentation

We consider the instance segmentation as an extension of object detection by changing the number of outputs of the network. We follow the concept of the Mask R-CNN method, which extends the Faster R-CNN method. Although the Mask R-CNN method is used for a different task (instance segmentation), this method has a strong connection with Faster R-CNN. Figure 5.2 shows the differences between these two methods. The outputs of each model used to define the task of interest are referred to as network heads. Mask R-CNN extends the Faster R-CNN by adding a mask branch to predict an object mask in parallel with the existing bounding box prediction branch and the class prediction branch. The mask branch predicts a fixed size mask ($m \times m$) for every class, resulting in N binary masks, where N is the number of classes. Mask R-CNN is an effective framework for instance segmentation. The proposed method is based on the Mask R-CNN method, but the quality of segmentation is improved.

5.2.3 RoI feature extraction

Proposal-based methods such as Faster R-CNN and Mask R-CNN are incorporated into a pooling method in order to pool the features within each RoI into fixed-size features. The first method is RoIPool (Girshick, 2015), which converts a feature with size $h \times w$ into a small feature map with size $H \times W$ (e.g., 7×7). RoIPool operates by max-pooling an $H \times W$ grid of sub-windows of approximate size $h/H \times w/W$. RoIPool is a special case of spatial pyramid pooling in SPPNet (He et al., 2014). Although RoIPool works well for the object detection task, it appears to decrease the pixel-level prediction performance because the RoI and the extracted features are misaligned. To address this problem, Mask R-CNN introduces the RoIAlign pooling layer to replace the RoIPool layer. RoIAlign is a "quantization-free" layer that preserves the spatial localizations by using a bilinear interpolation to compute the input features at four regular sample locations at each continuous RoI bin followed by max pooling. The extracted features have better-preserved spatial correspondence than RoIPool.

5.3 Multi-scale subnetwork for RoI pooling

As described in the Subsection 5.2.2, our overall network contains two components. The first component (RPN) extracts the set of network proposals, and the second stage uses Fast R-CNN to perform object classification, bounding box regression, and mask prediction from features that are extracted from each candidate box through a subnetwork.

5.3.1 Subnetwork design

Figure 5.3 shows the proposed model for the mask prediction branch. The subnetwork contains three branches, and at each branch, the features in the

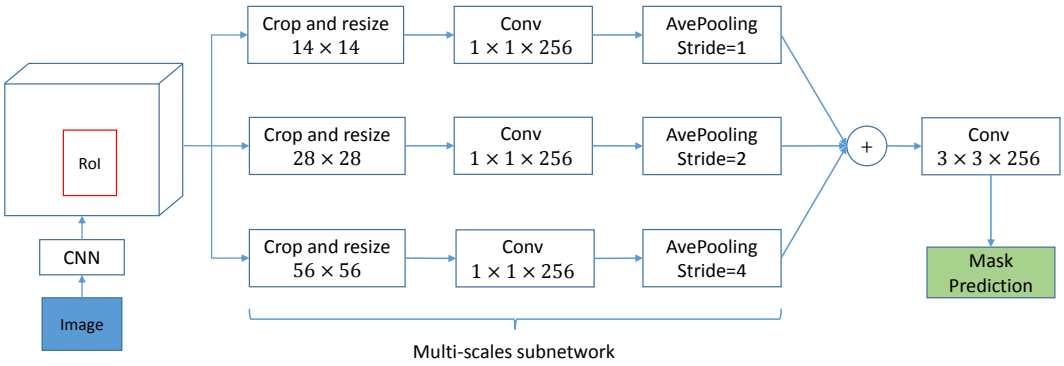


FIGURE 5.3: Subnetwork design for mask prediction branch. The subnetwork consists of three branches that pool the feature map at different scales. We then down-sample and concatenate the three branches using a convolutional layer for mask prediction.

proposed RoI are cropped into different scales. To preserve the alignment between the RoI and the extracted features in the corresponding RoI, we use the "crop_and_resize" function in TensorFlow (Abadi et al., 2016) to crop and bilinearly resize the input images to a fixed size. We use a 1×1 convolution layer after each "crop_and_resize" operation to maintain the number of outputs as 256. Feature maps are then down-sampled to the smallest fixed-size output (e.g., 14×14) by the average-pooling layer, and concatenated. Finally, we use a convolution layer to reduce the number of outputs to 256, and we use the convolution layer as a prediction mask, as in Mask R-CNN. We analyze two properties of the proposed subnetwork:

- Precise RoI and pooled RoI features alignment, and
- Multi-scale feature representation

First, the goal of the subnetwork is to extract small feature maps from each RoI to fixed-size features in the same manner as RoIPool and RoIAlign. Moreover, the "crop_and_resize" function and the pooling layer maintain the alignment between the RoI and the output features of each subnetwork branch. According to He et al., 2017, this property is crucial for the instance segmentation task. Second, the proposed three-level sub-branch covers multi-scale RoI features. Ensembling features across multiple scales has proven beneficial in computer vision tasks.

5.3.2 End-to-end training

We adopt end-to-end joint training of the RPN and network heads. For training, the multi-task loss function is a combination of three losses: the cross-entropy classification loss (L_{cls}), the box regression loss (L_{box}), and the mask loss (L_{mask}). To predict the class and box regression, the class loss $L_{cls}(p, u) = -log(p_u)$ is the cross-entropy loss for the true class u. The second loss $L_{box} = L_{box}(t, t^*)$ is defined over a tuple of true bounding-box regression targets t and a predicted tuple t^*, as in Fast R-CNN. The final loss L_{mask} is

defined over an $N \times m \times m$-dimensional output for each RoI, where N is the number of classes. Here, the k-th L_{mask} is the average binary cross-entropy loss and is defined as

$$L_{mask} = -\frac{1}{m^2} \sum_{1 \leq i,j \leq m} (y_{i,j} log \hat{y}^k_{i,j} + (1 - y_{i,j}) log(1 - \hat{y}^k_{i,j})) \tag{5.1}$$

where $y_{i,j}$ is the label of cell (i, j) and $\hat{y}^k_{i,j}$ is the predicted value of cell (i, j). The mask branch output is a pixel-wise binary classifier, because it outputs one mask for each class, and there is no competition between classes. The overall training loss is defined as

$$L = L_{cls} + \lambda[u \geq 1]L_{box} + L_{mask} \tag{5.2}$$

where $[u \geq 1]$ is equal to 1 when $u \geq 1$, and 0 otherwise. In the experiments, we set the balance loss weight λ to 1.

5.3.3 Training details

We initialize the subnetwork convolution layers with MSRA (He et al., 2015) and constant initialization for biases. Through experiments, we found that normalizing the output of the convolutional layers by using a batch normalization (Ioffe and Szegedy, 2015) layer after convolution boosts the performance. The last convolutional layer after concatenation does not need a batch normalization layer. We adjust the hyper-parameters from Mask R-CNN during training. The image is resized to a shorter edge of 800 pixels, and each GPU has a batch size of one image. For data augmentation and to prevent over-fitting, we apply random flipping to the training dataset. We train the dataset in a 2-GPU system with a total of 720k iterations. The base learning rate is 0.0025 (the learning rate is reduced by a factor of 10 after 480k iterations and 640k iterations). The weight decay is 10^{-4} and the momentum is 0.9. An RoI is considered to be positive if it has an IoU with a ground-truth box of at least 0.5. We maintain the ratio of positive to negative proposal RoI as $1 : 3$. The resolutions of RoI features for Resnet-50 and Resnet-101 (He et al., 2016) are 14×14 and 28×28, respectively.

