Object Recognition Based on Deep Learning

Thesis · June 2019

CITATIONS
0

READS
164

1 author:

Md Rabbi Al Islam
Nanjing University of Posts and Telecommunications
1 PUBLICATION **0** CITATIONS

SEE PROFILE

南京邮电大学

Nanjing University of Posts and Telecommunications

毕 业 设 计 (论 文)

Graduation project (Thesis)

题 目/Title	Object Recognition Based on Deep Learning
专 业 /Student's Major	Electronics and Information Engineering
学生姓名/Student Name	Md Rabbi Al Islam
班级学号/Student Number	F15010106
指导教师 /Supervisor	Xie Shi Peng
指导单位 /Supervisor's College	College of Information and Communication

Duration: 2018 /12 /01 to 2019 /06 /10

Object Recognition Based on Deep Learning

Undergraduate Project (Thesis) at the

School of Information & Communication Engineering

By

Md Rabbi Al Islam

F15010106

Electronic Information Engineering-2015

Supervisor: Xie Shi Peng

June 2019

Nanjing University of Posts & Telecommunications

Nanjing, China.

毕业设计（论文）原创性声明

Originality Declaration of Graduation Design (Thesis)

本人郑重声明：所提交的毕业设计（论文），是本人在导师指导下，独立进行研究工作所取得的成果。除文中已注明引用的内容外，本毕业设计（论文）不包含任何其他个人或集体已经发表或撰写过的作品成果。对本研究做出过重要贡献的个人和集体，均已在文中以明确方式标明并表示了谢意。

I hereby solemnly declare that the submitted graduation project (thesis) is the outcome of my own research work under the guidance of my supervisor. Except for the content quoted in the text, this thesis does not contain any research findings written or published by other individuals or teams. All the contributions of the research from the individuals or teams worked with me have been clearly stated and acknowledged.

论文作者签名/ Author's signature: 

日期 Date： 2019 年 06 月 10 日

Abstract

Due to object detection's close relationship with video analysis and image understanding, it has attracted much research attention in recent years. Traditional object detection methods are built on handcrafted features and shallow trainable architectures. Their performance easily stagnates by constructing complex ensembles, which combine multiple low-level image features with high-level context from object detectors and scene classifiers. With the rapid development in deep learning, more powerful tools, which are able to learn semantic, high-level, deeper features, are introduced to address the problems existing in traditional architectures. These models behave differently in network architecture, training strategy and optimization function, etc. In this paper, we provide a review on deep learning based object detection frameworks. Our review begins with a brief introduction on the history of deep learning and its representative tool, namely Convolutional Neural Network (CNN). Then we focus on typical generic object detection architectures along with some modifications and useful tricks to improve detection performance further. As distinct specific detection tasks exhibit different characteristics. Experimental analyses are also provided to compare various methods and draw some meaningful conclusions. Finally, several promising directions and tasks are provided to serve as guidelines for future work in both object detection and relevant neural network based learning systems.

Index Terms— Deep learning, Object detection, Neural network.

Contents

List of Figures

Meaning of the abbreviation

CNN	Convolutional Neural Network
R-CNN	Region based- Convolutional Neural Network
FR-CNN	Fast Region based Convolutional Neural Network
FR-CNN	Faster Region based Convolutional Neural Network
SS	Selective Search
HOG	Histograms of Oriented Gradients
SVM	Support Vector Machine
RPN	Region Proposal Network
FCN	Fully Convolutional Network
DPM	Deformable part-based model
LFW	Labeled Faces in the wild
DNN	Deep Neural Network
SGD	Stochastic gradient descent
ILSVRC	Large scale visual recognition challenge
FC	Fully connected
RFCN	Region based Fully Convolutional Network
YOLO	You only look once
SVD	Singular value decomposition
NMS	Non-maximum suppression
SPP	Spatial pyramid pooling
RPN	Region proposal network
BN	Batch normalization
Deconv layers	Deconvolution layers
GOP	Geodesic object proposals
MCG	Multiscale combinatorial grouping
SPM	Spatial pyramid matching
FPN	Feature pyramid network
SDP	Scale dependent pooling
CRC	Cascaded rejection classifiers
MR-CNN	Multi Region based Convolutional Neural Network
ION	Inside-outside net

G-CNN	Grid based Convolutional Neural Network
Conv	Convolution
COCO	Common Objects in Context
SSD	Single Shot Detector
EB	Edge Boxes
VGG	Visual Geometry Group
RoI	Region of interest

Chapter 1 Introduction

Recent advances in object detection are driven by the success of region proposal methods [7] and region-based convolutional neural networks (R-CNNs) [3]. Although region-based CNNs were computationally expensive as originally developed in [3], their cost has been drastically reduced thanks to sharing convolutions across proposals [4], [2]. The latest incarnation, Fast R-CNN [2], achieves near real-time rates using very deep networks [6], when ignoring the time spent on region proposals. Now, proposals are the computational bottleneck in state-of-the-art detection systems.

Region proposal methods typically rely on inexpensive features and economical inference schemes. Selective Search (SS) [7], one of the most popular methods, greedily merges super pixels based on engineered low-level features. Yet when compared to efficient detection networks [2], Selective Search is an order of magnitude slower, at 2s per image in a CPU implementation. EdgeBoxes [8] currently provides the best tradeoff between proposal quality and speed, at 0.2s per image. Nevertheless, the region proposal step still consumes as much running time as the detection network.

One may note that fast region-based CNNs take advantage of GPUs, while the region proposal methods used in research are implemented on the CPU, making such runtime comparisons inequitable. An obvious way to accelerate proposal computation is to re-implement it for the GPU. This may be an effective engineering solution, but re-implementation ignores the down-stream detection network and therefore misses important opportunities for sharing computation.

In this paper, we show that an algorithmic change computing proposals with a deep net leads to an elegant and effective solution, where proposal computation is nearly cost-free given the detection network’s computation. To this end, we introduce novel Region Proposal Networks (RPNs) that share convolutional layers with state-of-the-art object detection networks [4], [2]. By sharing convolutions attest-time, the marginal cost for computing proposals is small (e.g., 10ms per image).

Our observation is that the convolutional (conv) feature maps used by region-based detectors, like Fast R-CNN, can also be used for generating region proposals. On top of these conv features, we construct RPNs by adding two additional conv layers: one that encodes each conv map position into a short (e.g., 256-d) feature vector and a second that, at each conv map position, outputs an objectness score and regressed bounds for k region proposals relative to various scales and aspect ratios at that location (k= 9 is a typical value).

Our RPNs are thus a somewhat fully convolutional network (FCN) [5] and they can be trained end-to-end specifically for the task for generating detection proposals. To unify RPNs with Fast R-

CNN [2] object detection networks, we propose a simple training scheme that alternates between fine-tuning for the region proposal task and then fine-tuning for object detection, while keeping the proposals fixed. This scheme converges quickly and produces a unified network with conv features that are shared between both tasks.

RPNs are designed to efficiently predict region proposals with a wide range of scales and aspect ratios. In contrast to prevalent methods that use

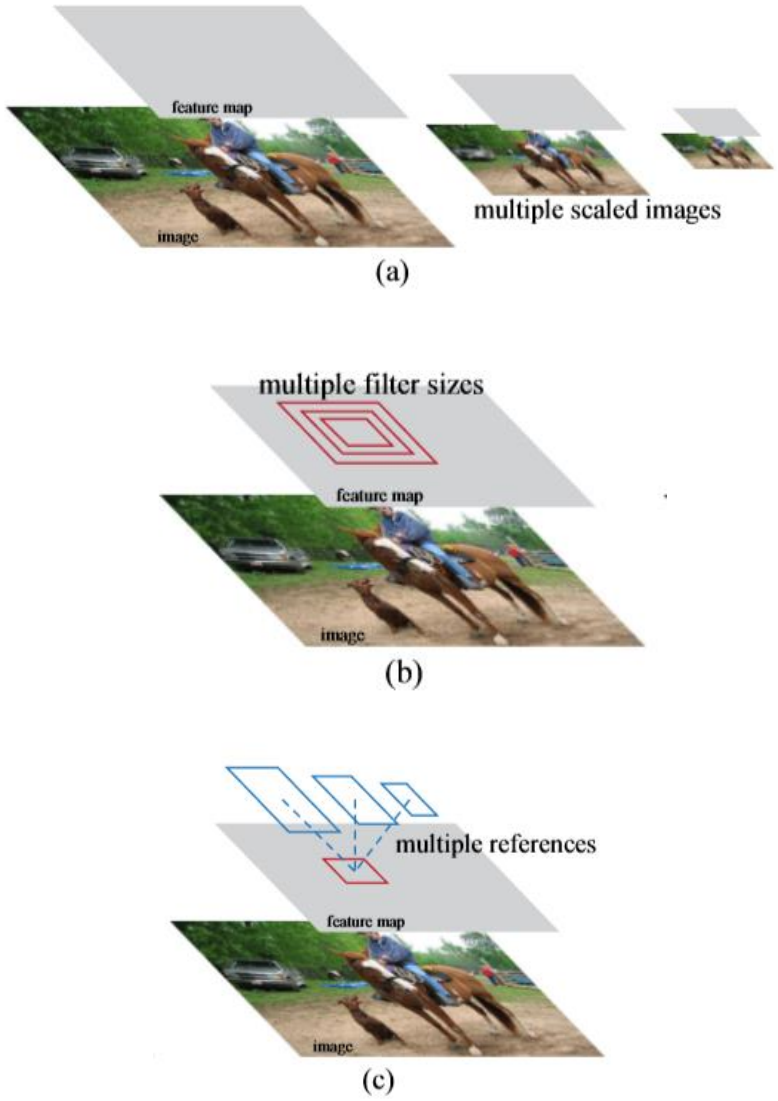

Figure 1.1: Different schemes for addressing multiple scales and sizes. (a) Pyramids of images and feature maps are built, and the classifier is run at all scales. (b) Pyramids of filters with multiple

scales/sizes are run on the feature map. (c) We use pyramids of reference boxes in the regression functions.

Pyramids of images (Figure 1.1, a) or pyramids of filters (Figure 1.1, b), we introduce novel "anchor" boxes that serve as references at multiple scales and aspect ratios. Our scheme can be thought of as a pyramid of regression references (Figure 1.1, c), which avoids enumerating images or filters of multiple scales or aspect ratios. This model performs well when trained and tested using single-scale images and thus benefits running speed. To unify RPNs with Fast R-CNN object detection networks, we propose a training scheme that alternates between fine-tuning for the region proposal task and then fine-tuning for object detection, while keeping the proposals fixed.

This scheme converges quickly and produces a unified network with convolutional features that are shared between both tasks. We comprehensively evaluate our method on the PASCAL VOC detection benchmarks where RPNs with Fast R-CNNs produce detection accuracy better than the strong baseline of Selective Search with Fast R-CNNs. Meanwhile, our method waives nearly all computational burdens of Selective Search at test-time—the effective running time for proposals is just 10 milliseconds. Using the expensive very deep models of Faster R-CNN, our detection method still has a frame rate of 5fps (including all steps) on a GPU, and thus is a practical object detection system in terms of both speed and accuracy.
We also report results on the MS COCO dataset and investigate the improvements on PASCAL VOC using the COCO data. Our fast and effective object detection system has also been built in commercial systems such as at Pinterest's, with user engagement improvements reported. In ILSVRC and COCO 2015 competitions, Faster R-CNN and RPN are the basis of several 1st-place entries in the tracks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation. RPNs completely learn to propose regions from data, and thus can easily benefit from deeper and more expressive features (such as the 101-layer residual nets adopted in [9]). Faster R-CNN and RPN are also used by several other leading entries in these competitions. These results suggest that our method is not only a cost-efficient solution for practical usage, but also an effective way of improving object detection accuracy.

We evaluate our method on the PASCAL VOC detection benchmarks [1], where RPNs with Fast R-CNNs produce detection accuracy better than the strong baseline of Selective Search with Fast R-CNNs. Meanwhile, our method waives nearly all computational burdens of SS at test-time—the effective running time for proposals is just 10 milliseconds. Using the expensive very deep models of [6], our detection method still has a frame rate of 5fps (including all steps) on a GPU, and thus is a practical object detection system in terms of both speed and accuracy (73.2% mAP on PASCAL VOC 2007 and 70.4% mAP on 2012).

1.1 Background

To gain a complete image understanding, we should not only concentrate on classifying different images, but also try to precisely estimate the concepts and locations of objects contained in each image. This task is referred as object detection, which usually consists of different subtasks such as face detection, pedestrian detection and skeleton detection. As one of the fundamental computer vision problems, object detection is able to provide valuable information for semantic understanding of images and videos, and is related to many applications, including image classification, human behavior analysis, face recognition and autonomous driving. Meanwhile, Inheriting from neural networks and related learning systems, the progress in these fields will develop neural network algorithms, and will also have great impacts on object detection techniques, which can be considered as learning systems. However, due to large variations in viewpoints, poses, occlusions and lighting conditions, it's difficult to perfectly accomplish object detection with an additional object localization task. So much attention has been attracted to this field in recent years. The problem definition of object detection is to determine where objects are located in a given image (object localization) and which category each object belongs to (object classification). Therefore, the pipeline of traditional object detection models can be mainly divided into three stages: informative region selection, feature extraction and classification.

I. Informative Region Selection

As different objects may appear in any positions of the image and have different aspect ratios or sizes, it is a natural choice to scan the whole image with a multi-scale sliding window. Although this exhaustive strategy can find out all possible positions of the objects, its shortcomings are also obvious. Due to a large number of candidate windows, it is computationally expensive and produces too many redundant windows. However, if only a fixed number of sliding window templates are applied, unsatisfactory regions may be produced.

II. Feature Extraction

To recognize different objects, we need to extract visual features, which can provide a semantic and robust representation. SIFT [23], HOG [24] and Haar-like [25] features are the representative ones. This is due to the fact that these features can produce representations associated with complex cells in human brain [23]. However, due to the diversity of appearances, illumination conditions and backgrounds, it's difficult to manually design a robust feature descriptor to perfectly describe all kinds of objects.

III. Classification

Besides, a classifier is needed to distinguish a target object from all the other categories and to make the representations more hierarchical, semantic and informative for visual recognition. Usually, the Supported Vector Machine (SVM) [26], AdaBoost [27] and Deformable Part-based Model (DPM) [28] are good choices. Among these classifiers, the DPM is a flexible model by combining object parts with deformation cost to handle severe deformations. In DPM, with the aid of a graphical model, carefully designed low-level features and kinematically inspired part decompositions are combined. Moreover, discriminative learning of graphical models allows for building high-precision part-based models for a variety of object classes.

Based on these discriminant local feature descriptors and shallow learnable architectures, state of the art results have been obtained on PASCAL VOC object detection competition [1] and real-time embedded systems have been obtained with a low burden on hardware. However, small gains are obtained during 2010-2012 by only building ensemble systems and employing minor variants of successful methods [3]. This fact is due to the following reasons:

a) The generation of candidate bounding boxes with a sliding window strategy is redundant, inefficient and inaccurate.
b) The semantic gap cannot be bridged by the combination of manually engineered low-level descriptors and discriminatively-trained shallow models.

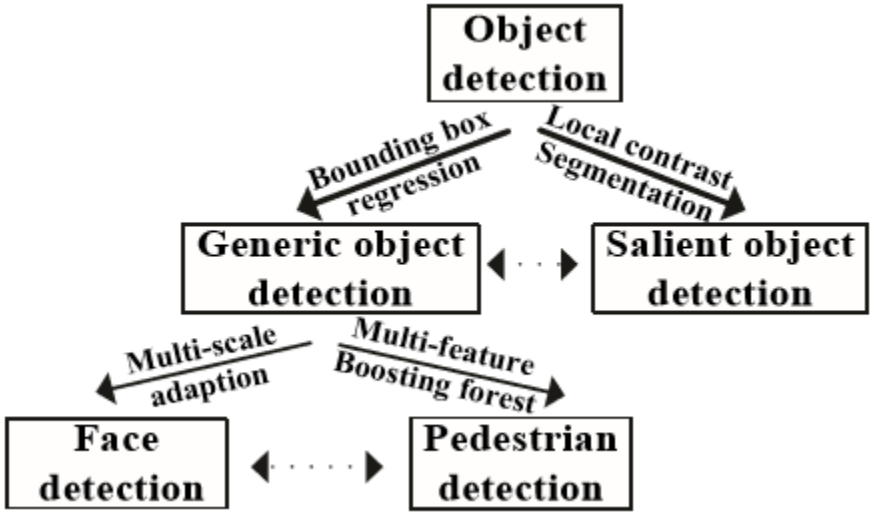

Figure 1.2: The application domains of object detection.

