# Deep learning for real-world object detection

Xiongwei WU
*Singapore Management University*, xwwu@smu.edu.sg

# DEEP LEARNING FOR REAL-WORLD OBJECT DETECTION

**WU XIONGWEI**

SINGAPORE MANAGEMENT UNIVERSITY

2020

# Deep Learning for Real-World Object Detection

**WU Xiongwei**

Submitted to School of Information Systems in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Computer Science

Singapore Management University
2020

I hereby declare that this PhD dissertation is my original work
and it has been written by me in its entirety.
I have duly acknowledged all the sources of information
which have been used in this dissertation.

This PhD dissertation has also not been submitted for any degree
in any university previously.

Wu Xiongwei

---

WU Xiongwei

15 July 2020

# Deep Learning for Real-World Object Detection

WU Xiongwei

## Abstract

Despite achieving significant progresses, most existing detectors are designed to detect objects in academic contexts but consider little in real-world scenarios. In real-world applications, the scale variance of objects can be significantly higher than objects in academic contexts; In addition, existing methods are designed for achieving localization with relatively low precision, however more precise localization is demanded in real-world scenarios; Existing methods are optimized with huge amount of annotated data, but in certain real-world scenarios, only a few samples are available. In this dissertation, we aim to explore novel techniques to address these research challenges to make object detection algorithms practical for real-world applications.

The first problem is scale-invariant detection. Detecting objects with multiple scales is covered in existing detection benchmarks. However, in real-world applications the scale variance of objects is extremely high and thus it requires more discriminative features. Face detection is a suitable benchmark to evaluate scale-invariant detection due to the vastly different scales of faces. In this dissertation, we propose a novel framework of "Feature Agglomeration Networks" (FAN) to build a new single stage face detector. A novel feature agglomeration block is proposed to enhance low-level feature representation and the model is optimized in a hierarchical manner. FAN achieved state-of-the-art results in real world face detection benchmarks with real-time inference speed.

The second problem is high-quality detection. This challenge requires detectors to predict more precise localization. In this dissertation, we propose two novel detection frameworks for high-quality detection: "Bidirectional Pyramid Networks" (BPN) and "KPNet". In BPN, a Bidirectional Feature Pyramid structure is proposed for robust feature representations, and a Cascade Anchor Refinement is proposed to gradually refine the quality of pre-designed anchors. To eliminate the initial anchor design step in BPN, KPNet is proposed which automatically learns to optimize a dynamic set of high-quality keypoints without heuristic anchor design. Both BP-N and KPNet show significant improvement over existing on MSCOCO dataset, especially in high quality detection settings.

The third problem is few-shot detection, where only a few training samples are available. Inspired by the principle of meta-learning methods, we propose two nov-el meta-learning based few-shot detectors: "Meta-RCNN" and "Meta Constrastive Detector" (MCD). Meta-RCNN learns an binary object detector in an episodic learning paradigm on the training data with a class-aware attention module, and it can be end-to-end meta-optimized. Based on Meta-RCNN, MCD follows the principle of contrastive learning to enhance the feature representation for few-shot detection, and a new hard negative sampling strategy is proposed to address im-balance of training samples. We demonstrate the effectiveness of Meta-RCNN and MCD in few-shot detection on Pascal VOC dataset and obtain promising results.

The proposed techniques address the problems discussed and show significant improvement on real-world utility.

# Table of Contents

# List of Figures

# List of Tables

# Acknowledgments

First and foremost, I wish to thank my advisers Professor Steven HOI for his patient support and generous mentorship. It's a huge honor and inspiration to work in his team. It's also my honor to have Professor Jing JIANG, Professor Hady W. LAUW and Professor Guoshen LIN on my dissertation committee.

I thank my seniors doyen SHAOO, Jing LU, Yue WU, Chenghao LIU and Peilin ZHAO for their extremely kind help and support. Many thanks to my collaborators, jialiang ZHANG, Daoxin ZHANG and Zhihao WANG. To work with all of you is an unforgettable memory to me.

I also thank the support and help from the whole SMU community. They give me a wonderful and advanced platform to conduct research tasks.

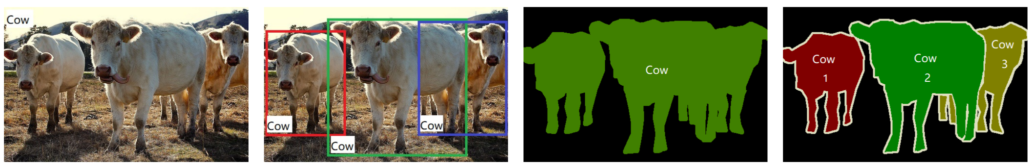
Last but not least, I thank the support of my family members in China. Their unfailing support gives me the energy to overcome all difficulties I meet.