5.4 Experimental results

5.4.1 Dataset setup

In this section, we present the experimental results on the COCO dataset as described in Subsection 4.5.3. Here, we evaluate the instance segmentation task. Each object in the image is annotated with a bounding box and a binary mask inside the bounding box. We used the COCO API (Lin and Dollar, 2016) to evaluate our results, which are measured by average precision (AP or mask AP) over IoU in various thresholds and object scales. We report the

AP in the 5,000-image valuation set (the minival set). The IoU threshold and the object scales are shown in Table 5.1

TABLE 5.1: COCO dataset instance segmentation settings.

Evaluation terms	IoU (%)	Object size: A
AP	50-95	All
AP_{50}	50	All
AP_{75}	75	All
AP_S	50-95	$A < 32 \times 32$
AP_M	50-95	$32 \times 32 < A < 96 \times 96$
AP_L	50-95	$96 \times 96 < A$

5.4.2 Instance segmentation results

We tested our proposed model with different network settings. In order to demonstrate the effect of the proposed subnetwork, we used different baselines for feature extraction over an input image. We compare the results of the Mask R-CNN method, which uses the RoIAlign pooling layer, and the proposed method, which uses the subnetwork for RoI feature extraction. To ensure a faithful comparison, we attempt to train the methods under the same conditions, including the baseline, weight initialization, number of iterations, and system configuration. We also explored the effect of the proposed multi-scale feature pooling under the feature pyramids baseline (e.g., the feature pyramid network (FPN) Lin et al., 2017). In our experiments, we used a computer with Intel Core i7-6800K CPU (3.40GHz), 64GB of RAM and a NVIDIA TITAN X GPU.

TABLE 5.2: Instance segmentation results (%) for the Resnet-50 baseline.

Model	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Mask R-CNN	31.4	52.8	33	12.1	34.5	49.6
Proposed model	**31.8**	**53.5**	**34.1**	**12.7**	**34.8**	**50.8**

The first experiment is performed with Resnet-50, which has a depth of 50 layers. The features are extracted from the final convolutional layer of the fourth stage. In this experimental setting, we do not use the feature pyramid extraction for baseline features. The results are shown in Table 5.2. Overall, the proposed model APs are improved for all IoU threshold and object size settings. At a high value of IoU (75%), a gap in AP of 1.1% indicates that the proposed model is beneficial under good conditions.

TABLE 5.3: Instance segmentation results (%) for the Resnet-101 baseline.

Model	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Mask R-CNN	37.5	60.6	39.9	17.7	**41.0**	55.4
Proposed model	**37.7**	**60.7**	**40.2**	**17.8**	40.9	**55.6**

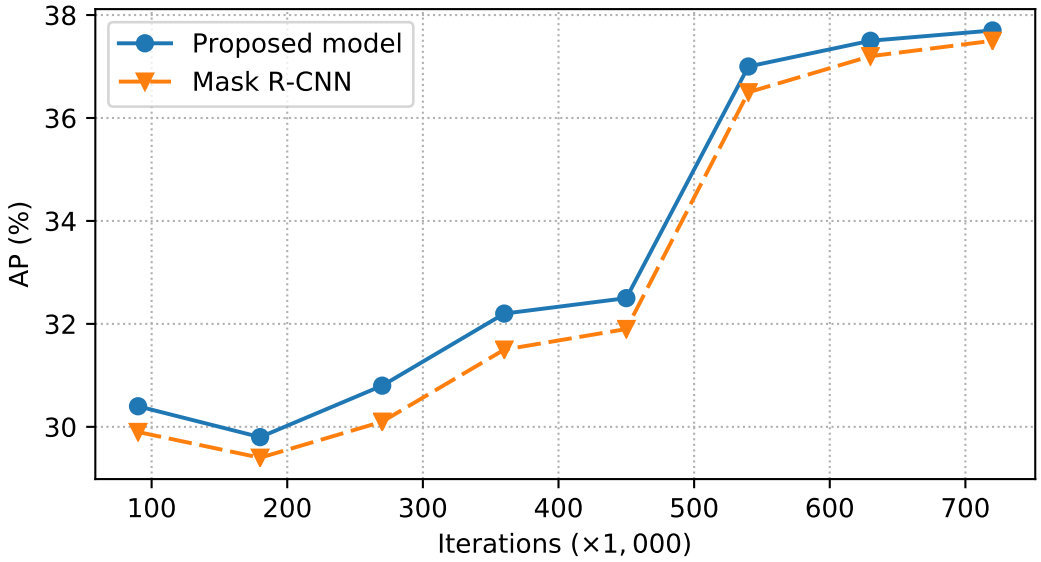


FIGURE 5.4: Average precision (AP) evolution during training of Mask R-CNN vs. the proposed model using the ResNet-101-FPN baseline. The results are evaluated in the COCO minival set.

The second experiment is performed with a higher quality feature extractor, i.e., Resnet-101 with FPN design. The FPN is used to augment a feedforward network (e.g., Resnet-101) with a top-down pathway and lateral connections. The FPN extracts the feature pyramid from a single-resolution input image. The extracted RoI features from different pyramid levels can be used for box detection and mask prediction. The overall AP is better than that of the model using the Resnet-50 baseline, which shows the relationship between the instance segmentation performance and the classification performance of the baseline on ImageNet (Russakovsky et al., 2015). Table 5.3 details the experimental results. The obtained results are better than those of Mask R-CNN at various settings. For IoUs ranging from 0.5 to 0.95, we obtained a mean AP of 37.7%, which is 0.2% higher than that of the Mask R-CNN model. However, the gaps are smaller than those in the first experiment. We believe that since the feature extractor (Resnet-101-FPN) outputs pyramid features, the effect of the proposed multi-scale subnetwork is not strong, as in the first experiment, which used a non-pyramid feature extractor.

Figure 5.4 shows the AP during training of Mask R-CNN and the proposed model. We trained both models under the same configuration and learning rate. The curve indicates the benefit of using the proposed subnetwork to replace the RoIAlign layer of the Mask R-CNN model. The proposed model increases the AP quicker than Mask R-CNN, especially at early iterations, because the proposed model embeds the RoI features at multi-scale, and thus outputs more robust features for mask prediction.

TABLE 5.4: Running time (s) and memory usage (GB) of the proposed model compared with Mask R-CNN.

Model	Baseline	Test time	Memory
Mask R-CNN	ResNet-50	0.163	8.5
Proposed model	ResNet-50	0.195	9.3
Mask R-CNN	ResNet-101-FPN	0.251	6.6
Proposed Model	ResNet-101-FPN	0.260	7.0

5.4.3 Complexity

We report the model running time of the proposed method in Table 5.4. Since the proposed subnetwork uses more parameters than the RoIAlign layer, the model running time is slower and uses more memory than Mask R-CNN. However, an increased time of less than approximately 20% and an increased memory usage of approximately 6% are reasonable.