Thanks to the emergency of Deep Neural Networks (DNNs) [14], a more significant gain is obtained with the introduction of Regions with CNN features (R-CNN) [3]. DNNs, or the most representative CNNs, act in a quite different way from traditional approaches. They have deeper architectures with the capacity to learn more complex features than the shallow ones. In addition, the expressivity and robust training algorithms allow to learn informative object representations without the need to design features manually.
Since the proposal of R-CNN, a great deal of improved models have been suggested, including Fast R-CNN which jointly optimizes classification and bounding box regression tasks [2], Faster R-CNN which takes an additional subnetwork to generate region proposals [22] and YOLO which accomplishes object detection via a fixed-grid regression. All of them bring different degrees of detection performance improvements over the primary R-CNN and make real-time and accurate object detection become more achievable.

In this paper, a systematic review is provided to summarize representative models and their different characteristics in several application domains, including generic object detection [3], [2], [22], salient object detection, face detection and pedestrian detection. Their relationships are depicted in Figure 1.2. Based on basic CNN architectures, generic object detection is achieved with bounding box regression, while salient object detection is accomplished with local contrast enhancement and pixel-level segmentation. Face detection and pedestrian detection are closely related to generic object detection and mainly accomplished with multi-scale adaption and multi-feature fusion/boosting forest, respectively. The dotted lines indicate that the corresponding domains are associated with each other under certain conditions. It should be noticed that the covered domains are diversified. Pedestrian and face images have regular structures, while general objects and scene images have more complex variations in geometric structures and layouts. Therefore, different deep models are required by various images.

There has been a relevant pioneer effort [29], which mainly focuses on relevant software tools to implement deep learning techniques for image classification and object detection, but pays little attention on detailing specific algorithms. Different from it, our work not only reviews deep learning based object detection models and algorithms covering different application domains in detail, but also provides their corresponding experimental comparisons and meaningful analyses.

1.1.1 Object Detection Basic

Object detection is a computer technology related to vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. Well-researched domains of object detection include face detection and pedestrian detection. Object detection has applications in many areas of computer

vision, including retrieval and video surveillance. It is widely used in computer vision task such as face detection, face recognition, video object co-segmentation. It is also used in tracking objects, for example tracking a ball during a football match, tracking movement of a cricket bat, tracking a person in a video.
Every object class has its own special features that helps in classifying the class – for example, all circles are round. Object class detection uses these special features. For example, when looking for circles, objects that are at a particular distance from a point (i.e. the center) are sought. Similarly, when looking for squares, objects that are perpendicular at corners and have equal side lengths are needed. A similar approach is used for face identification where eyes, nose, and lips can be found and features like skin color and distance between eyes can be found.

The goal of object detection is to detect all instances of objects from a known class, such as people, cars or faces in an image. Typically, only a small number of instances of the object are present in the image, but there is a very large number of possible locations and scales at which they can occur and that need to somehow be explored. Each detection is reported with some form of pose information. This could be as simple as the location of the object, a location and scale, or the extent of the object defined in terms of a bounding box. In other situations, the pose information is more detailed and contains the parameters of a linear or non-linear transformation. For example, a face detector may compute the locations of the eyes, nose and mouth, in addition to the bounding box of the face. An example of a bicycle detection that specifies the locations of certain parts is shown in Figure 1.3. The pose could also be defined by a three-dimensional transformation specifying the location of the object relative to the camera.

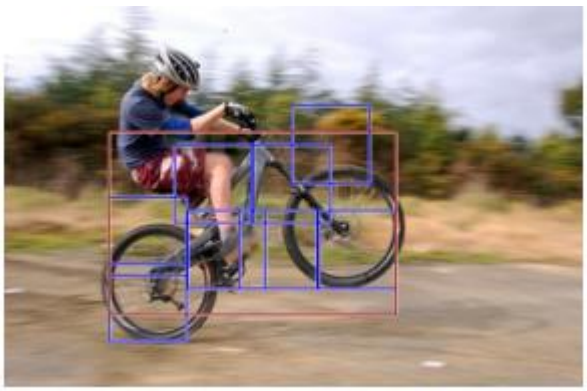

Figure 1.3: A bicycle detection specified in terms of the locations of certain parts.

Object detection systems construct a model for an object class from a set of training examples. In the case of a fixed rigid object only one example may be needed, but more generally multiple

training examples are necessary to capture certain aspects of class variability. Object detection methods fall into two major categories, generative and discriminative. The first consists of a probability model for the pose variability of the objects together with an appearance model: a probability model for the image appearance conditional on a given pose, together with a model for background, i.e. non-object images. The model parameters can be estimated from training data and the decisions are based on ratios of posterior probabilities. The second typically builds a classifier that can discriminate between images (or sub-images) containing the object and those not containing the object. The parameters of the classifier are selected to minimize mistakes on the training data, often with a regularization bias to avoid overfitting. Other distinctions among detection algorithms have to do with the computational tools used to scan the entire image or search over possible poses, the type of image representation with which the models are constructed, and what type and how much training data is required to build a model.

1.1.2 Basic Algorithm of Object Detection

How much time have you spent looking for lost room keys in an untidy and messy house? It happens to the best of us and till date remains an incredibly frustrating experience. Nevertheless, what if a simple computer algorithm could locate your keys in a matter of milliseconds?

That is the power of object detection algorithms. While this was a simple example, the applications of object detection span multiple and diverse industries, from round-the-clock surveillance to real-time vehicle detection in smart cities. In short, these are powerful deep learning algorithms. We will dive deeper and look at various algorithms that can be used for object detection. We will start with the algorithms belonging to RCNN family, i.e. RCNN, Fast RCNN and Faster RCNN.

a) CNN algorithm

Divides the image into multiple regions and then classify each region into various classes. Needs a lot of regions to predict accurately and hence high computation time.

b) R-CNN algorithm

Uses selective search to generate regions. Extracts around 2000 regions from each image. Prediction time per image 40 to 50 seconds. High computation time as each region is passed to the CNN separately also it uses three different model for making predictions.

c) Fast R-CNN algorithm

Each image is passed only once to the CNN and feature maps are extracted. Selective search is used on these maps to generate predictions. Combines all the three models used in RCNN together. Prediction time per image 2 seconds. Selective search is slow and hence computation time is still high.

d) Faster R-CNN algorithm

Replaces the selective search method with region proposal network, which made the algorithm much faster. Prediction time per image 0.2 seconds. Object proposal takes time and as there are different systems working one after the other, the performance of systems depends on how the previous system has performed.

1.2 Dataset and Neural Network

For deep learning, dataset and neural network are two important parts. The dataset is the fuel for deep learning so that the number and quality of the dataset will affect the accuracy of the neural network output, and the choice of neural network or the network architecture will also affect the accuracy.

1.2.1 Dataset

Dataset is one of the foundations of deep learning, for many researchers to get enough data to carry out the experiment just by themselves is a big problem, so we need a lot of open source dataset for everyone to use. Some commonly used datasets in computer vision is the following.

I. ImageNet

The ImageNet dataset [10] has more than 14 million images covering more than 20,000 categories. There are more than a million pictures with explicit class annotations and annotations of object locations in the image. The ImageNet dataset is one of the most widely used datasets in the field of deep learning. Most of the research work such as image classification, location, and detection is based on this dataset. The ImageNet dataset is detailed and is very easy to use. It is very widely used in the field of computer vision research, and has become the "standard" dataset of the current deep learning of image domain to test algorithm performance. There is a well-known challenge called "ImageNet International Computer Vision Challenge" (ILSVRC) [21] based on the ImageNet dataset.

II. PASCAL VOC

The PASCAL VOC (pattern analysis, statistical modelling and computational learning visual object classes) [1] provides standardized image data sets for object class recognition and provides a common set of tools for accessing the data sets and annotations. The PASCAL VOC dataset includes 20 classes and has a challenge based on this dataset. The PASCAL VOC Challenge is no longer available after 2012, but its dataset is of good quality and well-marked, and enables evaluation and comparison of different methods. Moreover, because the amount of data of the PASCAL VOC dataset is small, compared to the ImageNet dataset, very suitable for researchers to test network programs. Our dataset is also created based on the PASCAL VOC dataset standard.

III. COCO

COCO (Common Objects in Context) [9] is a new image recognition, segmentation, and captioning dataset, sponsored by Microsoft. COCO dataset has more than 300,000 images covering 80 object categories. The open source of this dataset makes great progress in semantic segmentation in recent years, and it has become a "standard" dataset for the performance of image semantic understanding, and also COCO has its own challenge.

1.2.2 Neural Network

Deep learning used by the network has been constantly improving, in addition to the changes in the network structure, the more is to do some tune based on the original network or apply some trick to make the network performance to enhance. The more well known algorithms of object detection are a series of algorithms based on R-CNN, mainly in the following.

I. RCNN

Paper, which the R-CNN (Regions with Convolutional Neural Network) is in, has been the state-of-art papers in field of object detection in 2014 years. The idea of this paper has changed the general idea of object detection. Later, algorithms in many literatures on deep learning of object detection inherited this idea, which is the core algorithm for object detection with deep learning. One of the most noteworthy points of this paper is that the CNN is applied to the candidate box to extract the feature vector, and the second is to propose a way to effectively train large CNNs. It is supervised pre-training on large dataset such ILSVRC, and then do some fine-tuning training in a specific range on a small dataset such PASCAL.

II. SSP-Net

SPP-Net [4] is an improvement based on the R-CNN with faster speed. SPP-Net proposed a spatial pyramid pooling (SPP) layer that removes restrictions on network fixed size. SPP-Net only needs to run the convolution layer once (the whole image, regardless of size), and then use the SPP layer to extract features, compared to the R-CNN, to avoid repeat convolution operation the candidate area, reducing the number of convolution times. The speed for SPP-Net calculating the convolution on the Pascal VOC 2007 dataset by 30-170 times faster than the R-CNN, and the overall speed is 24-64 times faster than the R-CNN.

III. Fast R-CNN

For the shortcomings of R-CNN and SPP-Net, Fast R-CNN [2] did the following improvements: higher detection quality (mAP) than R-CNN and SPP-Net; write the loss function of multiple tasks together to achieve single-level training process. For the shortcomings of R-CNN and SPP-Net, Fast R-CNN [2] did the following improvements: higher detection quality (mAP) than R-CNN and SPP-Net; write the loss function of multiple tasks together to achieve single-level training process, in the training can update all the layers, do not need to store features in the disk.

Fast R-CNN can improve the speed of training deeper neural networks, such as VGG16. Compared to R-CNN, The speed for Fast R-CNN training stage is 9 times faster and the speed for test is 213 times faster. The speed for Fast R-CNN training stage is 3 times faster than SPP-net and the speed for test is 10 times faster, the accuracy rate also have a certain increase.

IV. Faster R-CNN

The emergence of SPP-net and Fast R-CNN has greatly reduced the running time of the object detection network. However, the time they take for the regional proposal method is too long, and the task of getting regional proposal is a bottleneck. Faster R-CNN [22] presents a solution to this problem by converting traditional practices (such as Selective Search, SS) to use a deep network to compute a proposal box (such as Region Proposal Network, RPN). In this paper, our experiment chooses the Faster R-CNN network. Table shows the comparison of mean Average Precision (mAP) of above four kinds of network structure on the VOC 2007 dataset.

Network	R-CNN	SPP-Net	Fast R-CNN	Faster R-CNN
VOC07 mAP	0.66	0.631	0.669	0.732

Table: Mean average precision (mAP) of different network on the VOC2007 dataset

- **Chapter 2** related work, a brief introduction on the history of deep learning and the basic architecture of CNN is provided .
- **Chapter 3** in chapter 3, we describe the architecture of the RPN, generic object detection, optimization, sharing feature for RPN and Fast R-CNN.
- **Chapter 4** in chapter 4 we will describe about Experimental Evaluation.
- **Chapter 5** presents the summery of this thesis

Chapter 2 Literature review

2.1 Related Work

Several recent papers have proposed ways of using deep networks for locating class-specific or class-agnostic bounding boxes [30], [18], [3]. In the OverFeat method [18], a fully connected (fc) layer is trained to predict the box coordinates for the localization task that assumes a single object. The fc layer is then turned into a conv layer for detecting multiple class-specific objects. The MultiBox methods [31] generate region proposals from a network whose last fc layer simultaneously predicts multiple (e.g., 800) boxes, which are used for R-CNN [3], object detection. Their proposal network is applied on a single image or multiple large image crops (e.g., 224×224) [20]. We discuss OverFeat and MultiBox in more depth later in context with our method.

Shared computation of convolutions [18], [4], [2] has been attracting increasing attention for efficient, yet accurate, visual recognition. The OverFeat paper [18] computes conv features from an image pyramid for classification, localization, and detection. Adaptively sized pooling (SPP) [4] on shared conv feature maps is proposed for efficient region-based object detection [4], [33] and semantic segmentation [32]. Fast R-CNN [2] enables end-to-end detector training on shared conv features and shows compelling accuracy and speed.

2.2 A Brief Overview of Deep Learning

This topic will give an overview of the development of deep learning back to neural networks in 1940s, some high-impact results it has achieved since 2006, and the major differences between deep models and other machine learning models. It will also explain why neural networks were once given up by many researchers and why they became popular again since 2006. Prior to overview on deep learning based object detection approaches, we provide a review on the history of deep learning along with an introduction on the basic architecture and advantages of CNN.

2.2.1 Neural networks

Deep models are neural networks with deep structures. The history of neural networks can be traced back to the 1940s. It was inspired by simulating the human brain system and the goal was to find a principled way to solve general learning problems. It was popular in 1980s and 1990s. In 1986, Rumelhart, Hinton, and Williams published back propagation in Nature [17], and it has been widely used to train neural networks until now. In the following subsections, we will introduce the structure of multilayer neural networks, feedforward operation used to predict output from input,

and backward propagation. However, neural networks were eventually given up by most researchers because of multiple reasons that will be explained in Section 2.2.3.

2.2.2 Multilayer Neural Networks

The computation units of neural networks are called neurons and are organized into multiple layers. Neurons in adjacent layers are connected with weights. However, neurons in the same layer are not connected. In feedforward operation, neurons in a lower layer pass signals to neurons in its upper layer. A neuron is activated if it's received signals are strong enough. Similar to the brain, some connections between neurons are stronger, while some are weaker, indicated by different weights. Figure 2.1 shows an example of a three layer neural network with an input layer, a hidden layer, and an output layer. $\boldsymbol{X} = (\boldsymbol{x_1}, \ldots \boldsymbol{x_i}, \ldots \boldsymbol{x_d})$ is a $\boldsymbol{d}$ dimensional input data vector.

$\boldsymbol{h} = (\boldsymbol{h_1}, \ldots \boldsymbol{h_j}, \ldots \boldsymbol{h_k})$ are responses at $\boldsymbol{n_H}$ hidden neurons. $\boldsymbol{z} = (\boldsymbol{z_1}, \ldots \boldsymbol{z_k}, \ldots \boldsymbol{z_c})$ are the predicted outputs at c output neurons of the neural network. In the training set, each sample $\boldsymbol{x}$ is associated with a target vector $\boldsymbol{t}$. It is expected that output $\boldsymbol{y}$ predicted by the learned neural network is close to the target $\boldsymbol{t}$ as possible.