# Chapter 1

# Introduction

In this chapter, we first briefly introduce the background of object detection, and then we point out the limitations when applying generic detection algorithms into real-world problems. In this dissertation, we cover three real-world detection problems: scale-invariant detection, high-quality detection and few-shot detection. For each of the real-world problem, we present the main challenges of applying existing detection algorithms, and introduce our proposed methods. Finally we give the contribution summary and the structure of this dissertation.

## 1.1 Background



(a) Image Classification (b) Object Detection (c) Semantic Segmentation (d) Instance Segmentation

Figure 1.1: Comparison of different visual recognition tasks in computer vision. (a) "Image Classification" only needs to assign categorical class labels to the image; (b) "Object detection" not only predict categorical labels but also localize each object instance via bounding boxes; (c) "Semantic segmentation" aims to predict categorical labels for each pixel, without differentiating object instances; (d) "Instance segmentation", a special setting of object detection, differentiates different object instances by pixel-level segmentation masks.

Object detection is a fundamental computer vision problem. In the field of computer vision, there are several fundamental visual recognition problems: image classification [62], object detection and instance segmentation [48, 60], and semantic segmentation [14] (see Fig. 1.1). In particular, image classification (Fig 1.1.1(a)), aims to recognize semantic categories of objects in a given image. Object detection not only recognizes object categories, but also predicts the location of each object by a bounding box (Fig. 1.1(b)). Semantic segmentation (Fig. 1.1(c)) aims to predict pixel-wise classifiers to assign a specific category label to each pixel, thus providing an even richer understanding of an image. However, in contrast to object detection, semantic segmentation does not distinguish between multiple objects of the same category. A relatively new setting at the intersection of object detection and semantic segmentation, named "instance segmentation" (Fig. 1.1(d)), is proposed to identify different objects and assign each of them a separate categorical pixel-level mask. In fact, instance segmentation can be viewed as a special setting of object detection, where instead of localizing an object by a bounding box, pixel-level localization is desired. A good detection algorithm should have a strong understanding of semantic cues as well as the spatial information about the image. And thus, object detection is the basic step towards many computer vision applications, such as face recognition [189, 188, 124], pedestrian detection [105, 69, 2], video analysis [86, 136], and logo detection [67, 186, 185].

After the success of applying deep convolutional neural networks (DCNN) into image classification task [96], object detection also achieved remarkable progresses based on DCNN backbone architecture [162, 45]. Currently, deep learning based object detection frameworks can be primarily divided into two families: (i) two-stage detectors, such as Region-based CNN (R-CNN) [48] and its variants [47, 162, 118] and (ii) one-stage detectors, such as YOLO [157] and its variants [159, 123]. Two-stage detectors first use a proposal generator to generate a sparse set of proposals and extract features from each proposal, followed by region classifiers which predict the category of the proposed region. One-stage detectors directly make categorical prediction of objects on each location of the feature maps without the cascaded region classification step. Two-stage detectors commonly achieve better detection performance and report state-of-the-art results on public benchmarks, while one-stage detectors are significantly more time-efficient and have greater applicability to real-time object detection.

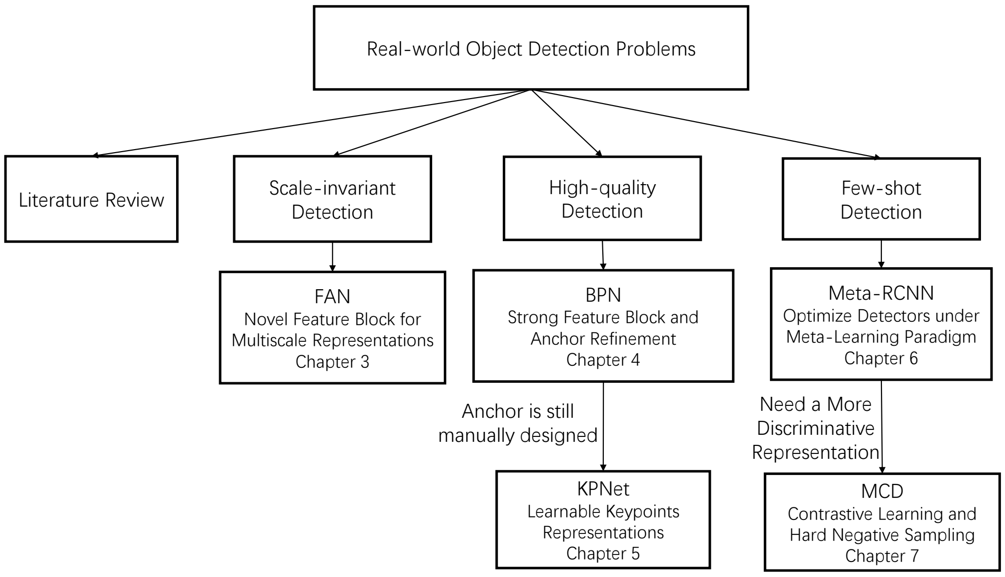

Real-world Object Detection Problems
Literature Review
Scale-invariant Detection
High-quality Detection
Few-shot Detection
FAN
Novel Feature Block for Multiscale Representations
Chapter 3
BPN
Strong Feature Block and Anchor Refinement
Chapter 4
Meta-RCNN
Optimize Detectors under Meta-Learning Paradigm
Chapter 6
Anchor is still manually designed
Need a More Discriminative Representation
KPNet
Learnable Keypoints Representations
Chapter 5
MCD
Contrastive Learning and Hard Negative Sampling
Chapter 7