5.4.4 Instance segmentation examples

We show some results in Figure 5.5. We used Resnet-101-FPN as the baseline for feature extraction. The proposed method achieves good results under difficult conditions.

5.5 Discussions and Conclusions

In experiments, we applied the proposed network for two baselines (with and without pyramid feature extraction). Although the proposed model outperforms Mask R-CNN using the same baseline, the gaps in the first model (using ResNet-50 as a baseline) are larger than those in the second model (using ResNet-101-FPN as a baseline). This is because ResNet-101-FPN already performs mask prediction in multi-level layers. The proposed multi-scale RoI pooling subnetwork effectively augments pooled feature maps at multiple feature resolutions. Thus, the proposed model greatly improves when using a non-pyramid feature extractor as a baseline.

Another problem with the proposed subnetwork is the choice of hyper-parameters, such as the number of sub-branches, the output size of cropped RoI features, and the number of outputs for each convolutional layer. Due to the limited GPU memory, we only use three sub-branches at three sizes ($14 \times 14, 28 \times 28$ and 56×56).

In this section, we present the potential for applying the proposed module for the two-stage object detection network described in Chapter 4.

5.5.1 Feature extractor for bounding box object detection

Here, we present the RoI feature extraction module that is applied for the instance segmentation task. In general, the RoIPool layer, the RoIAlign layer, and our proposed subnetwork extract the same dimensional output. These

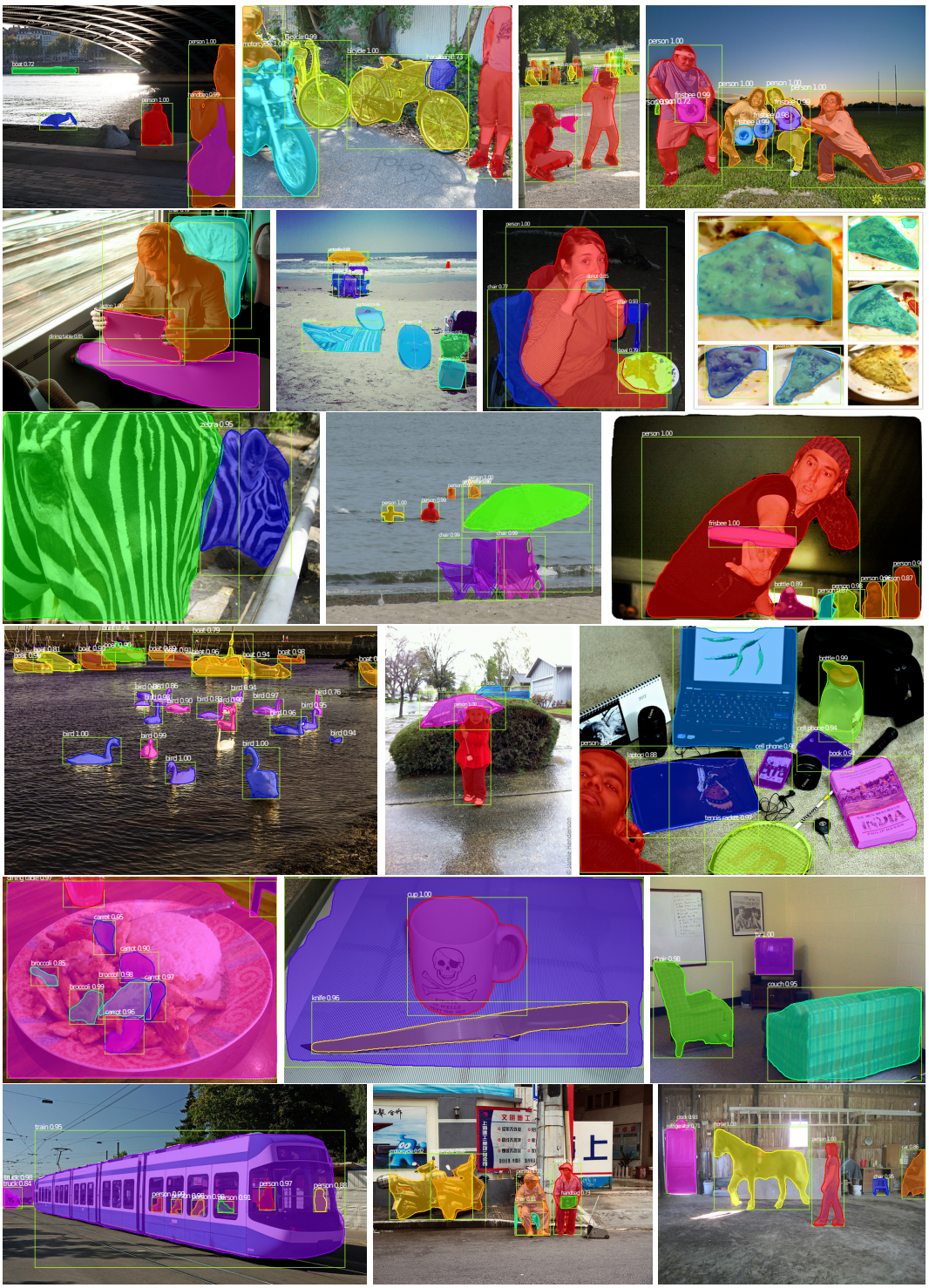


FIGURE 5.5: Detection examples on the COCO dataset based on Resnet-101-FPN. Each image was randomly selected. The method achieves mask AP of 37.7%. The masks are presented in color, bounding box, class and confidence score.

layers/subnetwork can be replaced with each other. The proposed module can be used for object detection by the bounding box.

TABLE 5.5: Object detection by bounding box on the COCO minival set. All models use same baseline ResNeXt-101-FPN.

Model	Baseline	AP	AP_50	AP_75	AP_S	AP_M	AP_L
Faster R-CNN	ResNeXt-101-FPN	41.5	63.8	44.9	24.8	45.4	53.5
Proposed model	ResNeXt-101-FPN	**42.3**	**64.0**	**46.4**	**25.5**	**46.4**	**54.4**

Table 5.5 shows the performance of our proposed model using the bounding box only on the COCO minival set. We used the ResNeXt-101-FPN (Xie et al., 2017) feature extractor for Faster R-CNN and the proposed model, which used the RoIAlign pooling layer. To obtain this result, we trained our proposed model with the mask branch and ignored the mask prediction at the inference time. The performance (bounding box AP) increased by 0.8%, indicating the efficiency of using the subnetwork module. We also observe the performance improvement for all settings.

5.5.2 Conclusions

We proposed a multi-scale, subnetwork RoI pooling module for the instance segmentation task. Although the rest of the detection network follows the design of Mask R-CNN (He et al., 2017), we presented an alternative design for better RoI feature pooling. The subnetwork module not only maintains the pixel-to-pixel alignment between input proposal regions, which is critical for segmentation, but also encodes multi-scale features for better mask prediction. The subnetwork allows end-to-end training with the whole network.