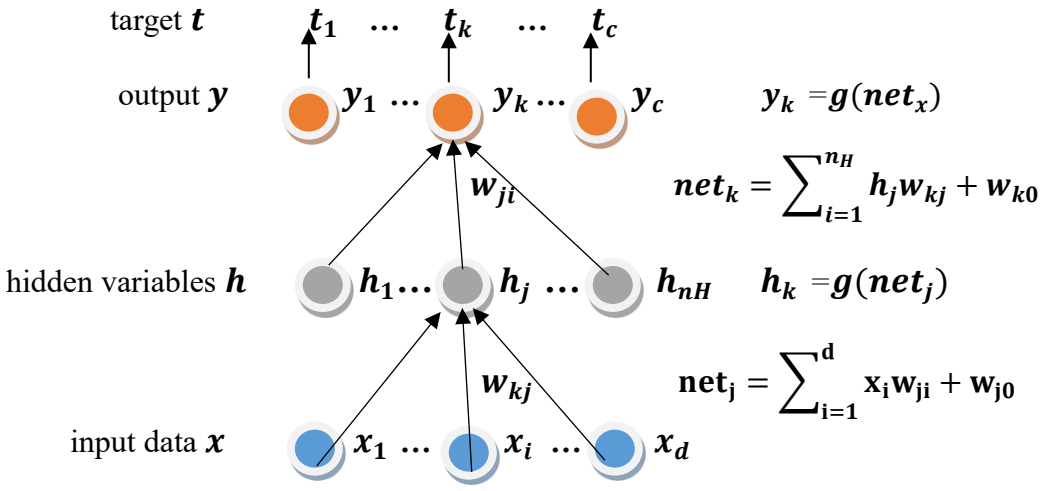

Figure 2.1: Architecture of a three-layer neural network

2.2.3 Difficulties of Employing Neural Networks

People encountered several major problems when employing neural networks in various applications in 1980s and 1990s. Neural networks typically have a large number of parameters and it was difficult to train them. It was easy for neural networks to overfit on training sets, while they performed poorly on test sets. It lacked large-scale training data, which made the over-fitting problem even more severe. Even a relatively large training set only had a few hundred training samples. Moreover, with very limited computational power available in 1980s and 1990s, it took a long time to train a small neural network. In general, the performance of neural networks was not significantly better than other machine learning tools and it was much more difficult to train neural networks. Therefore, many researchers gave up neural networks in early 2000s and turned to other machine learning tools such as SVM, Boosting, decision tree, and K-Nearest Neighbor.

Other machine learning models can be approximated with neural networks with only one or two hidden layers. Therefore, they are called models with shallow structures. An example of SVM is shown in Figure 2.2. The prediction function of SVM can be written as,

$$\boldsymbol{f(x) = b + \sum_{i=1}^{M} K(x_i, x)} \qquad \text{..... (1)}$$

$\boldsymbol{x}$ is a test sample. $\boldsymbol{x_i}$ is a support vector. There are totally $\boldsymbol{M}$ support vectors. $\boldsymbol{K}$ is the kernel function to measure the similarity between $\boldsymbol{x}$ and $\boldsymbol{x_i}$. As shown in Figure 1.4, SVM can be implemented with a three layer neural network with $\boldsymbol{M + 1}$ hidden neurons. $\boldsymbol{K(x_i, x)}$ is output at each hidden neuron $\boldsymbol{i}$. These models have loose ties with biological systems. Instead of solving general learning problems, people designed specific systems (models) for specific tasks and used different handcrafted features. For example, HMM-GMM was used in speech recognition, SIFT was used in object recognition, LBP was used in face recognition, and HOG was used in human detection.

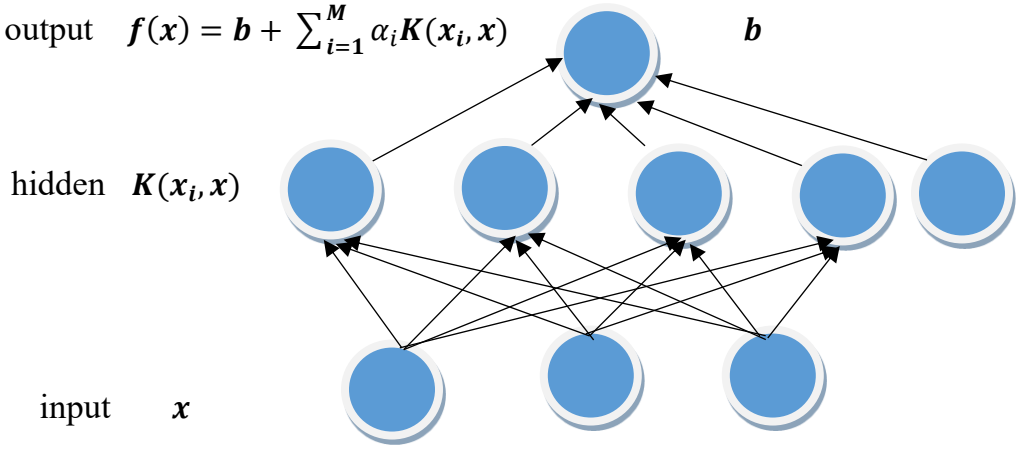

Figure 2.2: SVM can be approximated with a three layer neural network.

2.2.4 Deep Learning

Deep learning has become popular since 2006 [12]. A major break through in deep learning was first achieved in speech recognition [11]. It out-performed HMM-GMM, which dominated the field for many years, by a large margin. There are a few reasons making neural networks successful again. First of all, a key reason is the emergence of large-scale training data with annotations. For example, ImageNet [10] has millions of images with annotated class labels. With large-scale training data, deep neural networks show significant advantages compared with shallow models because of their very large learning capacity. With the fast development of high performance parallel computing systems, such as GPU clusters, it has become much easier to train large-scale deep neural networks with millions of parameters.

Moreover, there has been significant advances in the design of network structures, models, and training strategies. For example, unsupervised and layer wise pre-training has been proposed. It makes a neural network reach a good initialization point. Based on that, fine-tuning with BP can find a better local minimum. It helps to solve the under fitting problem in large-scale training sets to some extent. Dropout and data augmentation [14] have been proposed to solve the overfitting

problem in training. Batch normalization has been proposed to train very deep neural networks efficiently. Various network structures such as AlexNet [14], Overfeat [18], GoogLeNet [20], and VGG [6] have been extensively studied to optimize the performance of deep learning.

What prompts deep learning to have a huge impact on the entire academic community? It may owe to the contribution of Hinton's group, whose continuous efforts have demonstrated that deep learning would bring a revolutionary breakthrough on grand challenges rather than just obvious improvements on small datasets. Their success results from training a large CNN on 1.2 million labeled images together with a few techniques.

2.2.5 Deep learning achievements in computer vision

a) Object recognition and detection

Deep learning started to have a huge impact on computer vision in 2012, when Hinton's group won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) with deep learning [17]. Before that, there were attempts to apply deep learning to relatively small datasets and the obtained improvement was marginal compared with other computer vision methods. The computer vision community was not fully convinced that deep learning would bring revolutionary breakthrough without strong evidence on grand challenges until 2012.

ILSVRC is one of the most important grand challenges in computer vision, and has drawn the a lot of attention recently especially after the great success of deep learning in 2012. It was originally proposed in 2009 [10]. The challenge was to classify images collected from the web into 1,000 categories. Its training data includes more than one million images, much large than other datasets previously used to evaluate deep learning, such as MNIST. This competition has been running for several years and many top computer vision groups participated in the competition. However, different computer vision systems for object recognition tended to converge and there was no real break through until 2012. This section reviews the ILSVRC results from 2012 to 2014, so that readers can understand how fast deep learning has been developing in computer vision.

The differences between their classification accuracies were less than 1%. Since each image from ImageNet may contain multiple objects, top-5 error rate was commonly used for evaluation. The classification of an image is considered as correct if its labeled ground truth is among the top five classes predicted by the model. However, Hinton's group outperformed them by more than 10%,

reaching the top-5 error rate of 15.3%. They employed the convolutional neural network (CNN) [15] implemented with two GPUs.

The computer vision community was shocked by this result. Many people believed that a revolutionary break through was brought by deep learning to this field. Shortly thereafter, people found that the visual feature representation learned from ImageNet could be well generalized to other datasets and computer vision tasks, such as object detection [3], image segmentation [16], image retrieval and object tracking. For example, another well-known object recognition and detection challenge is PASCAL VOC.

However, its training set is too small to train deep models. Girshicket al. [3] applied the features learned from ImageNet with the image classification task and deep CNN to object detection on PSACALVOC. The detection rate was improved by 20%. This conclusion has significant impact. It indicates that once better features are learned by deep learning on ImageNet, many other computer vision problems can be improved accordingly. Therefore, deep learning on ImageNet has become the engine driving the computer vision field. That is one of the reasons that it has drawn most attention recently.

In ILSVRC 2013, the teams ranking top 20 all used deep learning. As shown in Table 1.2, the winner deep model was called Clarifai from NYU. The error rate was reduced to 11.19%. In that year, an object detection challenge was added. It required detecting objects of 200 categories from 40,000 test images. It is much more challenging than image classification, since each image may contain multiple objects of different categories. The highest mean Average Precision (mAP) was only 22.58%. The top two winners still used handcrafted features instead of deep learning.

	RCNN	*Berkley Vision*	*DeepInsight*	*GoogLeNet (Google)*	*DeepID-Net (CUHK)*
Avg	n/a	n/a	40.5	43.9	5o.3
Single	31.4	34.5	40.2	38.0	47.9

Table: Summary of mAP on ImageNet with different deep learning based object detection methods. “Single” represents the results achieved with single models. “Avg” represents the results achieved with model averaging. It has been well known that model averaging generally leads to improvement on image classification and object detection.

GoogLeNet [20] had more than 20 layers, and won both the image classification and object detection challenges. VGG [19] from Oxford won the localization challenge also with a very deep network. The image classification top-5 error rate was reduced to 6.66% and the mAP for object detection was largely improved to 43.93%. Summaries the progress of deep learning based object detection on ImageNet. RCNN [3] was the first widely used deep learning pipeline for general object detection and was proposed in 2013. The most recent work DeepID-Net has significantly advanced the state-of-the-art to mAP of 50.3.

b) Face recognition

Another major challenge in computer vision is face recognition. Labeled Faces in the Wild (LFW) [13] is the most well-known benchmark in face recognition. Most of the groups or companies working on face recognition reported their results on LFW. Many face recognition datasets were collected in lab environments under controlled condition.

In 2007, Huang et al. created the LFW dataset, which included face images of celebrities from the web, in order to evaluate face recognition performance in unconstrained conditions. Its test set includes 6,000 pairs of images and computation algorithms need to tell whether an image pair comes from the same person or not. The chance of random guess is 50%. According to the study, when only the central face regions (excluding hair) were cropped and shown to humans, the face verification accuracy by human eyes was 97.53%. When the whole images including hairs were shown to humans, the face verification accuracy by human eyes was 99.20%. A classical face recognition method, i.e. Eigen-face, only has 60% accuracy on LFW. It shows that the dataset is quite challenging. The best performing non-deep-learning technology obtained 96.33% face verification on LFW. With deep learning, it was the first time for DeepID2 to achieve face verification accuracy of 99.15% on LFW, comparable with human performance on this benchmark.Nowthenewstate-of-the-artDeepID2+ and FaceNet have achieved face verification accuracy of 99.45% and 99.63% on LFW respectively, surpassing human performance.

2.3 ZFNet

ZFNet, which was designed by Zeiler and Fergus, won the ILSVRC competition in 2013 by achieving a 11.2% top-5 error rate. This architecture was more of a fine-tuning to the previous AlexNet structure, but still developed some very keys ideas about improving performance. Another reason this was such a great paper is that the authors spent a good amount of time explaining a lot of the intuition behind ConvNets and showing how to visualize the filters and weights correctly. The architecture was a slight modification to AlexNet, with the following differences in Figure 2.3.

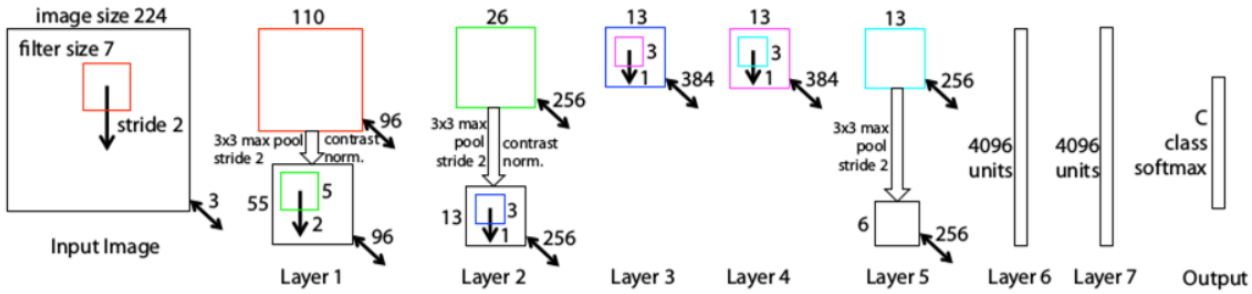

Figure 2.3: The architecture of ZFNet

Main Points

- Very similar architecture to AlexNet, except for a few minor modifications.
- AlexNet trained on 15 million images, while ZF Net trained on only 1.3 million images.
- Instead of using 11×11 Filters in the first Convolutional layer, ZFNet used a 7×7 Filter with $S = 2$. A smaller filter size helps to pick image features at a finer level of resolution.
- ZFNet increased the number of Activation Maps in the 3rd, 4th and 5th Convolutional layers from $(385,384,256)$ to $(512, 1024, 512)$, which increased the number of features that the network is capable of detecting.
- As the network grows, we also see a rise in the number of filters used.
- Used ReLUs for their activation functions, cross-entropy loss for the error function, and trained using batch stochastic gradient descent.
- Trained on a GTX 580 GPU for **twelve days.**

2.4 Architecture and Advantages of CNN

CNN is the most representative model of deep learning [36]. A typical CNN architecture, which is referred to as VGG16, can be found. Each layer of CNN is known as a feature map. The feature map of the input layer is a 3D matrix of pixel intensities for different color channels (e.g. RGB). The feature map of any internal layer is an induced multi-channel image, whose 'pixel' can be viewed as a specific feature. Every neuron is connected with a small portion of adjacent neurons from the previous layer (receptive field). Different types of transformations [14] can be conducted on feature maps, such as filtering and pooling. Filtering (convolution) operation convolutes a filter matrix (learned weights) with the values of a receptive field of neurons and takes a nonlinear function (such as sigmoid, ReLU) to obtain final responses. Pooling operation, such as max pooling,

average pooling, L2-pooling and local contrast normalization [37], summaries the responses of a receptive field into one value to produce more robust feature descriptions.

The typical VGG16 has totally 13 convolutional (conv) layers, 3 fully connected layers, 3 max-pooling layers and a softmax classification layer. The conv feature maps are produced by convoluting 3×3 filter windows, and feature map resolutions are reduced with 2 stride max-pooling layers. An arbitrary test image of the same size as training samples can be processed with the trained network.

Re-scaling or cropping operations may be needed if different sizes are provided [14].The advantages of CNN against traditional methods can be summarized as follows.

- Hierarchical feature representation, which is the multilevel representations from pixel to high-level semantic features learned by a hierarchical multi-stage structure [3], [38], can be learned from data automatically and hidden factors of input data can be disentangled through multi-level nonlinear mappings.

- Compared with traditional shallow models, a deeper architecture provides an exponentially increased expressive capability.

- The architecture of CNN provides an opportunity to jointly optimize several related tasks together (e.g., Fast RCNN combines classification and bounding box regression into a multi-task leaning manner).

- Benefitting from the large learning capacity of deep CNNs, some classical computer vision challenges can be recast as high-dimensional data transform problems and solved from a different viewpoint.

Due to these advantages, CNN has been widely applied into many research fields, such as image super-resolution reconstruction [39], image classification [35], image retrieval [40], face recognition [45], pedestrian detection [41], [42] and video analysis [43], [44].

2.5 CNN for Object Detection

With the advancement of accuracy in image classification, the research for object detection also developed in a fast speed. Before 2013, feature extraction techniques which proposed an combined application of HOG and SVM can achieve a high accuracy on the PASCAL data-set. In 2013, a fundamental revolution occurred in this field, which was caused by the introduction of Region based Convolutional Neural Networks (R-CNN), proposed by Girshick and Ross. R-CNN firstly proposes possible regions for residing objects, then makes use of CNN to classify objects in these regions. However, these two independent operations require high computation and make it time-consuming. An modification of R-CNN is made by Girshick and Ross, which is called fast R-CNN [2]. This architecture integrate the two independent tasks into one multi-task loss function, which accelerates the computation of proposals and classification. Later, a more integrated version of R-CNN, namely the faster R-CNN [22] was proposed by Ren et al., which achieves more than 10x faster than the original R-CNN. A recent proposal, R-FCN [46] with a fully convolutional layer as the final-parameterized layer further shortens the computation time used for region proposals.