We studied the effectiveness of the new module with different baseline networks. We conclude that multi-scale feature embedding improves the performance of instance segmentation. The proposed model improves the performance of a non-pyramid feature extractor as a baseline. However, in our design, it is unclear how to specify the optimal number of multi-scale pooling branches or the size of the pooled features.

We also analyzed the effect of the proposed model on the speed of the training process. In our experiment, with the same baseline settings, the new model improved the AP faster than Mask R-CNN, i.e., the model required less time to converge.

Further, for bounding box object detection, the proposed model with subnetwork outperformed Faster R-CNN using the same baseline and RoIAlign for RoI feature extraction. We think one reason is the benefit of multi-task learning.

Chapter 6

Conclusions

The present thesis described deep learning frameworks for the object detection problem in images. Our proposed approach builds in concepts of image recognition and feature extraction based on deep learning to provide an efficient method for object detection. Our main contribution was to propose a robust framework for object detection tasks by reducing misclassification. We introduced several combination methods and proved the advantages of a two-stage model for object detection. The trained combination method improved the classification performance and overall detection performance of the model. We also discussed how to choose an appropriate first stage, second stage, and combination method in order to construct a strong two-stage detector for a specific application. We showed that the use of two networks gives state-of-the-art performance for object detection.

6.1 Key contributions

The thesis has several contributions as follows:

- **Two-stage network detector**. Our first contribution is the proposed two-stage network based on deep learning to improve object detection performance. The hierarchical network aims to reduce misclassification by processing the image through a proposal network and a classification network. The testing is simple because we do not use enhanced methods such as multi-crop, horizontal flipping, or multi-scale during testing. The overall conclusion is that the intensive connection and high-level inference detection improve detection quality by eliminating the detection errors of the given object detector.

- **Combination functions between two networks**. We combined two network detection results by providing connection functions between two networks. In this thesis, we discussed how to combine two networks by two types of combinations: post-combination and trained combination. The detection performance of a given deep-learning-based detector (the proposal network) can be significantly improved if the outputs of that network are re-classified. We contributed simple post-combination functions to combine two networks and re-score each detection by a secondary classifier. Moreover, the trained combination that utilizes the output detection scores as additional inputs for the

second-stage classifier improves the overall performance of the detection network. The processing time of the proposed framework is reasonable, and our experiments show that the framework achieves good performance for object detection under difficult conditions, such as small-scale objects.

- **New feature extraction for instance segmentation**. To improve network performance, we introduced a new feature extractor design. The new network aims to improve the performance of object detection by segmentation. We built a rich and multi-scale feature extractor from the proposed Region of Interest to a fixed-size feature map. The new design allows multi-scale feature to be encoded based on a deep learning subnetwork. Our design can be trained while simultaneously maintaining the pixel-to-pixel alignment for object location and mask prediction. Experimental results show the potential of applying the subnetwork for proposal-based detection networks.

6.2 Limitations of the method

Our proposed framework has a number of limitations, both intrinsic and other issues, which we plan to investigate in the near future. In this section, we describe the intrinsic limitations, and in the next section, we discuss future work.

- **Time and space complexity.** Although the proposed models outperformed conventional proposal detection networks and performed well in detecting small objects, one limitation of the proposed method is time and space complexity. As per Subsection 4.6.5, the proposed model consists of two deep learning networks, so the running time of the overall detector is slower than single detectors. The two-stage network requires training the first stage to extract the set of proposal detections. The second stage uses the proposals for training classifiers. Thus, the two-stage needs more time for training and testing. Moreover, the memory usage of the proposed framework is higher than that for single-stage networks.

- **Model complexity.** Increasing the model capacity affects not only the model running time and memory usage, but also the training process. For instance, increasing the model complexity raises the probability of over-fitting and unstable training. In other words, it is more difficult to optimize two stages than a single stage . The classification network has a relatively large number of parameters to be optimized. The other problem is that the number of training samples for the classification network is relatively small because the training samples are focused on the object only. This drawback leads our proposed model to not be suitable for small-size datasets.

6.3 Future work

Joint training two network stages

The proposed two-stage network can be considered as two learning tasks: object detection and object classification. For the training process, we trained two networks separately and combined the two stages at the end of the training. There are some benefits to training two stages at once (joint training) or end-to-end training (Ruder, 2017). Joint training aims to improve detection and classification simultaneously by combining the common knowledge from both tasks. Moreover, with joint training, we can reduce the complexity of the training pipeline, making the detector easier to test and deploy.

Extend the trained combination

In the trained combination model in Subsection 4.4.2, we used the scores of the proposal network as additional input for the classification network, However, there are additional first-stage outputs that could be used in addition to the confidence scores. For example, the box sizes can be used as input for the second stage. The box size can be used to predict which proposal model will perform detection better, improving the second-stage classification performance.

Model running time and memory reduction

Generally, the more accurate the model is, the more time and space complexities are required. Huang et al. extensively surveyed the trade-off between accuracy and speed for modern convolutional object detectors (Huang et al., 2017). However, we think the requirement for model deployment in devices with low memory resources is critical. The constraints of hardware in various small platforms (e.g., mobile, robotic) are also a problem for deep learning applications. With regard to reducing model running time and memory usage, there are several interesting approaches for model compression and acceleration for deep neural networks with minimal loss of accuracy, such as quantization and binarization (Gong et al., 2014), and pruning and sharing (Srinivas and Babu, 2015; Han et al., 2015; He, Zhang, and Sun, 2017). It is worth investigating the effect of model compression on our detection model.

Appendix A

Image object detection and evaluation

We review some common metrics for the object detection problem that are used in this thesis. The evaluation of object detection, which requires the evaluation of object class and location prediction, can actually be reduced to the evaluation of the classification problem. A predicted bounding box is matched with the ground-truth to form a true positive or false positive detection. Most object detection evaluations define a fixed threshold of IoU (e.g., threshold = 0.5) to match the predicted box. Likewise, we use the definition of the Pascal object detection challenge in Everingham et al., 2010. A detection is represented by the box coordinates $B_{dt} = \{x_1, y_1, x_2, y_2\}$ and the detection score. The detection is then matched with the ground-truth box $B_{gt} = \{x_1^*, y_1^*, x_2^*, y_2^*\}$. A detection is considered true if the overlap between the predicted region and ground-truth region

$$IoU = \frac{area(B_{dt} \cap B_{gt})}{area(B_{dt} \cup B_{gt})} \tag{A.1}$$

exceed 0.5.

There is the possibility that one ground-truth matches with multiple true detections. In this case, only the detection with the highest IoU value is considered a true positive detection; the remaining detections are considered false positive detections. Table A.1 summarizes the error types.

TABLE A.1: Table of error types.

		Detector	
		Object	Other
Ground-truth	Object	True Positive TP	False Negative FN
	Other	False Positive FP	True Negative TN

Non-maximum suppression is a method used to reduce number of multiple detections as depicted in Figure A.1. Note that the evaluation threshold is not necessarily equal to the threshold of non-maximum suppression post-processing.

For multiple-class evaluation, the evaluator processes each class separately to calculate the metric for each class. The final performance of the detector is the average value of the performance for each class.

In this appendix, we review metrics commonly used in object detection: the average precision and the miss rate. These metrics are used to evaluate the proposed model in this thesis and to compare the proposed model with other methods.

A.1 Object detection

A.1.1 Ground-truth

In object detection, ground-truth represents the desired output of an detector on an input image. It is represented by the box coordinates $B_{gt} = \{x_1, y_1, x_2, y_2\}$ and the object class.

A.1.2 Confident level and score

The confident level (confident, confident score) and score are used interchangeably in this thesis and they are same calculation. However, they are different in the context of object detection. Confident level is the probability of predicted object (e.g., in the Equation B.2), confident level is a number between 0 and 1. Score is the output of detector that is used to rank the detections. Score can be any value.

A.1.3 Non-maximum suppression (NMS) algorithm

In object detection, the detector typically produces multiple detections with high scores close to the ground-truth location. NMS is used to make sure that a particular predicted object is identified only once by eliminating multi-detection. Figure A.1 shows the pseudo code for NMS algorithm.

A.2 Precision/Recall and ROC

Precision indicates how often the prediction is correct. Recall indicates the percent of correctly detected positive instances over the total amount of relevant instances. These are respectively defined as

$$precision = \frac{TP}{TP + FP} \tag{A.2}$$

and

$$recall = \frac{TP}{TP + FN} \tag{A.3}$$

A perfect precision score of 1.0 means that every sample is a true positive, but does not indicate whether all positive samples were detected. A perfect

Input: $\mathcal{B} = \{b_1, ..., b_N\}, \mathcal{S} = \{s_1, ..., s_N\}, t$
$\mathcal{B}$ is the list of initial detection boxes
$\mathcal{S}$ contains corresponding detection scores
t is the NMS threshold
begin
  $\mathcal{D} \leftarrow \{\}$
  while $\mathcal{B} \neq empty$ **do**
    $m \leftarrow argmax(\mathcal{S})$
    $\mathcal{M} \leftarrow b_m$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{M}$
    $\mathcal{B} \leftarrow \mathcal{B} - \mathcal{M}$
    for b_i *in* $\mathcal{B}$ **do**
      if $IoU(\mathcal{M}, b_i) \geq t$ **then**
        $\mathcal{B} \leftarrow \mathcal{B} - b_i$
        $\mathcal{S} \leftarrow \mathcal{S} - s_i$
      end
    end
  end
  return $\mathcal{D}, \mathcal{S}$
end

FIGURE A.1: NMS algorithms.

recall score of 1.0 means that all positive samples were detected, but does not tell us how many false positive samples were also detected. There is an inverse relationship between precision and recall. Usually, precision and recall are not discussed in isolation. Rather, they are viewed together as a "precision-recall tradeoff". Depending on the requirements of the specific task, the application may require a high-precision or high-recall detector. The tradeoff between precision and recall can be observed using the precision-recall curve, which plot precision versus recall.

On the other hand, the Receiver Operating Characteristic (ROC) curve represents the relationship between sensitivity (recall or True Positive Rate (TPR)) and False Positive Rate (FPR).

$$FPR = \frac{FP}{FP + TN} \tag{A.4}$$

and

$$TPR = \frac{TP}{TP + FN} \tag{A.5}$$

Thus, we have two curves to estimate the performance of a detector. However, the two curves have different uses. In particular, if the positive samples are very rare compared to the negative samples or if true negative samples are not very valuable to the problem, the Precision/Recall curve is more appropriate. Otherwise, the ROC curve is preferred.

A.3 Area Under the Curve (AUC)

When summarizing the model performance, the Area Under the Curve (AUC) is used to represent the tradeoff between two values. That is, the AUC represents the entire curve by a single reference value. For example, the high AUC of the Precision/Recall curve represents both high precision and high recall. Figure A.2 shows an example of AUC used to summarize the Precision/Recall tradeoff[1].

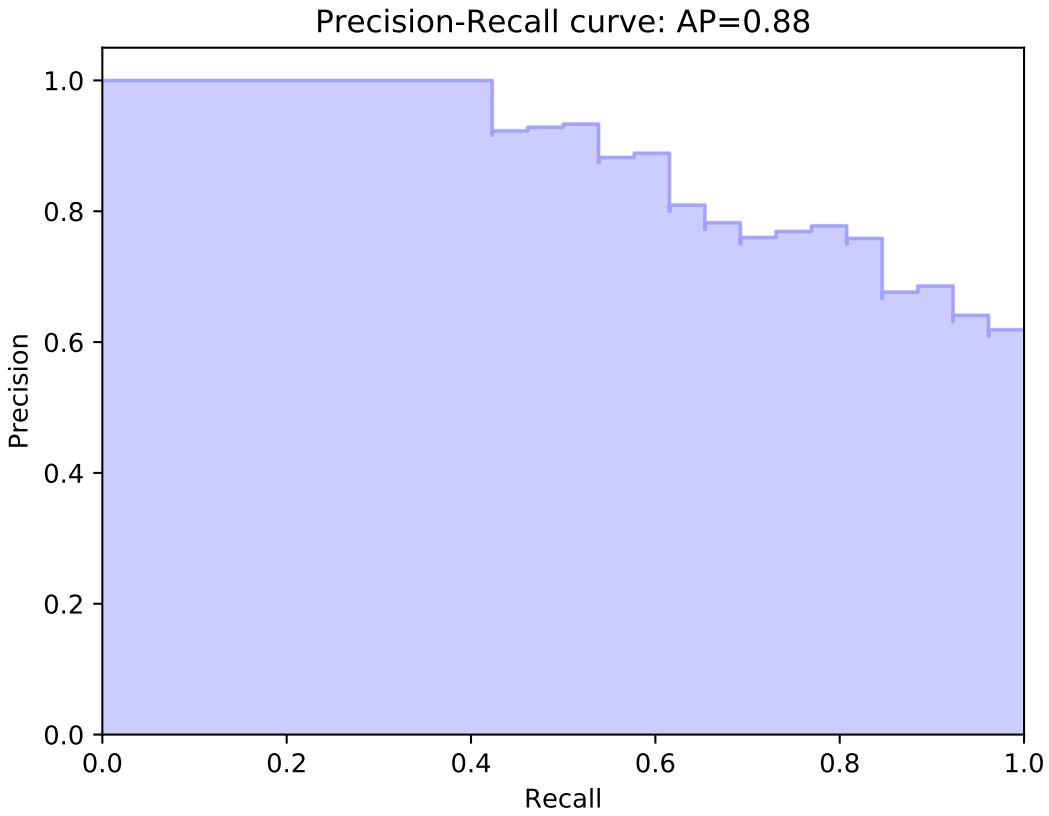


FIGURE A.2: AUC of Precision/Recall curve, namely, average precision (AP).

It is also common to calculate the AUC at several x-axis (e.g., recall) levels. For example, the Pascal VOC challenge (Everingham et al., 2010) measures the average precision at a set of eleven equally spaced recall levels $[0, 0.1, ..., 1]$. The Caltech pedestrian detection (Dollar et al., 2012) calculates the miss rate at nine FPPI rates in the range from 10^{-2} to 10^{0}.