R-CNN can be regarded as a cornerstone for the development of CNN for object detection. A large amount of work is based on this architecture and achieves great accuracy. However, a recent work shows that CNN based object detection can be even faster. YOLO (You Only Look Once) is such an architecture integrating region proposition and object classification into one single stage, which significantly contributes to simplification of the pipeline of object detection, as well as reduction of the total computation time.

Chapter 3 Architecture

3.1 Proposed Method

Object detection algorithm usually contains three parts. the first is the design of features, the second is the choice of detection window, and the third is the design of classifier. Feature design methods include artificial feature design and neural network feature extraction. The selection of detection window mainly includes: Exhaustive Search [4], Selective Search [5], and RPN method based on deep learning. This article adopts deep convolutional neural network (CNN) image feature extraction, using the most advanced RPN as the detection window selection method, the bounding box regression analysis, using softmax classification processing, and output the detection result. The model structure is shown with the help of blocks in Figure 3.1.

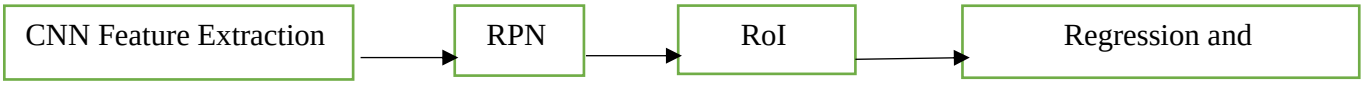

Figure 3.1: Object Detection Model

3.1.1 CNN Feature Extraction

At present, the image feature extraction using CNN mainly includes three steps: convolution, activation and pooling. Convolution is the core of feature extraction, and the feature of the image is obtained by different convolution kernels. Convolution kernel is equivalent to a filter, and different filters extract different features. Convolution is used to cross the convolution of the image with the size of 3*3 and the number of 2n. The activation function generally uses ReLU to perform non-linear operation on the output of the stacked layer, and has the function of accelerating convergence. Formula is as follows:

$$y = f(\sum_{j=0}^{j-1} \sum_{i=0}^{i-1} x_{ij} w_{ij} + b) \qquad \text{..... (2)}$$

Where x represents the input vector, w represents the parameters of a convolution kernel, b represents the bias term, f represents the activation function, and y represents the output.

A pooling layer is placed behind each roll layer to reduce dimension. Generally, the output matrix size of the original convolution layer is changed to half of the original one, which is convenient for the operation at the back. In addition, the pooling layer increases the robustness of the system, turns the original accurate description into a rough description, and avoids overfitting to some extent.

3.1.2 Region Proposal Networks

Region Proposal Networks (RPN) take the feature map extracted from the upper CNN as the input of this layer, maps the midpoint of the feature map back to the original image, and designs these different fixed scale windows in the original design. According to the window and the ground truth Intersection-over-Union (IoU) value to its positive and negative labels, let it learn whether there are objects inside, so training a Region Proposal Network.

Only the approximate place need to be found, because that the precise positioning position and size can be accomplished by following works. As the consequence the anchors can be fixed in three aspects: fixed scale changes (three scales), fixed length and width ratio changes (three ratio), fixed sampling method, only in the eigenvalues of each point in the original map of the corresponding Region of Interest (RoI) on the sampling, in the back of the work can be adjusted. This can reduce the complexity of the task.

After extracting the proposal on the feature map, the convolution calculation can be shared in front of the network. The result of this network is that each point of the convolution layer has an output about the k anchor boxes, including whether it is an object or not, adjusting the corresponding position of the box.

3.1.3 Translation-Invariant Anchors

At each sliding-window location, we simultaneously predict k region proposals, so the reg layer has $4k$ outputs encoding the coordinates of k boxes. The cls layer outputs $2k$ scores that estimate probability of object / not-object for each proposal. The k proposals are parameterized relative to k reference boxes, called anchors. Each anchor is centered at the sliding window in question, and is associated with a scale and aspect ratio. We use 3 scales and 3 aspect ratios, yielding $k = 9$ anchors at each sliding position. For a conv feature map of a size $W \times H$ (typically~2,400), there are WHk anchors in total. An important property of our approach is that it is translation invariant, both in terms of the anchors and the functions that compute proposals relative to the anchors.

As a comparison, the MultiBox method [48] uses k-means to generate 800 anchors, which are not translation invariant. If one translates an object in an image, the proposal should translate and the

same function should be able to predict the proposal in either location. Moreover, because the MultiBox anchors are not translation invariant, it requires a $(4+1)\times 800$-dimensional output layer, whereas our method requires a$(4+2)\times 9$-dimensional output layer. Our proposal layers have an order of magnitude fewer parameters (27 million for MultiBox using GoogLeNet [20] vs. 2.4 million for RPN using VGG-16), and thus have less risk of overfitting on small datasets, like PASCAL VOC.

3.1.4 A Loss Function for Learning Region Proposals

For training RPNs, we assign a binary class label (of being an object or not) to each anchor. We assign a positive label to two kinds of anchors:

(i) the anchor/anchors with the highest Intersection over-Union (IoU) overlap with a ground-truth box, or

(ii) an anchor that has an IoU overlap higher than 0.7 with any ground-truth box.

Note that a single ground-truth box may assign positive labels to multiple anchors. We assign a negative label to a non-positive anchor if its IoU ratio is lower than 0.3 for all ground-truth boxes. Anchors that are neither positive nor negative do not contribute to the training objective.
With these definitions, we minimize an objective function following the multi-task loss in Fast R-CNN. Our loss function for an image is defined as:

$$L(\{\mathrm{p_i}\}\{\mathrm{t_i}\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*) \quad \text{..... (3)}$$

Here, i is the index of an anchor in a mini-batch and p_i is the predicted probability of anchor i being an object. The ground-truth label p_i^* is 1 if the anchor is positive, and is 0 if the anchor is negative. t_i is a vector representing the 4 parameterized coordinates of the predicted bounding-box, and t_i^* is that of the ground-truth box associated with a positive anchor. The classification loss L_{cls} is log loss over two classes (object vs. not object). For the regression loss, we use $L_{reg}(t_i, t_i^*) = R(t_i - t_i^*)$ where R is the robust loss function (smoothL_1) defined in [5]. The term $p_i^* L_{reg}$ means the regression loss is activated only for positive anchors $(p_i^* = 1)$ and is disabled otherwise $(p_i^* = 0)$. The outputs of the cls and reg layers consist of $\{p_i\}$ and $\{t_i\}$ respectively. The two terms are normalized with N_{cls} and N_{reg}, and a balancing weight $\boldsymbol{\lambda}$.

For regression, we adopt the parameterizations of the 4 coordinates following [6]

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

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

where x, y, w and h denote the two coordinates of the box center, width, and height. Variables x, x_a and x^* are for the predicted box, anchor box, and ground-truth box respectively (likewise for y, w, h). This can be thought of as bounding-box regression from an anchor box to a nearby ground-truth box.

Nevertheless, our method achieves bounding-box regression by a different manner from previous feature-map-based methods. Bounding-box regression is performed on features pooled from arbitrarily sized regions, and the regression weights are shared by all region sizes. In our formulation, the features used for regression are of the same spatial size ($n \times n$) on the feature maps. To account for varying sizes, a set of k bounding-box regressors are learned. Each regressor is responsible for one scale and one aspect ratio, and the k regressors do not share weights. As such, it is still possible to predict boxes of various sizes even though the features are of a fixed size/scale.

3.1.5 RoI Pooling

For traditional CNN, when the network is trained, the input image size must be a fixed value, and the network output is also a fixed size vector or matrix. If the input image size is uncertain, the problem becomes more cumbersome. There are two kinds of solutions:

I. Cut from the image part of the incoming network;
II. Zoom the image into the desired size and then into the network.

However, the clipping will destroy the complete structure of the image, and the zoom will destroy the original shape information of the image. Therefore, Faster R-CNN proposed RoI Pooling to solve this problem.

First, the region proposal is mapped back to the original feature graph scale, and then each proposal level and vertical are divided into k copies, each of which is max pooling processing. After this processing, even if the size of the proposal, the output is $k \times k$ size, to achieve a fixed-length output.

3.2 Generic Object Detection

Generic object detection aims at locating and classifying existing objects in any one image, and labeling them with rectangular bounding boxes to show the confidences of existence. The frameworks of generic object detection methods can mainly be categorized into two types (see Figure 3.2). One follows traditional object detection pipeline, generating region proposals at first and then classifying each proposal into different object categories. The other regards object detection as a regression or classification problem, adopting a unified framework to achieve final results (categories and locations) directly. The region proposal based methods mainly include R-CNN [3], SPP-net [4], Fast R-CNN [2], Faster R-CNN [22], R-FCN [46], FPN [51] and Mask R-CNN [52], some of which are correlated with each other (e.g. SPP-net modifies RCNN with a SPP layer). The regression/classification based methods mainly includes MultiBox [31], G-CNN [53]. The correlations between these two pipelines are bridged by the anchors introduced in Faster RCNN. Details of these methods are as follows.

3.2.1 Region Proposal Based Framework

The region proposal based framework, a two-step process, matches the attentional mechanism of human brain to some extent, which gives a coarse scan of the whole scenario firstly and then focuses on regions of interest. Among the pre-related works [18], [55], [56], the most representative one is Overfeat [18]. This model inserts CNN into sliding window method, which predicts bounding boxes directly from locations of the topmost feature map after obtaining the confidences of underlying object categories.

I. R-CNN:

It is of significance to improve the quality of candidate bounding boxes and to take a deep architecture to extract high-level features. To solve these problems, R-CNN [3] was proposed by Ross Girshick in 2014 and obtained a mean average precision (mAP) of 53.3% with more than 30% improvement over the previous best result (DPM HSC [57]) on PASCAL VOC 2012. Figure 3 shows the flowchart of R-CNN, which can be divided into three stages as follows.

3.2.2 Region proposal generation

The R-CNN adopts selective search [7] to generate about 2k region proposals for each image. The selective search method relies on simple bottom-up grouping and saliency cues to provide more accurate candidate boxes of arbitrary sizes quickly and to reduce the searching space in object detection [28], [10].

CNN based deep feature extraction

In this stage, each region proposal is warped or cropped into a fixed resolution and the CNN module in [14] is utilized to extract a 4096 dimensional feature as the final representation. Due to large learning capacity, dominant expressive power and hierarchical structure of CNNs, a high-level, semantic and robust feature representation for each region proposal can be obtained.

3.2.3 Classification and localization

With pre-trained category specific linear SVMs for multiple classes, different region proposals are scored on a set of positive regions and background (negative) regions. The scored regions are then adjusted with bounding box regression and filtered with a greedy non-maximum suppression (NMS) to produce final bounding boxes for preserved object location.

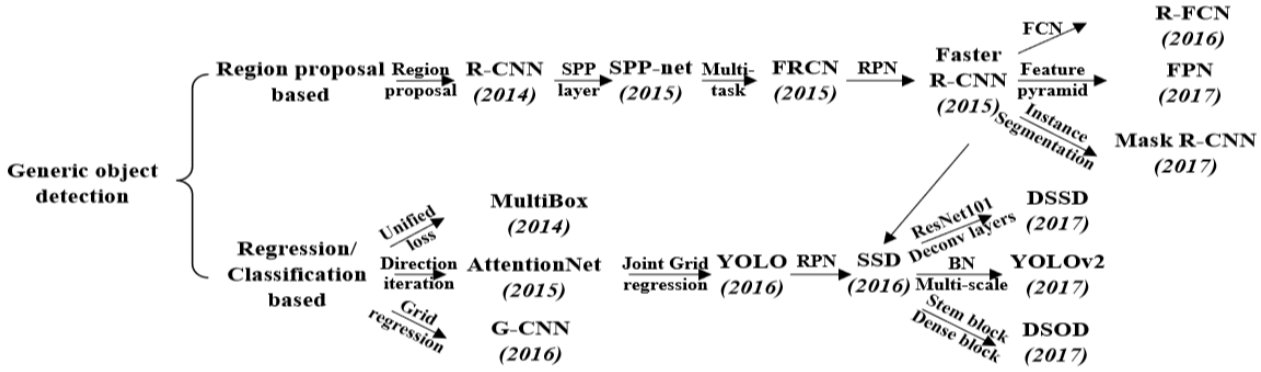

Figure 3.2: Two types of frameworks: region proposal based and regression/classification based. SPP: Spatial Pyramid Pooling, FRCN: Faster R-CNN, RPN: Region Proposal Network, FCN: Fully Convolutional Network, BN: Batch Normalization [19], Deconv layers: Deconvolution layers [55].

When there are scarce or insufficient labeled data, pre-training is usually conducted. Instead of unsupervised pre-training [58], R-CNN firstly conducts supervised pre-training on ILSVRC, a very large auxiliary dataset, and then takes a domain-specific fine-tuning. This scheme has been adopted by most of subsequent approaches [22], [2].

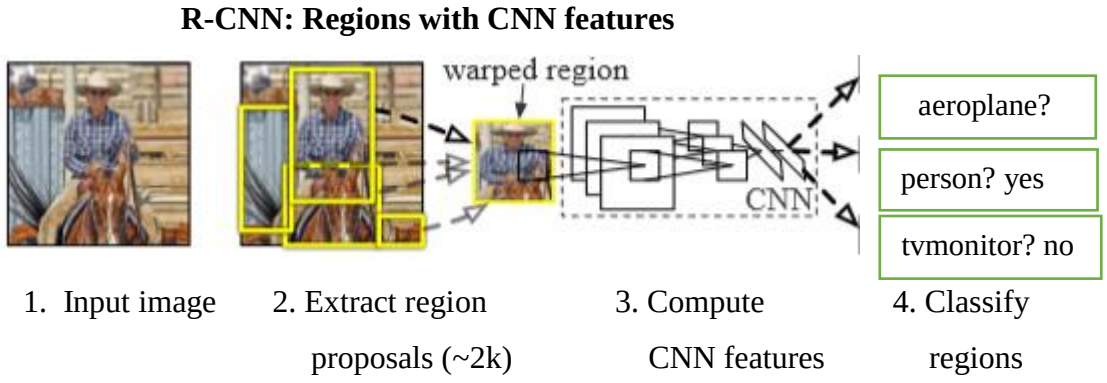

Figure 3.3: The flowchart of R-CNN, which consists of 3 stages: (1) extracts bottom-up region proposals, (2) computes features for each proposal using a CNN, and then classifies each region with class-specific linear SVMs

In spite of its improvements over traditional methods and significance in bringing CNN into practical object detection, there are still some disadvantages.

- Due to the existence of FC layers, the CNN requires a fixed-size (e.g., 227×227) input image, which directly leads to the re-computation of the whole CNN for each evaluated region, taking a great deal of time in the testing period.
- Training of R-CNN is a multi-stage pipelines. At first, a convolutional network (ConvNet) on object proposals is fine-tuned. Then the softmax classifier learned by fine-tuning is replaced by SVMs to fit in with ConvNet features. Finally, bounding-box regressors are trained.
- Training is expensive in space and time. Features are extracted from different region proposals and stored on the disk. It will be take a long time to process a relatively small training set with very deep networks, such as VGG16. At the same time, the storage memory required by these features should also be a matter of concern.
- Although selective search can generate region proposals are still redundant and this procedure is time-consuming (around 2 seconds to extract 2k region proposals).

To solve these problems, many methods have been proposed. GOP [59] takes a much faster geodesic based segmentation to replace traditional graph cuts. MCG [60] searches different scales of the image for multiple hierarchical segmentations and combinatorically groups different regions to produce proposals. Instead of extracting visually distinct segments, the edge boxes method [8] adopts the idea that objects are more likely to exist in bounding boxes with fewer contours straggling their boundaries. Also, some researches tried to re-rank or refine pre-extracted region proposals to remove unnecessary ones and obtained a limited number of valuable ones, such as DeepBox [61] and SharpMask [62].

In addition, there are some improvements to solve the problem of inaccurate localization. Zhang et al. [63] utilized a bayesian optimization based search algorithm to guide the regressions of different bounding boxes sequentially, and trained class-specific CNN classifiers with a structured loss to penalize the localization inaccuracy explicitly. Saurabh Gupta et al. improved object detection for RGB-D images with semantically rich image and depth features [64], and learned a new geocentric embedding for depth images to encode each pixel. The combination of object detectors and super pixel classification framework gains a promising result on semantic scene segmentation task. Ouyang et al. proposed a deformable deep CNN (DeepID-Net) [65], which introduces a novel deformation constrained pooling (def-pooling) layer to impose geometric penalty on the deformation of various object parts and makes an ensemble of models with different settings. Lenc et al. [66] provided an analysis on the role of proposal generation in CNN-based detectors and tried to replace this stage with a constant and trivial region generation scheme. The goal is achieved by biasing sampling to match the statistics of the ground truth bounding boxes with K-means clustering. However, more candidate boxes are required to achieve comparable results to those of R-CNN.

II. SPP-net

FC layers must take a fixed-size input. That’s why R-CNN chooses to warp or crop each region proposal into the same size. However, the object may exist partly in the cropped region and unwanted geometric distortion may be produced due to the warping operation. These content losses or distortions will reduce recognition accuracy, especially when the scales of objects vary. To solve this problem, He et al. took the theory of spatial pyramid matching (SPM) [67] into consideration and proposed a novel CNN architecture named SPP-net [64]. SPM takes several finer to coarser scales to partition the image into a number of divisions and aggregates quantized local features into mid-level representations.

The architecture of SPP-net for object detection can be found in Figure 3.4. Different from R-CNN, SPP-net reuses feature maps of the 5-th conv layer (conv5) to project region

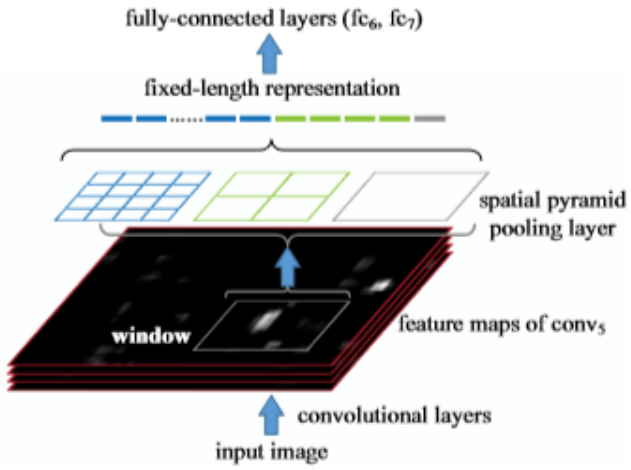

Figure 3.4: The architecture of SSP-net for object detection

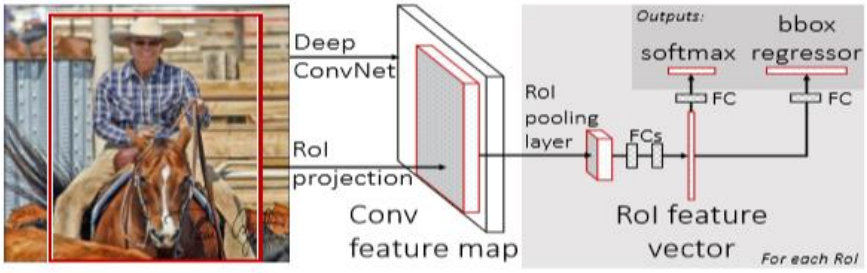

Figure 3.5: The Architecture of Fast R-CNN

Proposals of arbitrary sizes to fixed-length feature vectors. The feasibility of the reusability of these feature maps is due to the fact that the feature maps not only involve the strength of local response, but also have relationships with their spatial positions [4]. The layer after the final conv layer is referred to as spatial pyramid pooling layer (SPP layer). If the number of feature maps in conv5 is 256, taking a 3-level pyramid, the final feature vector for each region proposal obtained after SPP layer has a dimension of $256 \times (12 + 22 + 42) = 5376$.

SPP-net not only gains better results with correct estimation of different region proposals in their corresponding scales, but also improves detection efficiency in testing period with the sharing of computation cost before SPP layer among different proposals.

III. Fast R-CNN:

Although SPP-net has achieved impressive improvements in both accuracy and efficiency over R-CNN, it still has some notable drawbacks. SPP-net takes almost the same multi-stage pipeline as R-CNN, including feature extraction, network fine-tuning. SVM training and bounding-box regressor fitting. So an additional expense on storage space is still required. Additionally, the conv layers preceding the SPP layer cannot be updated with the fine-tuning algorithm introduced in [4]. As a result, an accuracy drop of very deep networks is unsurprising. To this end, Girshick [2] introduced a multi-task loss on classification and bounding box regression and proposed a novel CNN architecture named Fast R-CNN.

The architecture of Fast R-CNN is exhibited in Figure 3.5. Similar to SPP-net, the whole image is processed with conv layers to produce feature maps. Then, a fixed–length feature vector is extracted from each region proposal with a region of interest (RoI) pooling layer. The RoI pooling layer is a spatial case of the SPP layer, which has only one pyramid level. Each feature vector is then fed into a sequence of FC layers before finally branching into two sibling output layers. One output layer is responsible for producing softmax probabilities for all $C + 1$ categories (C object classes plus one 'background class') and the other output layer encodes refined bounding box positions with four real-valued numbers. All parameters in these procedures (except the generation of region proposal) are optimized via a multi-task loss in an end-t-end way.

The multi-tasks loss L is defined as below to jointly train classification and bounding-box regression,

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(t^u, v) \qquad \text{..... (4)}$$

Where $L_{cls}(p, u) = -logpu$ calculates the log loss for ground truth class u and p_u is driven from the discrete probability distribution $p = (p_0, \ldots\ldots, pC)$ over the $C + 1$ outputs from the last FC layer. $L_{loc}(t^u, v)$ is defined over the predicted offsets $t^u = (t_x^u, t_y^u, t_w^u, t_h^u)$ and ground-truth bounding-box regression targets $v = (v_x, v_y, v_w, v_h)$, where x, y, w, h denote the two coordinates of the box center, width, and height, respectively. Each t^u adopts the parameter settings in [3] to specify an object proposal with a log-space height/width shift and scale in variant translation. The Iverson bracket indicator function $[u \geq 1]$ is employed to omit all background RoIs. To provide more robustness against outliers and eliminate the sensitivity in exploding gradients, a smooth L_1 loss is adopted to fit bounding-box regressors as below,

$$L_{loc}(t^u, v) = \sum_{i \in x,y,w,h} smooth_{L_1}(t_i^u - v_i) \quad \ldots\ldots\ldots\ldots\ldots (5)$$

Where,

$$smooth_{L_1}(x) = f(x) = \begin{cases} 0.5x^2, & if\ |x| < 1 \\ |x| - 0.5, & otherwise \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots (6)$$

To accelerate the pipeline of Fast R-CNN, another two tricks are of necessity. On one hand, if training samples (i.e. RoIs) come from different images, back-propagation through the SPP layer becomes highly inefficient. Fast R-CNN samples mini-batches hierarchically, namely N images sampled randomly at first and then R/N RoIs sampled in each image, where R represents the number of RoIs. Critically, RoIs share computation and memory from the same image in the forward and backward pass. On the other hand, much time is spent in computing the FC layers during the forward pass [2]. The truncated Singular Value Decomposition (SVD) [68] can be utilized to compress large FC layers and to accelerate the testing procedure.

In the Fast R-CNN, regardless of region proposal generation, the training of all network layers can be processed in a single-stage with a multi-task loss. It saves the additional expense on storage space, and improves both accuracy and efficiency with more reasonable training schemes.

IV. Faster R-CNN

Despite the attempt to generate candidate boxes with biased sampling [66], state-of-the-art object detection networks mainly rely on additional methods, such as selective search and Edgebox, to generate a candidate pool of isolated region proposals. Region proposal computation is also a bottleneck in improving efficiency. To solve this problem, Ren et al. introduced an additional Region Proposal Network (RPN) [22], which acts in a nearly cost-free way by sharing full-image conv features with detection network.

RPN is achieved with a fully-convolutional network, which has the ability to predict object bounds and scores at each position simultaneously. Similar to [7], RPN takes an image of arbitrary size to generate a set of rectangular object proposals. RPN operates on a specific conv layer with the preceding layers shared with object detection network.

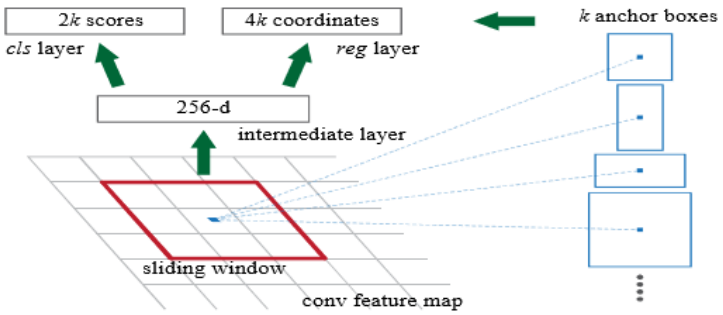

Figure 3.6: The RPN in Faster R-CNN [18], K predefined anchor boxes are convoluted with each sliding window to produce fixed-length vectors, which are taken, by cls and reg layer to obtain corresponding output.

The architecture of RPN is shown in figure 3.6. The network slides over the conv feature map and fully connects to an $n \times n$ spatial window. A low dimensional vector (512-d for VGG16) is obtained in each sliding window and fed into two sibling FC layers, namely box-classification layer (cls) and box-regression layer (reg). This architecture is implemented with an $n \times n$ conv layer followed by two sibling 1×1 conv layers. To increase non-linearity, ReLU is applied to the output of the $n \times n$ conv layer.

The regressions towards true bounding boxes are achieved by comparing proposals relative to reference boxes (anchors). In the Faster R-CNN, anchors of 3 scales and 3 aspect ratios are adopted. The loss function is similar to (4).

$$L(\{\mathrm{p_i}\}\{\mathrm{t_i}\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*) \quad \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

where, p_i shows the predicted probability of the i-th anchor being an object. The ground truth label p_i^* is 1 if the anchor is positive, otherwise 0. t_i stores 4 parameterized coordinates of the predicted bounding box while t_i^* is related to the ground-truth box overlapping with a positive anchor. L_{cls} is a binary $\log$ loss and L_{reg} is a smoothed L_1 loss similar to (5). These two terms are normalized with the mini-batch size (N_{cls}) and the number of anchor locations (N_{reg}), respectively. In the form of fully- convolutional networks, Faster R-CNN can be trained end-to-end by back-propagation and SGD in an alternate training manner.

With the proposal of Faster R-CNN, region proposal based CNN architectures for object detection can really be trained in an end-to-end way. Also a frame rate of 5 FPS (Frame Per Second) on a GPU is achieved with state-of-the-art object detection accuracy on PASCAL VOC 2007 and 2012. However, the alternate training algorithm is very time-consuming and RPN produces object-like

regions (including backgrounds) instead of object instances and is not skilled in dealing with objects with extreme scales or shapes.

V. R-FCN

Divided by the RoI pooling layer, a prevalent family [2], [22] of deep networks for object detection are composed of two subnetworks: a shared fully convolutional subnetwork (independent of RoIs) and an unshared RoI-wise subnetwork. This decomposition originates from pioneering classification architectures (e.g. AlexNet [14] and VGG16 [6]) which consist of a convolutional subnetwork and several FC layers separated by a specific spatial pooling layer.

Recent state-of-the-art image classification networks, such as Residual Nets (ResNets) [4] and GoogLeNets [20], are fully convolutional. To adapt to these architectures, it's natural to construct a fully convolutional object detection network without RoI-wise subnetwork. However, it turns out to be inferior with such a native solution [4]. This inconsistence is due to the dilemma of respecting translation invariance in image classification. In other words, shifting an object inside an image should be indiscriminative in image classification while any translation of an object in a bounding box may be meaningful in object detection. A manual insertion of the RoI pooling layer into convolutions can break down translation invariance at the expense of additional unshared region-wise layers.

Different from Faster R-CNN, for each category, the last conv layer of R-FCN produces a total of k^2 position-sensitive score maps with a fixed grid of $k \times k$ firstly and a position-sensitive RoI pooling layer is then appended to aggregate the responses from these score maps. Finally, in each RoI, k^2 position-sensitive scores are averaged to produce a $C + 1$-d vector and softmax responses across categories are computed. Another $4k^2$-d conv layer is appended to obtain class-agnostic bounding boxes.

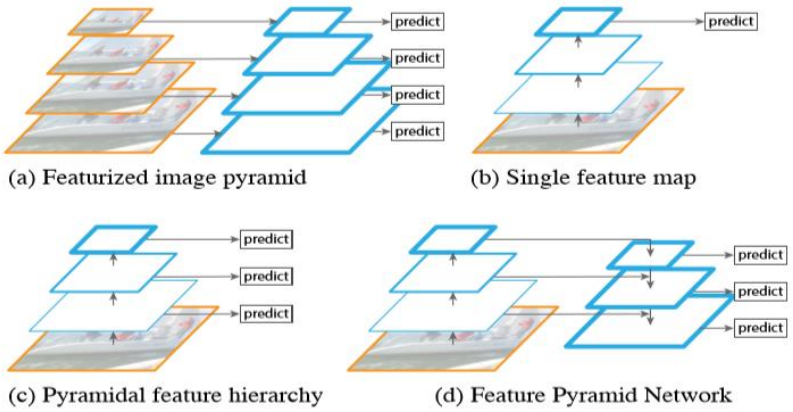

Figure 3.7: The main concern of FPN. (a) It is slow to use an image pyramid to build a feature pyramid. (b) Only single scale features is adopted for faster detection. (c) An alternative to the featured image pyramid is to reuse the pyramidal feature hierarchy computed by a ConvNet. (d) FPN integrates both (b) and (c). Blue outlines indicate feature maps and thicker outlines denote semantically stronger features.

With R-FCN, more powerful classification networks can be adopted to accomplish object detection in a fully convolutional architecture by sharing nearly all the layers, and state-of-the art results are obtained on both PASCAL VOC and Microsoft COCO [9] datasets at a test speed of 170ms per image.

VI. FPN

Feature pyramids built upon image pyramids (featured image pyramids) have been widely applied in many object detection systems to improve scale invariance [4] (Figure 3.7(a)). However, training time and memory consumption increase rapidly. To this end, some techniques take only a single input scale to represent high-level semantics and increase the robustness to scale changes (Figure 3.7(b)), and image pyramids are built at test time which results in an inconsistency between train/test-time inferences [2], [22]. The in-network feature hierarchy in a deep ConvNet produces feature maps of different spatial resolutions while introduces large semantic gaps caused by different depths (Figure 3.7(c)). To avoid using low-level features, pioneer works [54], usually build the pyramid starting from middle layers or just sum transformed feature responses, missing the higher-resolution maps of the feature hierarchy.

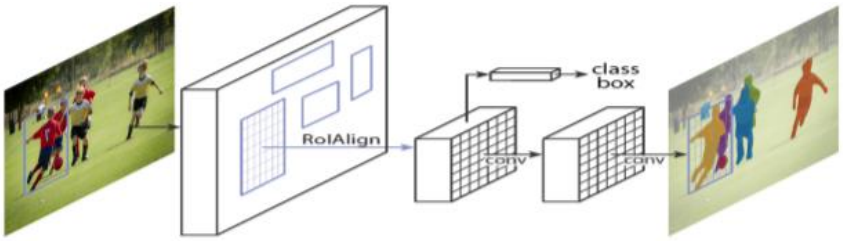

Figure 3.8: The Mask R-CNN framework for instance segmentation [67]

Different from these approaches, FPN [51] holds an architecture with a bottom-up pathway, a top-down pathway and several lateral connections to combine low-resolution and semantically strong features with high-resolution and semantically weak features (Figure 3.7(d)). The bottom-up pathway, which is the basic forward backbone ConvNet, produces a feature hierarchy by down sampling the corresponding feature maps with a stride of 2. The layers owning the same size of output maps are grouped into the same network stage and the output of the last layer of each stage is chosen as the reference set of feature maps to build the following top-down pathway.