A.4 Average Precision and Miss Rate

We used two metrics to evaluate the performance of object detection methods in Section 4.5 and Section 5.4. The definition of miss rate is

$$MissRate = \frac{FN}{FN + TP} \tag{A.6}$$

[1] http://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html

In general, the miss rate (or log-average miss rate) and average precision are similar. AP considers the precision in relation to the recall value. Some benchmarks, such as Pascal VOC 2007 and COCO (Lin et al., 2014), use AP as the metric for performance measurement. It is equivalent to the AUC under the Precision/Recall curve as discussed in appendix Section A.3. Note that the AUC of the ROC curve was also used in the Pascal VOC 2006 (Everingham et al., 2006), but it was later replaced by the AUC of the Precision/Recall curve to improve the sensitivity of the metric. The miss rate represents the relationship between miss rate and FPPI. Although the shapes of the two curves are generally similar, the smaller the value of miss rate, the better the detector.

Appendix B

Deep learning classifier

In this appendix, we review two common classifiers used in CNN. Although the proposed method often uses the softmax classifier only, it is good practice to consider training with both classifiers.

B.1 Softmax classifier

Deep learning classifier often uses the softmax classifier at the last layer of the network. In many deep learning frameworks (e.g., Caffe), the softmax classifier is a layer that is attached after the fully-connected layer. The softmax classifier gives normalized class probabilities as output and also has a probabilistic interpretation. The softmax layer outputs a $1 \times K$ output vector where K is the number of classes. The softmax layer has K nodes denoted by p_i where $i = 1, ..., K$, p_i is a discrete probability distribution. We denote a correct label y_i given the image x_i. Let h be the activation of the last fully-connected layer, W be the weight that connects the softmax layer and the fully-connected layer, and the input of the softmax layer be a vector $\mathbf{a}$ where

$$a_i = \sum_k h_k W_{ki} \tag{B.1}$$

The probability

$$p_i = P(y_i|x_i; W) = \frac{exp(a_i)}{\sum_j^K exp(a_j)} \tag{B.2}$$

can be interpreted as the normalized probability of the correct label y_i given the image x_i and parameterized by W.

To predict the class $\hat{i}$, the classifier uses the following formula:

$$\begin{aligned} \hat{i} &= \underset{i}{\text{argmax}}\, p_i \\ &= \underset{i}{\text{argmax}}\, a_i \end{aligned} \tag{B.3}$$

B.2 Support Vector Machine (SVM) classifier

Support vector machine (SVM) is a supervised learning model that is widely used for classification and regression analysis.

$$\begin{aligned} & \min_{\gamma,w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i \\ s.t. \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, \quad i = 1, ..., m \\ & \xi_i \geq 0, \quad i = 1, ..., m \end{aligned} \tag{B.4}$$

where ξ_i is a slow down variable that penalizes data points that violate the margin requirements. The parameter C is used to control the relative weighting between the dual goals of making the $\|w\|^2$ small and of ensuring that most examples have a functional margin of at least 1 (Ng, 2000). To use deep learning with SVM, Tang, 2013 used a differentiable variation of SVM known as L2-SVM, which minimizes the square hinge loss

$$\min_{w} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} max(1 - w^T x_n t_n, 0)^2 \tag{B.5}$$

where $t_n \in \{-1, +1\}$. To predict the class label of test data x

$$\underset{t}{\text{argmax}}(w^t x)t \tag{B.6}$$

A common approach to using SVM for multiple-class classification is to use a combination of several binary SVM classifiers. For multiple-class SVM, we can extend the binary SVM classifier to distinguish between either:

- One of the labels and the rest (one-versus-all) using a winner-takes-all strategy, or
- Every pair of classes (one-versus-one) implemented by max-wins voting.

Finally, the class and score prediction are the same as the prediction by the softmax classifier as in Section B.3.

Appendix C

Training box regression

The object detector must predict both the classes and the object locations (bounding boxes). The predicted object's bounding box is represented as four numbers $\{x_1, y_1, x_2, y_2\}$, which specify the top-left and bottom-right of the bounding box. Another representation of the bounding box is $\{x_1, y_1, w, h\}$, which specifies the top-left and the size of the bounding box (width and height). We also use the notation $\{x_c, y_c, w, h\}$ where x_c, y_c are the coordinates of the center of the predicted bounding box. The detector predicts the bounding boxes based on the features extracted by the CNN. The detector uses the "raw" data for training. Another approach to address this problem is box encoding, by which the predicted bounding box can be calculated by two inputs: the associate ground-truth box and the box offset.

In this appendix, we review several box encoding methods that have been used in recent deep-learning-based object detectors.

C.1 Box encoding

For each proposal box p (anchor box, default box), there is a best matching ground-truth box g (if at least one matching ground-truth box exists). The proposal box p is labeled positive and p is encoded with respect to the ground-truth box. The detector learns the transform function that maps a proposal box p to a ground-truth box g.

We call these transform functions $d_x(p), d_y(p), d_w(p)$, and $d_h(p)$. For example, in the R-CNN method (Girshick et al., 2014), the two first functions specify a scale-invariant translation of the center of p's bounding box, whereas the second two specify log-space translations of the width and height of p's bounding box. The bounding box is predicted by applying the transformation

$$\hat{g}_x = p_w d_x(p) + p_x \tag{C.1}$$

$$\hat{g}_y = p_h d_y(p) + p_y \tag{C.2}$$

$$\hat{g}_w = p_w exp(d_w(p)) \tag{C.3}$$

$$\hat{g}_h = p_h exp(d_h(p)) \tag{C.4}$$

Each function $d_*(p)$ (where $*$ is one of x, y, w, h) can be modeled as a linear function of CNN features, denoted $\phi(p)$. Thus, we have $d_*(p) = w_*^T \phi(p)$, where w_*^T are parameters needed for training. The box loss function can be

modeled as

$$L_{box}(t_* - w_*^T \phi(p)) \tag{C.5}$$

The term t in equation C.5 is called the "bounding box target" or "encoded box". For instance, the encoded box of the R-CNN method is defined as

$$t_x = (g_x - p_x)/p_w \tag{C.6}$$

$$t_y = (g_y - p_y)/p_h \tag{C.7}$$

$$t_w = log(g_w/p_w) \tag{C.8}$$

$$t_h = log(g_h/p_h) \tag{C.9}$$

Table C.1

TABLE C.1: Box encoding methods.

Method	Encoded box	Location loss functions
Szegedy et al., 2014	$[x_1, y_1, x_2, y_2]$	L_2
Faster R-CNN	$[(x - x_p)/w_a, (y - y_p)/h_p, log(w/w_p), log(h/h_p)]$	$SmoothL_1$
Redmon et al., 2016	$[x_1, y_1, \sqrt{w}, \sqrt{h}]$	L_2

$$\begin{aligned} t_x &= (x - x_p)/\sqrt{w_p * h_p} \\ t_y &= (y - y_p)/\sqrt{w_p * h_p} \\ t_w &= log(\sqrt{w * h}/\sqrt{w_p * h_p}) \\ t_h &= log(\sqrt{w * h}/\sqrt{w_p * h_p}) \end{aligned} \tag{C.10}$$

Other encoding is shown in equation C.10. This box encoding is used in the Faster R-CNN method to detect objects which tend to be square (e.g., faces), and when the input images aspect ratios are preserved via resizing.