To build the top-down pathway, feature maps from higher network stages are up sampled at first and then enhanced with those of the same spatial size from the bottom-up pathway via lateral connections. A 1×1 conv layer is appended to the up sampled map to reduce channel dimensions and the mergence is achieved by element-wise addition. Finally, a 3×3 convolution is also appended to each merged map to reduce the aliasing effect of up sampling and the final feature map is generated. This process is iterated until the finest resolution map is generated.
As feature pyramid can extract rich semantics from all levels and be trained end-to-end with all scales, state-of-the art representation can be obtained without sacrificing speed and memory. Meanwhile, FPN is independent of the backbone CNN architectures and can be applied to different stages of object detection (e.g. region proposal generation) and to many other computer vision tasks (e.g. instance segmentation).

VII. Mask R-CNN

Instance segmentation is a challenging task, which requires detecting all objects in an image and segmenting each instance (semantic segmentation [70]). These two tasks are usually regarded as two independent processes. In addition, the multi-task scheme will create spurious edge and exhibit systematic errors on overlapping instances. To solve this problem, parallel to the existing branches in Faster R-CNN for classification and bounding box regression, the Mask R-CNN [52] adds a branch to predict segmentation masks in a pixel-to-pixel manner (Figure 3.8).

Different from the other two branches, which are inevitably collapsed into short output vectors by FC layers, the segmentation mask branch encodes an $m \times m$ mask to maintain the explicit object spatial layout. This kind of fully convolutional representation requires fewer parameters but is more accurate than that of [70]. Formally, besides the two losses in (4) for classification and bounding box regression, an additional loss for segmentation mask branch is defined to reach a multi-task loss. Moreover, this loss is only associated with ground-truth class and relies on the classification branch to predict the category.

Because RoI pooling, the core operation in Faster R-CNN, performs a coarse spatial quantization for feature extraction, misalignment is introduced between the RoI and the features. It affects classification little because of its robustness to small translations. However, it has a large negative effect on pixel-to-pixel mask prediction. To solve this problem, Mask R-CNN adopts a simple and quantization-free layer, namely RoIAlign, to preserve the explicit per-pixel spatial correspondence faithfully. RoIAlign is achieved by replacing the harsh quantization of RoI pooling with bilinear interpolation [99], computing the exact values of the input features at four regularly sampled locations in each RoI bin. In spite of its simplicity, this seemingly minor change improves mask accuracy greatly, especially under strict localization metrics.

Given the Faster R-CNN framework, the mask branch only adds a small computational burden and its cooperation with other tasks provides complementary information for object detection. As a result, Mask R-CNN is simple to implement with promising instance segmentation and object detection results. In a word, Mask R-CNN is a flexible and efficient framework for instance-level recognition, which can be easily generalized to other tasks (e.g. human pose estimation [49]) with minimal modification.

VIII. Multi-task Learning, Multi-scale Representation and Contextual Modelling

Although the Faster R-CNN gets promising results with several hundred proposals, it still struggles in small-size object detection and localization, mainly due to the coarseness of its feature maps and limited information provided in particular candidate boxes. The phenomenon is more obvious on the Microsoft COCO dataset, which consists of objects at a broad range of scales, less prototypical images, and requires localization that is more precise. To tackle these problems, it is of necessity to accomplish object detection with multi-task learning [71], multi-scale representation [69] and context modelling [72] to combine complementary information from multiple sources.

a) Multi-task Learning

Multi-task Learning learns a useful representation for multiple correlated tasks from the same input [73]. Brahmbhatt et al. introduced conv features trained for object segmentation and 'stuff' (amorphous categories such as ground and water) to guide accurate object detection of small objects (StuffNet) [71]. Dai et al. [70] presented Multitask Network Cascades of three networks, namely class-agnostic region proposal generation, pixel-level instance segmentation and regional instance classification. Li et al. incorporated the weakly supervised object segmentation cues and region-based object detection into a multi-stage architecture to fully exploit the learned segmentation features [34].

b) Multi-scale Representation

Multi-scale Representation combines activations from multiple layers with skip-layer connections to provide semantic information of different spatial resolutions [51]. Cai et al. proposed the MS-CNN [74] to ease the inconsistency between the sizes of objects and receptive fields with multiple scale-independent output layers. Yang et al. investigated two strategies, namely scale-dependent pooling (SDP) and layerwise cascaded rejection classifiers (CRC), to exploit appropriate scale-dependent conv features [33]. Kong et al. proposed the HyperNet to calculate the shared features between RPN and object detection network by aggregating and compressing hierarchical feature maps from different resolutions into a uniform space [72].

c) Contextual Modelling

Contextual Modelling improves detection performance by exploiting features from or around RoIs of different support regions and resolutions to deal with occlusions and local similarities [69]. Zhu et al. proposed the SegDeepM to exploit object segmentation that reduces the dependency on initial candidate boxes with Markov Random Field [75]. Moysset et al. took advantage of 4 directional 2D-LSTMs to convey global context between different local regions and reduced trainable parameters with local parameter-sharing. Zeng et al. proposed a novel GBD-Net by introducing gated functions to control message transmission between different support regions.

d) The Combination

The Combination incorporates different components above into the same model to improve detection performance further. Gidaris et al. proposed the Multi-Region CNN (MR-CNN) model [76] to capture different aspects of an object, the distinct appearances of various object parts and semantic segmentation-aware features. To obtain contextual and multiscale representations, Bell et al. proposed the Inside-Outside Net (ION) by exploiting information both inside and outside the RoI [69] with spatial recurrent neural networks [77] and skip pooling [72]. Zagoruyko et al. proposed the MultiPath architecture by introducing three modifications to the Fast R-CNN [78], including multi-scale skip connections [69], a modified foveal structure and a novel loss function summing different IoU losses.

3.3 Optimization

The RPN, which is naturally implemented as a fully convolutional network [16], can be trained end-to-end by back-propagation and stochastic gradient descent (SGD) [47]. We follow the "image-centric" sampling strategy from [2] to train this network. Each mini-batch arises from a single image that contains many positive and negative anchors. It is possible to optimize for the loss functions of all anchors, but this will bias towards negative samples as they are dominate. Instead, we randomly sample 256 anchors in an image to compute the loss function of a mini-batch, where the sampled positive and negative anchors have a ratio of up to1: 1. If there are fewer than 128 positive samples in an image, we pad the mini-batch with negative ones.

We randomly initialize all new layers by drawing weights from a zero-mean Gaussian distribution with standard deviation 0.01. All other layers (i.e., the shared conv layers) are initialized by pretraining a model for ImageNet classification, as is standard practice [3]. We tune all layers of the ZF net, and conv3 1 and up for the VGG net to conserve memory [2]. We use a learning rate of 0.001 for 60k mini-batches, and 0.0001 for the next 20k mini-batches on the PASCAL dataset. We also use a momentum of 0.9 and a weight decay of 0.0005 [14].

3.4 Sharing Features for RPN and Fast R-CNN

Thus far, we have described how to train a network for region proposal generation, without considering the region-based object detection CNN that will utilize these proposals. For the detection network, we adopt Fast R-CNN [2]. Next, we describe algorithms that learn a unified network composed of RPN and Fast R-CNN with shared convolutional layers (Figure 3.9).

Both RPN and Fast R-CNN, trained independently, will modify their convolutional layers in different ways. We therefore need to develop a technique that allows for sharing convolutional layers between

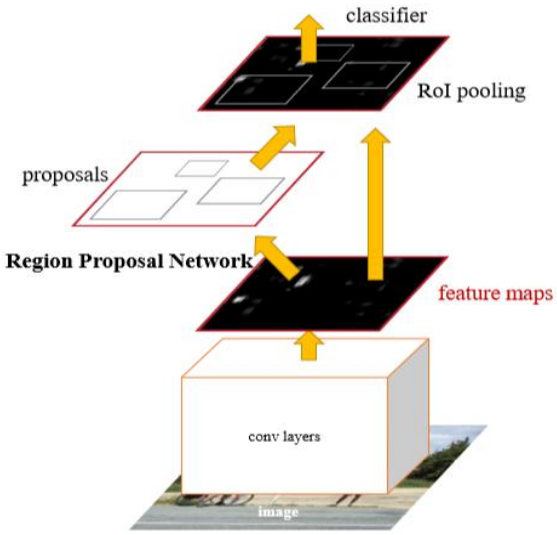

Figure 3.9: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

the two networks, rather than learning two separate networks. We discuss three ways for training networks with features shared:

i. **Alternating training:** In this solution, we first train RPN, and use the proposals to train Fast R-CNN. The network tuned by Fast R-CNN is then used to initialize RPN, and this process is iterated.

ii. **Approximate joint training:** In this solution, the RPN and Fast R-CNN networks are merged into one network during training as in Figure 3.9. In each SGD iteration, the forward pass generates region proposals that are treated just like fixed, pre-computed proposals when training a Fast R-CNN detector. The backward propagation takes place as usual, where for the shared layers the backward propagated signals from both the RPN loss and the Fast R-CNN loss are combined. This solution is easy to implement. But this solution ignores the derivative w.r.t. the proposal boxes' coordinates that are also network responses, so is approximate. In our experiments, we have empirically found this solver produces close results, yet reduces the training time by about 25-50% comparing with alternating training.

iii. **Non-approximate joint training:** As discussed above, the bounding boxes predicted by RPN are also functions of the input. The RoI pooling layer [2] in Fast R-CNN accepts the convolutional features and also the predicted bounding boxes as input, so a theoretically valid backpropagation solver should also involve gradients the box coordinates. These gradients are ignored in the above approximate joint training. In a non-approximate joint training solution, we need an RoI pooling layer that is differentiable w.r.t. the box coordinates. This is a nontrivial problem and a solution can be given by an "RoI warping" layer as developed in [15].

iv. **4-Step Alternating Training:** In this paper, we adopt a pragmatic 4-step training algorithm to learn shared features via alternating optimization. This network is initialized with an ImageNet-pre-trained model and fine-tuned end-to-end for the region proposal task. In the second step, we train a separate detection network by Fast R-CNN using the proposals generated by the step-1 RPN. This detection network is also initialized by the ImageNet-pre-trained model. At this point, the two networks do not share convolutional layers. In the third step, we use the detector network to initialize RPN training, but we fix the shared convolutional layers and only fine-tune the layers unique to RPN. Now the two networks share convolutional layers. Finally, keeping the shared convolutional layers fixed, we fine-tune the unique layers of Fast R-CNN. As such, both networks share the same convolutional layers and form a unified network. A similar alternating training can be run for more iterations, but we have observed negligible improvements.

3.5 Implementation Details

We train and test both region proposal and object detection networks on single-scale images [7, 5]. We re-scale the images such that their shorter side is $s = 600$ pixels [5]. Multi-scale feature extraction may improve accuracy but does not exhibit a good speed-accuracy trade-off [5]. We also note that for ZF and VGG nets, the total stride on the last conv layer is 16 pixels on the re-scaled image, and thus is $\sim$10 pixels on a typical PASCAL image($\sim 500 \times 375$). Even such a large stride provides good results, though accuracy may be further improved with a smaller stride. For anchors, we use 3 scales with box areas of 128^2, 256^2, and 512^2 pixels, and 3 aspect ratios of $1:1$, $1:2$, and $2:1$. We note that our algorithm allows the use of anchor boxes that are larger than the underlying receptive field when predicting large proposals. Such predictions are not impossible- one may still roughly infer the extent of an object if only the middle of the object is visible. With this design, our solution does not need multi-scale features or multi-scale sliding windows to predict large regions, saving considerable running time. Fig. 1 (right) shows the

capability of our method for a wide range of scales and aspect ratios. The table below shows the learned average proposal size for each anchor using the ZF net (numbers for $s = 600$).

anchor	128^2,2:1	128^2,1:1	128^2,1:2	256^2,2:1	256^2,1:1	256^2,1:2	512^2,2:1	512^2,1:1	512^2,1:2
proposal	188×111	113×114	70×92	416×229	261×284	174×332	768×437	499×501	355×715

The anchor boxes that cross image boundaries need to be handled with care. During training, we ignore all cross-boundary anchors so they do not contribute to the loss. For a typical 1000×600 image, there will be roughly 20k ($\approx 60 \times 40 \times 9$) anchors in total. With the cross-boundary anchors ignored, there are about 6k anchors per image for training. If the boundary-crossing outliers are not ignored in training, they introduce large, difficult to correct error terms in the objective, and training does not converge. During testing, however, we still apply the fully convolutional RPN to the entire image. This may generate cross-boundary proposal boxes, which we clip to the image boundary.

Some RPN proposals highly overlap with each other. To reduce redundancy, we adopt non-maximum suppression (NMS) on the proposal regions based on their cls scores. We fix the IoU threshold for NMS at 0.7, which leaves us about 2k proposal regions per image. As we will show, NMS does not harm the ultimate detection accuracy, but substantially reduces the number of proposals. After NMS, we use the top-N ranked proposal regions for detection. In the following, we train Fast R-CNN using 2k RPN proposals, but evaluate different numbers of proposals at test-time.

Chapter 4 Experimental Evaluation

4.1 Experiment on PASCAL VOC

For the ImageNet pre-trained network, we use the “fast” version of ZF net [50] that has 5 conv layers and 3 fc layers, and the public VGG-16 model5 [6] that has 13 conv layers and 3 fc layers. We primarily evaluate detection mean Average Precision (mAP), because this is the actual metric for object detection. These results use the ZF net. For Selective Search (SS) [7], we generate about 2k SS, proposals by the “fast” mode. For EdgeBoxes (EB) [8], we generate the proposals by the default EB setting tuned for 0.7 IoU. SS has an mAP of 58.7% and EB has an mAP of 58.6%. RPN with Fast R-CNN achieves competitive results, with an mAP of 59.9% while using up to 300 proposals6. Using RPN yields a much faster detection system than using either SS or EB because of shared conv computations; the fewer proposals also reduce the region-wise fc cost. Next, we consider several ablations of RPN and then show that proposal quality improves when using the very deep network.

4.1.1 Ablation Experiments

To investigate the behavior of RPNs as a proposal method, we conducted several ablation studies. First, we show the effect of sharing conv layers between the RPN and Fast R-CNN detection network. To do this, we stop after the second step in the 4-step training process. Using separate networks reduces the result slightly to 58.7% (RPN+ZF, unshared, Table 1). We observe that this is because in the third step when the detector-tuned features are used to fine-tune the RPN, the proposal quality is improved.

Next, we disentangle the RPN’s influence on training the Fast R-CNN detection network. For this purpose, we train a Fast R-CNN model by using the 2k SS proposals and ZF-net. We fix this detector and evaluate the detection mAP by changing the proposal regions used attest-time. In these ablation experiments, the RPN does not share features with the detector.
Replacing SS with 300 RPN proposals at test-time leads to an mAP of 56.8%. The loss in mAP is because of the inconsistency between the training/testing proposals. This result serves as the baseline for the following comparisons.

Somewhat surprisingly, the RPN still leads to a competitive result (55.1%) when using the top ranked 100 proposals at test-time, indicating that the top-ranked RPN proposals are accurate. On

the other extreme, using the top-ranked 6k RPN proposals (without NMS) has a comparable mAP (55.2%), suggesting NMS does not harm the detection mAP and may reduce false alarms.

Next, we separately investigate the roles of RPN's cls and reg outputs by turning off either of them attest-time. When the cls layer is removed at test-time (thus no NMS / ranking is used), we randomly sample N proposals from the unscored regions. The mAP is nearly unchanged with N = 1k (55.8%), but degrades considerably to 44.6% when N = 100. This shows that the cls scores account for the accuracy of the highest ranked proposals.