C.2 Training objective of the box regression

We review two loss functions used to train the bounding box regression. From equation C.5, total regression loss is defined as

$$L_{box} = \sum_{i \in \{x,y,w,h\}} \ell(t_i - w_i^T \phi(p)) \tag{C.11}$$

where ℓ is defined as robust L_1 (smooth L_1)or L_2 loss functions, where

$$Smooth_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \tag{C.12}$$

Bibliography

Abadi, Martín et al. (2016). "TensorFlow: A System for Large-Scale Machine Learning." In: *OSDI*. Vol. 16, pp. 265–283.

Bell, Sean et al. (2016). "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*, pp. 2874–2883.

Benenson, Rodrigo et al. (2012). "Pedestrian detection at 100 frames per second". In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, pp. 2903–2910.

Benenson, Rodrigo et al. (2013). "Seeking the strongest rigid detector". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3666–3673.

Benenson, Rodrigo et al. (2014). "Ten years of pedestrian detection, what have we learned?" In: *European Conference on Computer Vision*. Springer, pp. 613–627.

Bengio, Yoshua (2012). "Practical recommendations for gradient-based training of deep architectures". In: *Neural networks: Tricks of the trade*. Springer, pp. 437–478.

Cai, Zhaowei, Mohammad Saberian, and Nuno Vasconcelos (2015). "Learning complexity-aware cascades for deep pedestrian detection". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3361–3369.

Cai, Zhaowei et al. (2016). "A unified multi-scale deep convolutional neural network for fast object detection". In: *European Conference on Computer Vision*. Springer, pp. 354–370.

Canny, John (1986). "A computational approach to edge detection". In: *IEEE Transactions on pattern analysis and machine intelligence* 6, pp. 679–698.

CireşAn, Dan et al. (2012). "Multi-column deep neural network for traffic sign classification". In: *Neural networks* 32, pp. 333–338.

Dai, Jifeng, Kaiming He, and Jian Sun (2016). "Instance-aware semantic segmentation via multi-task network cascades". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3150–3158.

Dai, Jifeng et al. (2016). "R-fcn: Object detection via region-based fully convolutional networks". In: *Advances in neural information processing systems*, pp. 379–387.

Dai, Jifeng et al. (2017). "Deformable convolutional networks". In: *CoRR, abs/1703.06211* 1.2, p. 3.

Dalal, Navneet and Bill Triggs (2005a). "Histograms of oriented gradients for human detection". In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE, pp. 886–893.

— (2005b). "INRIA person dataset". In: *Online: http://pascal.inrialpes.fr/data/human*.

Daniel Costea, Arthur and Sergiu Nedevschi (2016). "Semantic channels for fast pedestrian detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2360–2368.

Deng, Jia et al. (2009). "Imagenet: A large-scale hierarchical image database". In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. Ieee, pp. 248–255.

Dollár, Piotr et al. (2009). "Integral channel features". In:

Dollar, Piotr et al. (2012). "Pedestrian detection: An evaluation of the state of the art". In: *IEEE transactions on pattern analysis and machine intelligence* 34.4, pp. 743–761.

Dollár, Piotr et al. (2014). "Fast pyramids for object detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.8, pp. 1532–1545.

Du, Xianzhi et al. (2017). "Fused DNN: A deep neural network fusion approach to fast and robust pedestrian detection". In: *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*. IEEE, pp. 953–961.

Erhan, Dumitru et al. (2014). "Scalable object detection using deep neural networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2147–2154.

Ess, Andreas et al. (2008). "A mobile vision system for robust multi-person tracking". In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, pp. 1–8.

Everingham, Mark et al. (2006). "The pascal visual object classes challenge 2006 (voc 2006) results". In:

Everingham, Mark et al. (2010). "The pascal visual object classes (voc) challenge". In: *International journal of computer vision* 88.2, pp. 303–338.

Felzenszwalb, Pedro, David McAllester, and Deva Ramanan (2008). "A discriminatively trained, multiscale, deformable part model". In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, pp. 1–8.

Felzenszwalb, Pedro F and Daniel P Huttenlocher (2005). "Pictorial structures for object recognition". In: *International journal of computer vision* 61.1, pp. 55–79.

Felzenszwalb, Pedro F et al. (2010). "Object detection with discriminatively trained part-based models". In: *IEEE transactions on pattern analysis and machine intelligence* 32.9, pp. 1627–1645.

Fischler, Martin A and Robert A Elschlager (1973). "The representation and matching of pictorial structures". In: *IEEE Transactions on computers* 100.1, pp. 67–92.

Freund, Yoav, Robert E Schapire, et al. (1996). "Experiments with a new boosting algorithm". In: *Icml*. Vol. 96. Citeseer, pp. 148–156.

Friedman, Jerome, Trevor Hastie, Robert Tibshirani, et al. (2000). "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)". In: *The annals of statistics* 28.2, pp. 337–407.

Fu, Cheng-Yang et al. (2017). "DSSD: Deconvolutional single shot detector". In: *arXiv preprint arXiv:1701.06659*.

Geiger, Andreas, Philip Lenz, and Raquel Urtasun (2012). "Are we ready for autonomous driving? the kitti vision benchmark suite". In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, pp. 3354–3361.

Girshick, Ross (2015). "Fast r-cnn". In: *arXiv preprint arXiv:1504.08083*.

Girshick, Ross et al. (2014). "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587.

Girshick, Ross B, Pedro F Felzenszwalb, and David McAllester (2012). "Discriminatively trained deformable part models, release 5". In:

Glorot, Xavier and Yoshua Bengio (2010). "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256.

Gong, Yunchao et al. (2014). "Compressing deep convolutional networks using vector quantization". In: *arXiv preprint arXiv:1412.6115*.

Gregor, Karol et al. (2015). "Draw: A recurrent neural network for image generation". In: *arXiv preprint arXiv:1502.04623*.

Han, Song et al. (2015). "Learning both weights and connections for efficient neural network". In: *Advances in neural information processing systems*, pp. 1135–1143.

Hariharan, Bharath et al. (2015). "Hypercolumns for object segmentation and fine-grained localization". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 447–456.

He, Kaiming et al. (2014). "Spatial pyramid pooling in deep convolutional networks for visual recognition". In: *European conference on computer vision*. Springer, pp. 346–361.

— (2015). "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034.

— (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

He, Kaiming et al. (2017). "Mask r-cnn". In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, pp. 2980–2988.

He, Yihui, Xiangyu Zhang, and Jian Sun (2017). "Channel pruning for accelerating very deep neural networks". In: *International Conference on Computer Vision (ICCV)*. Vol. 2. 6.

Hoiem, Derek, Yodsawalai Chodpathumwan, and Qieyun Dai (2012). "Diagnosing error in object detectors". In: *European conference on computer vision*. Springer, pp. 340–353.