On the other hand, when the reg layer is removed at test-time (so the proposals become anchor boxes), the mAP drops to 52.1%. This suggests that the high-quality proposals are mainly due to regressed positions. The anchor boxes alone are not sufficient for accurate detection.

method	#proposals	data	mAP (%)	time (ms)
SS	2k	07	66.9	1830
SS	2k	07+12	70.0	1830
RPN+VGG, unshared	300	07	68.5	342
RPN+VGG, shared	300	07	69.9	198
RPN+VGG, shared	300	07+12	73.2	198

Table 1: Detection results on PASCALVOC 2007 testset. The detector is Fast R-CNN and VGG16. Training data: "07": VOC 2007 trainval, "07+12": union set of VOC 2007 trainval and VOC 2012 trainval. For RPN, the train-time proposals for Fast R-CNN are 2k. †: this was reported in [2]; using the repository provided by this paper, this number is higher (68.0 ± 0.3 in six runs).

method	#proposals	data	mAP (%)
SS	2k	07	66.9
SS	2k	07+12	70.0
RPN+VGG, unshared	300	07	68.5
RPN+VGG, shared	300	07	69.9
RPN+VGG, shared	300	07+12	73.2
RPN+VGG, shared	300	COCO+07+12	78.8

Table 2: Detection results on PASCAL VOC 2007 test set. The detector is Fast R-CNN and VGG-16. Training data: "07": VOC 2007 trainval, "07+12": union set of VOC 2007 trainval and VOC 2012 trainval. For RPN, the train-time proposals for Fast R-CNN are 2000 using the repository provided by this paper, this result is higher (68.1).

method	#proposals	data	mAP (%)
SS	2k	12	65.7
SS	2k	07++12	68.4
RPN+VGG, shared	300	12	67.0
RPN+VGG, shared	300	07++12	70.4

Table 3: Detection results on PASCAL VOC2012 testset. The detector is Fast R-CNN and VGG16. Training data: “07”: VOC 2007 trainval, “07++12”: union set of VOC 2007 trainval+test and VOC 2012 trainval. For RPN, the train-time proposals for Fast R-CNN are 2k.

model	system	conv	proposal	region-wise	total	rate
VGG	SS + Fast R-CNN	146	1510	174	1830	0.5 fps
VGG	RPN + Fast R-CNN	141	10	47	198	5 fps
ZF	RPN + Fast R-CNN	31	3	25	59	17 fps

Table 4: Timing (ms) on a K40 GPU, except SS proposal is evaluated in a CPU. “Region-wise” includes NMS, pooling, fc, and softmax. See our released code for the profiling of running time.

We also evaluate the effects of more powerful networks on the proposal quality of RPN alone. We use VGG-16 to train the RPN, and still use the above detector of SS+ZF. The mAP improves from 56.8% (using RPN+ZF) to 59.2% (using RPN+VGG). This is a promising result, because it suggests that the proposal quality of RPN+VGG is better than that of RPN+ZF. Because proposals of RPN+ZF are competitive with SS (both are 58.7% when consistently used for training and testing), we may expect RPN+VGG to be better than SS. The following experiments justify this hypothesis.

Performance of VGG-16

Table 3 shows the results of VGG-16 for both proposal and detection. Using RPN+VGG, the result is 68.5% for unshared features, slightly higher than the SS baseline. As shown above, this is because the proposals generated by RPN+VGG are more accurate than SS. Unlike SS that is predefined, the RPN is actively trained and benefits from better networks. For the feature-shared variant, the result is 69.9% better than the strong SS baseline, yet with nearly cost-free proposals. We further train the RPN and detection network on the union set of PASCAL VOC 2007 trainval and 2012 trainval. The mAP is 73.2%. Table 3 shows some results on the PASCAL VOC 2007 test set. On the PASCAL VOC 2012 test set, our method has an mAP of 70.4% trained on the union

set of VOC 2007 trainval+test and VOC 2012 trainval. Table 5 show the detailed numbers.

method	#box	data	mAP	bus	car	cat	dog	person	bike	bottle
SS	2000	12	65.7	73.9	68.6	87.7	86.0	69.8	74.7	37.7
SS	2000	07++12	68.4	77.8	71.6	89.3	87.5	72.0	78.4	38.7
RPN	300	12	67.0	72.1	72.3	87.3	86.8	77.4	76.4	45.2
RPN	300	07++12	70.4	77.5	75.9	88.5	86.9	79.6	79.8	49.8
RPN	300	COCO+07++12	75.9	81.9	82.0	91.3	89.0	84.1	83.6	59.6

Table 5: Results on PASCAL VOC 2012 test set with Fast R-CNN detectors and VGG-16. For RPN, the train-time proposals for Fast R-CNN are 2000.

4.1.2 Detection Accuracy and Running Time of VGG-16

Table 2 shows there sults of VGG-16 for both proposal and detection. Using RPN+VGG, the Fast R-CNN result is 68.5% for unshared features, slightly higher than the SS baseline. As shown above, this is because the proposals generated by RPN+VGG are more accurate than SS. Unlike SS that is pre-defined, the RPN is actively trained and benefits from better networks. For the feature-shared variant, the result is 69.9%—better than the strong SS baseline, yet with nearly cost-free proposals. We further train the RPN and detection network on the union set of PASCAL VOC 2007 trainval and 2012 trainval, following [2]. The mAP is 73.2%. On the PASCAL VOC 2012 test set (Table 3), our method has an mAP of 70.4% trained on the union set of VOC 2007 trainval+test and VOC 2012 trainval, following [2].

In Table 4, we summarize the running time of the entire object detection system. SS takes 1-2 seconds depending on content (on average 1.51s), and Fast R-CNN with VGG-16 takes 320ms on 2k SS proposals (or 223ms if using SVD on fc layers [2]). Our system with VGG-16 takes in total 198ms for both proposal and detection. With the conv features shared, the RPN alone only takes 10ms computing the additional layers. Our region-wise computation is also low, thanks to fewer proposals (300). Our system has a frame-rate of 17 fps with the ZF net.

4.1.3 Analysis of Recall-to-IoU

Next, we compute the recall of proposals at different IoU ratios with ground-truth boxes. It is noteworthy that the Recall-to-IoU metric is just loosely related to the ultimate detection accuracy. It is more appropriate to use this metric to diagnose the proposal method than to evaluate it.

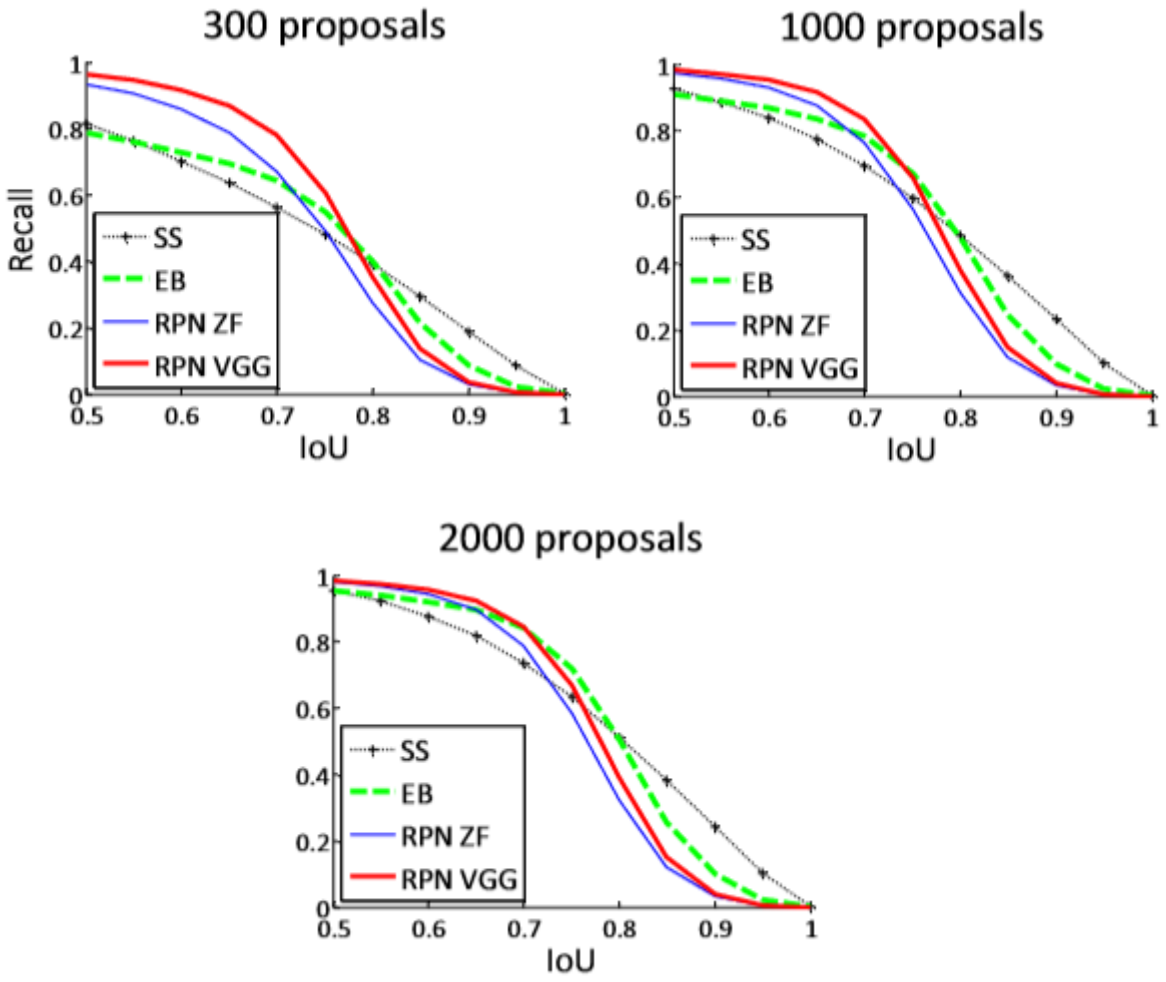

Figure 4.1: Recall vs. IoU overlap ratio on the PASCAL VOC 2007 test set.

	regions		detector	mAP (%)
Two-Stage	RPN + ZF, unshared	300	Fast R-CNN + ZF, 1 scale	58.7
One-Stage	dense, 3 scales, 3 asp. ratios	20k	Fast R-CNN + ZF, 1 scale	53.8
One-Stage	dense, 3 scale, 3 asp. ratios	20k	Fast R-CNN + ZF, 5 scales	53.9

Table 6: One-Stage Detection vs. Two-Stage Proposal + Detection. Detection results are on the PASCAL VOC 2007 test set using the ZF model and Fast R-CNN. RPN uses unshared features.

In Figure 4.1, we show the results of using 300, 1k, and 2k proposals. We compare with SS and EB, and the N proposals are the top-N ranked ones based on the confidence generated by these methods. The plots show that the RPN method behaves gracefully when the number of proposals

drops from 2kto300. This explains why the RPN has a good ultimate detection mAP when using as few as 300 proposals. As we analyzed before, this property is mainly attributed to the cls term of the RPN. The recall of SS and EB drops more quickly than RPN when the proposals are fewer.

4.1.4 One-Stage Detection vs. Two-Stage Proposal + Detection

The OverFeat paper [18] proposes a detection method that uses regressors and classifiers on sliding windows over conv feature maps. OverFeat is a one-stage, class-specific detection pipeline, and ours is a two-stage cascade consisting of class-agnostic proposals and class-specific detections. In OverFeat, the region-wise features come from a sliding window of one aspect ratio over a scale pyramid. These features are used to simultaneously determine the location and category of objects. In RPN, the features are from square (3×3) sliding windows and predict proposals relative to anchors with different scales and aspect ratios. Though both methods use sliding windows, the region proposal task is only the first stage of RPN + Fast R-CNN, the detector attends to the proposals to refine them. In the second stage of our cascade, the region-wise features are adaptively pooled [4], [2] from proposal boxes that more faithfully cover the features of the regions. We believe these features lead to more accurate detections.

To compare the one-stage and two-stage systems, we emulate the OverFeat system (and thus also circumvent other differences of implementation details) by one-stage Fast R-CNN. In this system, the "proposals" are dense sliding windows of 3 scales (128, 256, 512) and 3 aspect ratios (1:1, 1:2, 2:1). Fast R-CNN is trained to predict class-specific scores and regress box locations from these sliding windows. Because the OverFeat system uses an image pyramid, we also evaluate using conv features extracted from 5 scales. We use those 5 scales as in [4], [2].

Table 5 compares the two-stage system and two variants of the one-stage system. Using the ZF model, the one-stage system has an mAP of 53.9%. This is lower than the two-stage system (58.7%) by 4.8%. This experiment justifies the effectiveness of cascaded region proposals and object detection. Similar observations are reported in [2], [66], where replacing SS region proposals with sliding windows leads to ~6% degradation in both papers. We also note that the one-stage system is slower as it has considerably more proposals to process.

4.2 Experiments on MS COCO

We present more results on the Microsoft COCO object detection dataset [9]. This dataset involves 80 object categories. We experiment with the 80k images on the training set, 40k images on the validation set, and 20k images on the test-dev set. We evaluate the mAP averaged for IoU ∈ [0.5: 0.05: 0.95] (COCO's standard metric, simply denoted as mAP@[.5, .95]) and mAP@0.5

(PASCAL VOC's metric).

There are a few minor changes of our system made for this dataset. We train our models on an 8-GPU implementation, and the effective mini-batch size becomes 8 for RPN (1 per GPU) and 16 for Fast R-CNN (2 per GPU). The RPN step and Fast R-CNN step are both trained for 240k iterations with a learning rate of 0.003 and then for 80k iterations with 0.0003. We modify the learning rates (starting with 0.003 instead of 0.001) because the mini-batch size is changed. For the anchors, we use 3 aspect ratios and 4 scales (adding 642), mainly motivated by handling small objects on this dataset. In addition, in our Fast R-CNN step, the negative samples are defined as those with a maximum IoU with ground truth in the interval of [0, 0.5), instead of [0.1, 0.5) used in [4], [2]. We note that in the SPPnet system [4], the negative samples in [0.1, 0.5) are used for network fine-tuning, but the negative samples in [0, 0.5) are still visited in the SVM step with hard-negative mining. But the Fast R-CNN system [2] abandons the SVM step, so the negative samples in [0, 0.1) are never visited. Including these [0, 0.1) samples improves mAP@0.5 on the COCO dataset for both Fast R-CNN and Faster R-CNN systems (but the impact is negligible on PASCAL VOC).

			COCO val		COCO test-dev	
method	proposals	training data	mAP@.5	mAP@[.5,.95]	mAP@.5	mAP@[.5,.95]
Fast R-CNN	SS, 2000	COCO train	-	-	35.9	19.7
Fast R-CNN	SS, 2000	COCO train	38.6	18.9	39.3	19.3
Fast R-CNN	RPN, 300	COCO train	41.5	21.2	42.1	21.5
Fast R-CNN	RPN, 300	COCO trainval	-	-	42.7	21.9

Table 7: Object detection results (%) on the MS COCO dataset. The model is VGG-16.

The rest of the implementation details are the same as on PASCAL VOC. In particular, we keep using 300 proposals and single-scale (s = 600) testing. The testing time is still about 200ms per image on the COCO dataset.

In Table 7, we first report the results of the Fast R-CNN system [2] using the implementation in this paper. Our Fast R-CNN baseline has 39.3% mAP@0.5 on the test-dev set, higher than that reported in [2]. We conjecture that the reason for this gap is mainly due to the definition of the negative samples and also the changes of the mini-batch sizes. We also note that the mAP@[.5, .95] is just comparable.

Next, we evaluate our Faster R-CNN system. Using the COCO training set to train, Faster R-CNN has 42.1% mAP@0.5 and 21.5% mAP@[.5, .95] on the COCO test-dev set. This is 2.8% higher for mAP@0.5 and 2.2% higher for mAP@[.5, .95] than the Fast RCNN counterpart under the same

protocol (Table 7). This indicates that RPN performs excellent for improving the localization accuracy at higher IoU thresholds. Using the COCO trainval set to train, Faster RCNN has 42.7% mAP@0.5 and 21.9% mAP@[.5,.95] on the COCO test-dev set. Figure 6 shows some results on the MS COCO test-dev set.