Hosang, Jan et al. (2015). "Taking a deeper look at pedestrians". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4073–4082.

Huang, Jonathan et al. (2017). "Speed/accuracy trade-offs for modern convolutional object detectors". In: *IEEE CVPR*.

Ioffe, Sergey and Christian Szegedy (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167*.

Jaderberg, Max, Karen Simonyan, Andrew Zisserman, et al. (2015). "Spatial transformer networks". In: *Advances in neural information processing systems*, pp. 2017–2025.

Kim, Kye-Hyeon et al. (2016). "PVANET: deep but lightweight neural networks for real-time object detection". In: *arXiv preprint arXiv:1608.08021*.

Kong, Tao et al. (2016). "Hypernet: Towards accurate region proposal generation and joint object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 845–853.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*, pp. 1097–1105.

LeCun, Yann et al. (1990). "Handwritten digit recognition with a back-propagation network". In: *Advances in neural information processing systems*, pp. 396–404.

LeCun, Yann et al. (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.

Li, Jianan et al. (2018). "Scale-aware fast R-CNN for pedestrian detection". In: *IEEE Transactions on Multimedia* 20.4, pp. 985–996.

Li, Yi et al. (2016). "Fully convolutional instance-aware semantic segmentation". In: *arXiv preprint arXiv:1611.07709*.

Lin, Tsung-Yi et al. (2014). "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer, pp. 740–755.

Lin, Tsung-Yi et al. (2017). "Feature pyramid networks for object detection". In: *CVPR*. Vol. 1. 2, p. 4.

Lin, TY and P Dollar (2016). *Ms coco api*.

Linh, Tran Duy and Masayuki Arai (2017). "A two-stage training deep neural network for small pedestrian detection". In: *Machine Learning for Signal Processing (MLSP), 2017 IEEE 27th International Workshop on*. IEEE, pp. 1–6.

Liu, Wei, Andrew Rabinovich, and Alexander C Berg (2015). "Parsenet: Looking wider to see better". In: *arXiv preprint arXiv:1506.04579*.

Liu, Wei et al. (2016). "Ssd: Single shot multibox detector". In: *European conference on computer vision*. Springer, pp. 21–37.

Long, Jonathan, Evan Shelhamer, and Trevor Darrell (2015). "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440.

Lowe, David G (1999). "Object recognition from local scale-invariant features". In: *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*. Vol. 2. Ieee, pp. 1150–1157.

Mayer, Nikolaus et al. (2016). "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4040–4048.

Miller, George (1998). *WordNet: An electronic lexical database*. MIT press.

Ng, Andrew (2000). "CS229 Lecture notes". In: *CS229 Lecture notes* 1.1, pp. 1–3.

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

Park, Dennis et al. (2013). "Exploring weak stabilization for motion feature extraction". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2882–2889.

Poirson, Patrick et al. (2016). "Fast single shot detection and pose estimation". In: *3D Vision (3DV), 2016 Fourth International Conference on*. IEEE, pp. 676–684.

Redmon, Joseph and Ali Farhadi (2017). "YOLO9000: better, faster, stronger". In: *arXiv preprint*.

Redmon, Joseph et al. (2016). "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788.

Ren, Shaoqing et al. (2015). "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*, pp. 91–99.

Ruder, Sebastian (2017). "An overview of multi-task learning in deep neural networks". In: *arXiv preprint arXiv:1706.05098*.

Russakovsky, Olga et al. (2015). "Imagenet large scale visual recognition challenge". In: *International Journal of Computer Vision* 115.3, pp. 211–252.

Sermanet, Pierre et al. (2013). "Overfeat: Integrated recognition, localization and detection using convolutional networks". In: *arXiv preprint arXiv:1312.6229*.

Shrivastava, Abhinav and Abhinav Gupta (2016). "Contextual priming and feedback for faster r-cnn". In: *European Conference on Computer Vision*. Springer, pp. 330–348.

Shrivastava, Abhinav, Abhinav Gupta, and Ross Girshick (2016). "Training region-based object detectors with online hard example mining". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 761–769.

Simonyan, Karen and Andrew Zisserman (2014). "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556*.

Srinivas, Suraj and R Venkatesh Babu (2015). "Data-free parameter pruning for deep neural networks". In: *arXiv preprint arXiv:1507.06149*.

Srivastava, Nitish et al. (2014). "Dropout: a simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1, pp. 1929–1958.

Szegedy, Christian et al. (2014). "Scalable, high-quality object detection". In: *arXiv preprint arXiv:1412.1441*.

Szegedy, Christian et al. (2015). "Going deeper with convolutions". In: Cvpr.

Szegedy, Christian et al. (2016). "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826.

Szegedy, Christian et al. (2017). "Inception-v4, inception-resnet and the impact of residual connections on learning." In: *AAAI*. Vol. 4, p. 12.

Taigman, Yaniv et al. (2014). "Deepface: Closing the gap to human-level performance in face verification". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1701–1708.

Tang, Yichuan (2013). "Deep learning using linear support vector machines". In: *arXiv preprint arXiv:1306.0239*.

Tian, Yonglong et al. (2015). "Deep learning strong parts for pedestrian detection". In: *Proceedings of the IEEE international conference on computer vision*, pp. 1904–1912.

Uijlings, Jasper RR et al. (2013). "Selective search for object recognition". In: *International journal of computer vision* 104.2, pp. 154–171.

Viola, Paul and Michael Jones (2001). "Rapid object detection using a boosted cascade of simple features". In: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. Vol. 1. IEEE, pp. I–I.

Viola, Paul and Michael J Jones (2004). "Robust real-time face detection". In: *International journal of computer vision* 57.2, pp. 137–154.

Viola, Paul, Michael J Jones, and Daniel Snow (2003). "Detecting pedestrians using patterns of motion and appearance". In: *null*. IEEE, p. 734.

Wojek, Christian and Bernt Schiele (2008). "A performance evaluation of single and multi-feature people detection". In: *Joint Pattern Recognition Symposium*. Springer, pp. 82–91.

Wojek, Christian, Stefan Walk, and Bernt Schiele (2009). "Multi-cue onboard pedestrian detection". In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, pp. 794–801.

Xie, Saining et al. (2017). "Aggregated residual transformations for deep neural networks". In: *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, pp. 5987–5995.

Yan, Junjie et al. (2013). "Robust multi-resolution pedestrian detection in traffic scenes". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3033–3040.

Yang, Bin et al. (2014). "Aggregate channel features for multi-view face detection". In: *Biometrics (IJCB), 2014 IEEE International Joint Conference on*. IEEE, pp. 1–8.

— (2015). "Convolutional channel features". In: *Proceedings of the IEEE international conference on computer vision*, pp. 82–90.

— (2016). "Craft objects from images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6043–6051.

Zagoruyko, Sergey et al. (2016). "A multipath network for object detection". In: *arXiv preprint arXiv:1604.02135*.

Zhai, Andrew et al. (2017). "Visual discovery at pinterest". In: *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, pp. 515–524.

Zhang, Liliang et al. (2016). "Is faster r-cnn doing well for pedestrian detection?" In: *European Conference on Computer Vision*. Springer, pp. 443–457.