Faster R-CNN in ILSVRC & COCO 2015 competitions We have demonstrated that Faster R-CNN benefits more from better features, thanks to the fact that the RPN completely learns to propose regions by neural networks. This observation is still valid even when one increases the depth substantially to over 100 layers [18]. Only by replacing VGG-16 with a 101layer residual, net (ResNet-101), the Faster R-CNN system increases the mAP from 41.5%/21.2% (VGG16) to 48.4%/27.2% (ResNet-101) on the COCO val set. With other improvements orthogonal to Faster RCNN, He et al. obtained a single-model result of 55.7%/34.9% and an ensemble result of 59.0%/37.4% on the COCO test-dev set, which won the 1st place in the COCO 2015 object detection competition. The same system also won the 1st place in the ILSVRC 2015 object detection competition, surpassing the second place by absolute 8.5%. RPN is also a building block of the 1st-place winning entries in ILSVRC 2015 localization and COCO 2015 segmentation competitions, for which the details are available in [70] respectively.

training data	2017 test	2012 test
VOC07	69.9	67.0
VOC07+12	73.2	-
VOC07++12	-	70.4
COCO (no VOC)	76.1	73.0
COCO+VOC07+12	78.8	-
COCO+VOC07++12	-	75.9

Table 8: Detection mAP (%) of Faster R-CNN on PASCAL VOC 2007 test set and 2012 test set using different training data. The model is VGG-16. “COCO” denotes that the COCO trainval set is used for training.

4.3 From MS COCO to PASCAL VOC

Large-scale data is of crucial importance for improving deep neural networks. Next, we investigate how the MS COCO dataset can help with the detection performance on PASCAL VOC.

As a simple baseline, we directly evaluate the COCO detection model on the PASCAL VOC dataset, without fine-tuning on any PASCAL VOC data. This evaluation is possible because the categories on COCO are a superset of those on PASCAL VOC. The categories that are exclusive on COCO are ignored in this experiment, and the softmax layer is performed only on the 20 categories plus background. The mAP under this setting is 76.1% on the PASCAL VOC 2007 test

set (Table 8).This result is better than that trained on VOC07+12 (73.2%) by a good margin, even though the PASCAL VOC data are not exploited.

Then we fine-tune the COCO detection model on the VOC dataset. In this experiment, the COCO model is in place of the ImageNet-pre-trained model (that is used to initialize the network weights), and the Faster R-CNN system is fine-tuned. Doing so leads to 78.8% mAP on the PASCAL VOC 2007 test set. The extra data from the COCO set increases the mAP by 5.6%. Table 6 shows that the model trained on COCO+VOC has the best AP for every individual category on PASCAL VOC 2007. Similar improvements are observed on the PASCAL VOC 2012 test set (Table 8 and Table 5). We note that the test-time speed of obtaining these strong results is still about 200ms per image.

4.4 Results

After training, some effect on the testset is shown below:

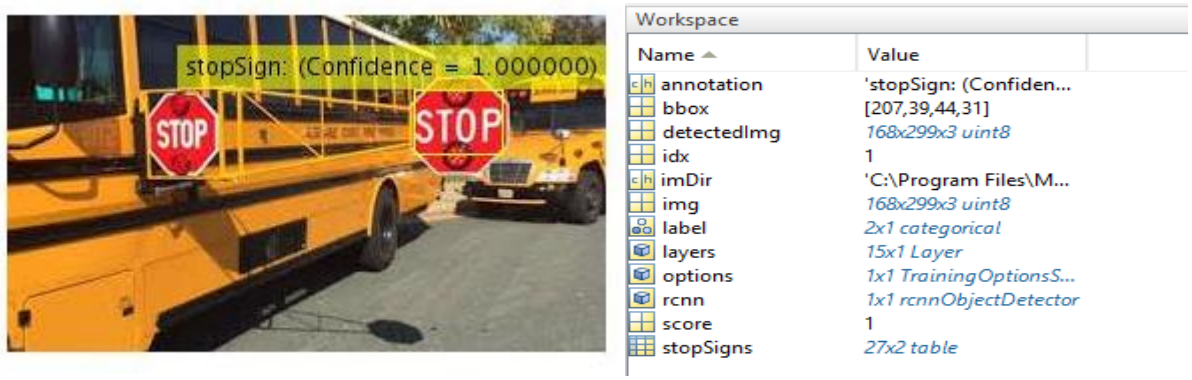

Figure 4.2: Train R-CNN object detector

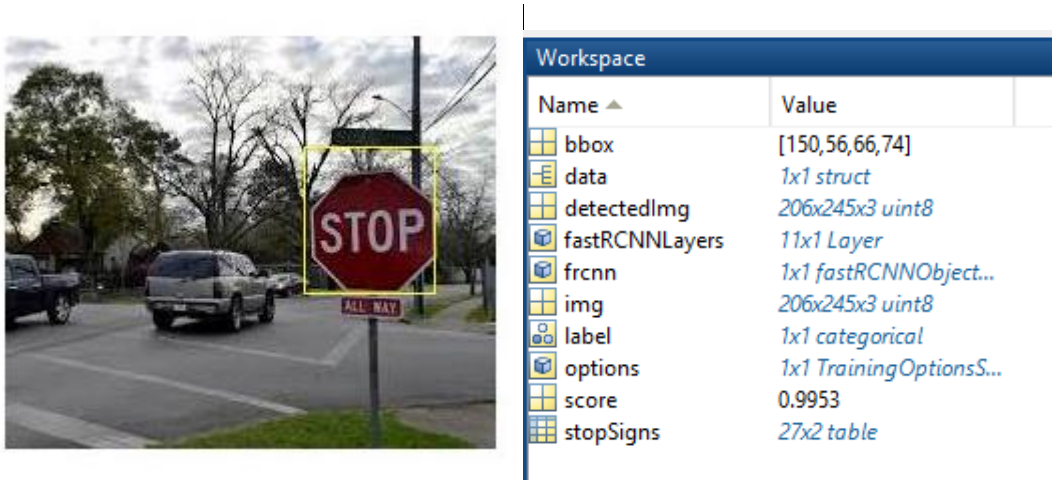

Figure 4.3: Train Fast R-CNN object detector

Figure 4.3: Train Faster R-CNN object detector

Chapter 5 Conclusion

Due to its powerful learning ability and advantages in dealing with occlusion, scale transformation and background switches, deep learning based object detection has been a research hotspot in recent years. This paper provides on deep learning based object detection frameworks that handle different sub-problems, such as occlusion, clutter and low resolution, with different degrees of modifications on R-CNN. By sharing convolutional features with the down-stream detection network, the region proposal step is nearly cost-free. They only require a distribution of classification scores per detection. Depending on the specific task, an object detector that will report objects of unknown classes is also important. Additionally, we propose a sample weighting scheme to balance selections among classes.

We evaluate the proposed metrics on the PASCAL VOC 2012 dataset and offer quantitative and qualitative results and analysis. We show that the proposed metrics are able to guide the annotation process efficiently that leads to superior performance in comparison to a random selection baseline. In our experimental evaluation, the Sum metric is able to achieve best results overall which can be attributed to the fact that it tends to select batches with many single objects in it. However, the targeted scenario is an application with huge amounts of unlabeled data where we consider the amount of images to be evaluated as more critical than the time needed to draw single bounding boxes. To expedite annotation, our approach could be combined with a weakly supervised learning approach as presented. We also showed that our weighting scheme leads to even increased accuracies.

Moreover, our proposed metrics are not restricted to deep object detection and could be applied to arbitrary object detection methods if they fulfill there quirements. It only requires a complete distribution of classifications scores per detection.
Our method enables a unified, deep-learning-based object detection system to run. The learned RPN also improves region proposal quality and thus the overall object detection accuracy.

Reference

[1] M. Everingham, L.Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results, 2007.

[2] R. Girshick, “Fast r-cnn,” in ICCV, 2015.

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 2014.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In ECCV. 2014.

[5] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015.

[6] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.

[7] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A.W. Smeulders. Selective search for object recognition. IJCV, 2013.

[8] C. L. Zitnick and P. Doll´ar. Edge boxes: Locating object proposals from edges. In ECCV, 2014.

[9] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Doll´ar, and C. L. Zitnick, “Microsoft COCO: Common Objects in Context,” in European Conference on Computer Vision (ECCV), 2014.

[10] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. Imagenet: a large-scale hierarchical image database. In Proc. IEEE Int’l Conf. Computer Vision and Pattern Recognition, 2009.

[11] G. E. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition — the shared views of four research groups. IEEE Signal Processing Magazine, 2012.

[12] G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. Neural Computation, 18:1527–1544, 2006.

[13] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miler. Labeled faces in the wild: a database for studying face recognition in unconstrained environments. Technical report, University of Massachusetts, Amherst, 2007.

[14] A. Krizhevsky, L. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In Proc. Neural Information Processing Systems, 2012.

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86:2278– 2324, 1998.

[16] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. InCVPR, 2015. [17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Nature, 323(99):533–536, 1986.

[18] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In Proc. Int’l Conf. Learning Representations, 2014.

[19] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in ICML, 2015.
[20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. arXiv: 1409.4842, 2014.
[21] J. Deng, A. Berg, S. Satheesh, H. Su, A. Khosla, and L. FeiFei. ImageNet Large Scale Visual Recognition Competition 2012 (ILSVRC2012).
[22] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In NIPS, 2015.
[23] D. G. Lowe, “Distinctive image features from scale-invariant key points,” Int. J. of Comput. Vision, vol. 60, no. 2, pp. 91–110, 2004.
[24] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in CVPR, 2005.
[25] R. Lienhart and J. Maydt, “An extended set of haar-like features for rapid object detection,” in ICIP, 2002.
[26] C. Cortes and V. Vapnik, “Support vector machine,” Machine Learning, vol. 20, no. 3, pp. 273–297, 1995.
[27] Y. Freund and R. E. Schapire, “A desicion-theoretic generalization of on-line learning and an application to boosting,” J. of Comput. & Sys. Sci., vol. 13, no. 5, pp. 663–671, 1997.
[28] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, “Object detection with discriminatively trained part-based models,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 32, pp. 1627–1645, 2010.
[29] P. Druzhkov and V. Kustikova, “A survey of deep learning methods and software tools for image classification and object detection,” Pattern Recognition and Image Anal., vol. 26, no. 1, p. 9, 2016.
[30] C. Szegedy, A. Toshev, and D. Erhan. Deep neural networks for object detection. In NIPS, 2013.
[31] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In CVPR, 2014.
[32] J. Dai, K. He and J. Sun. Convolutional feature masking for joint object and stuff segmentation. In CVPR, 2015.
[33] S. Ren, K. He, R. Girshick, X. Zhang, and J. Sun. Object detection networks on convolutional feature maps. arXiv: 1504.06066, 2015.
[34] J. Li, X. Liang, J. Li, T. Xu, J. Feng, and S. Yan, “Multi-stage object detection with group recursive learning,” arXiv:1608.05159, 2016.
[35] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional architecture for fast feature embedding,” in ACM MM, 2014.
[36] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, pp. 436–444, 2015.
[37] K. Kavukcuoglu, R. Fergus, Y. LeCun et al., “Learning invariant features through topographic filter maps,” in CVPR, 2009.
[38] K. Kavukcuoglu, P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, and Y. LeCun, “Learning convolutional feature hierarchies for visual recognition,” in NIPS, 2010.
[39] H. Noh, S. Hong, and B. Han, “Learning deconvolution network for semantic segmentation,” in ICCV, 2015.

[40] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky, “Neural codes for image retrieval,” in ECCV, 2014.
[41] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, “Subcategory-aware convolutional neural networks for object proposals and detection,” in WACV, 2017.
[42] Z.-Q. Zhao, H. Bian, D. Hu, W. Cheng, and H. Glotin, “Pedestrian detection based on fast r-cnn and batch normalization,” in ICIC, 2017.
[43] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, “Multimodal deep learning,” in ICML, 2011.
[44] Z. Wu, X. Wang, Y.-G. Jiang, H. Ye, and X. Xue, “Modeling spatial temporal clues in a hybrid deep learning framework for video classification,” in ACM MM, 2015.
[45] Z. Yang and R. Nevatia, “A multi-scale cascade fully convolutional network face detector,” in ICPR, 2016.
[46] Y. Li, K. He, J. Sun, et al. R-fcn: Object detection via region based fully convolutional networks. In Advances in Neural Information Processing Systems, pages 379–387, 2016.
[47] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Back propagation applied to handwritten zip code recognition. Neural computation, 1989.
[48] C. Szegedy, S. Reed, D. Erhan, and D. Anguelov. Scalable, high-quality object detection. arXiv: 1412.1441v2, 2015.\
[49] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, “Real time multi-person 2d pose estimation using part affinity fields,” in CVPR, 2017.
[50] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, “Deconvolutional networks,” in CVPR, 2010.
[51] T.-Y. Lin, P. Doll´ar, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, “Feature pyramid networks for object detection,” in CVPR, 2017.
[52] K. He, G. Gkioxari, P. Doll´ar, and R. B. Girshick, “Mask r-cnn,” in ICCV, 2017.
[53] M. Najibi, M. Rastegari, and L. S. Davis, “G-cnn: an iterative grid based object detector,” in CVPR, 2016.
[54] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in ECCV, 2016.
[55] G. E. Hinton, A. Krizhevsky, and S. D. Wang, “Transforming autoencoders,” in ICANN, 2011.
[56] G. W. Taylor, I. Spiro, C. Bregler, and R. Fergus, “Learning invariance through imitation,” in CVPR, 2011.
[57] X. Ren and D. Ramanan, “Histograms of sparse codes for object detection,” in CVPR, 2013.
[58] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun, “Pedestrian detection with unsupervised multi-stage feature learning,” in CVPR, 2013.
[59] P. Kr¨ahenb¨uhl and V. Koltun, “Geodesic object proposals,” in ECCV, 2014.
[60] P. Arbel´aez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik, “Multiscale combinatorial grouping,” in CVPR, 2014.
[61] W. Kuo, B. Hariharan, and J. Malik, “Deep box: Learning objectness with convolutional networks,” in ICCV, 2015.

[62] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Doll´ar, “Learning to refine object segments,” in ECCV, 2016.
[63] Y. Zhang, K. Sohn, R. Villegas, G. Pan, and H. Lee, “Improving object detection with deep convolutional networks via bayesian optimization and structured prediction,” in CVPR, 2015.
[64] S. Gupta, R. Girshick, P. Arbel´aez, and J. Malik, “Learning rich features from rgb-d images for object detection and segmentation,” in ECCV, 2014.
[65] W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, C.-C. Loy et al., “Deepid-net: Deformable deep convolutional neural networks for object detection,” in CVPR, 2015.
[66] K. Lenc and A. Vedaldi, “R-cnn minus R,” arXiv: 1506.06981, 2015.
[67] S. Lazebnik, C. Schmid, and J. Ponce, “Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories,” in CVPR, 2006.
[68] J. Xue, J. Li, and Y. Gong, “Restructuring of deep neural network acoustic models with singular value decomposition.” in Interspeech, 2013.
[69] S. Bell, C. Lawrence Zitnick, K. Bala, and R. Girshick, “Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks,” in CVPR, 2016.
[70] J. Dai, K. He and J. Sun, “Instance-aware semantic segmentation via multi-task network cascades,” in CVPR, 2016.
[71] S. Brahmbhatt, H. I. Christensen, and J. Hays, “Stuffnet: Using stuffto improve object detection,” in WACV, 2017.
[72] T. Kong, A. Yao, Y. Chen, and F. Sun, “Hypernet: Towards accurate region proposal generation and joint object detection,” in CVPR, 2016.
[73] A. Pentina, V. Sharmanska, and C. H. Lampert, “Curriculum learning of multiple tasks,” in CVPR, 2015.
[74] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, “A unified multi-scale deep convolutional neural network for fast object detection,” in ECCV, 2016.
[75] Y. Zhu, R. Urtasun, R. Salakhutdinov, and S. Fidler, “segdeepm: Exploiting segmentation and context in deep neural networks for object detection,” in CVPR, 2015.
[76] S. Gidaris and N. Komodakis, “Object detection via a multi-region and semantic segmentation-aware cnn model,” in CVPR, 2015.
[77] M. Schuster and K. K. Paliwal, “Bidirectional recurrent neural networks,” IEEE Trans. Signal Process., vol. 45, pp. 2673–2681, 1997.
[78] S. Zagoruyko, A. Lerer, T.-Y. Lin, P. O. Pinheiro, S. Gross, S. Chintala, and P. Doll´ar, “A multipath network for object detection,” arXiv: 1604.02135, 2016.

Acknowledgment

Mainly thanks to the supervisor and the organizations or individuals who made special contributions to this research work:

Also express thanks to the fund, department, enterprise, organization or individual who sponsored or backed this thesis; (The fund item should includes the fund’s name, the item’s name, number and executives, the time the research started and the time completed.)

And gratitude to the organization and individual who assists in the completion or offer convenience for the research work.