Figure 1.2: The organization of the dissertation

## 1.2 Real-world Problems

Despite the state-of-the-art performances on public benchmarks, directly applying existing detection frameworks into real-world problems in not optimal. Real-world problems open many new challenges such as insufficient training samples, large scale variance and high localization demand. In this section, we introduce three important real-world problems: **scale-invariant detection**, **high-quality detection** and **few-shot detection**. We present the main difficulties of these problems and introduce our proposed methods. Figure 1.2 shows the organization of the topics in this dissertation.

### 1.2.1 Scale-invariant Detection

The first problem is to detect objects in multiple scales. Although existing detection benchmarks such as MSCOCO [120] also present multiscale property of the objects, the scale variance of objects in real-world problems is significantly larger, which requires the detectors to learn more robust features for prediction. For this real-world problem, we select face detection as our research topic. Face detection is a real world computer vision problem to detect human face in the wild. There is one critical difference between face detection with generic detection: the scale ranges of object in face detection is much larger than objects in generic detection. And thus face detection is suitable to validate the algorithms for scale-invariant detection. Considering the properties of face detection, a carefully designed feature representation of face objects is required to handle varied scales of faces and diverse characteristics of real-world faces captured from different scenarios. One existing routine is to train multi-shot single-scale detectors by using the idea of image pyramid to train multiple separate single-scale detectors each of which is tuned for one specific scale (e.g., the HR detector in [72] trained multiple scale-specific RPN detectors). However, such approach with the image pyramid is computationally expensive since it has to pass a very deep network multiple times during inference, which is also not suitable in real-world applications.

In this dissertation, we propose a novel face detector, **FAN** [232] with real-time inference speed, which achieves state-of-the-art results in many public datasets and is extremely robust to faces with various scales. FAN consists two novel structures, Feature Agglomerate Block and Hierarchical Loss. Feature Agglomerate Block integrates adjacent layer features in hierarchical way, which is more robust to learn scale-invariance features, while Hierarchical Loss makes the whole training process more stable. FAN is discussed in Chapter 3 in detail.

### 1.2.2 High-quality Detection

The second problem is to detect objects with high IoU thresholds. Most existing object detectors are designed for achieving localization with relatively low-quality precision, i.e., with a default IoU threshold of 0.5. However, real-world applications such as autopilot, requires detectors with high localization ability, where the goal is to achieve higher quality localization precision (IoU=0.7 etc.). Unfortunately, the detection performance often drops significantly for more precise prediction [10]. There are two major difficulties in real-world high-quality detection: (i) Naively increasing the IoU threshold during training is not effective since a high IoU will lead to significantly less amount of positive training samples and thus make the training results prone to overfitting, especially for single-shot SSD-like detectors. (ii) In real-world applications, anchor design is non-trivial which requires domain knowledge and makes the problem even harder to address. The ill-designed anchors and the heuristic IoU matching strategy significantly can reduce detection accuracy.

In this dissertation we are motivated to investigate an effective single-shot detection scheme towards high-quality object detection. We first proposed frameworks **BPN**. Our proposed framework BPN [214] is based on a set of pre-defined anchors to match objects and it calibrates the shape of the anchors to better match the objects during training. Two novel components are proposed in BPN: anchor refinement and bidirectional feature pyramid. Bidirectional feature pyramid enhances the feature representation of objects, while anchor refinement calibrates the shape of prior anchors to better match the objects. BPN achieves state-of-the-art performance in high-quality scenario while still keep real-time inference speed.

However, in BPN the initial anchors are still manually designed, and the IoU threshold is also set heuristically. Based on BPN, we propose an anchor-free framework **KPNet** which directly regresses object predictions from the learned keypoints instead of anchors. Different with the existing anchor-free methods where the keypoints layout is fixed, KPNet is able to automatically optimize a dynamic set of high-quality keypoints by keypoints predictor, and eliminates the heuristic anchor design step. A set of high-quality keypoints can be automatically learned from image pixels with different types of objects, and thus shows significant improvement in detection accuracy, especially in high-quality settings. We will discuss BPN in Chapter 4 and KPNet in Chapter 5 in detail.

### 1.2.3 Few-shot Detection

The last problem is to train detectors with only a few annotated images. Although existing deep learning based detection frameworks achieve remarkable progresses, all these methods are data hungry, which requires large amounts of annotated data to learn an immense number of parameters. For object detection, annotating the data is very expensive (far more than image classification), as it requires not only identifying the categorical labels for every object in the image, but also providing accurate localization information through bounding box coordinates. Moreover, in some real-world applications, such as medical research, it's often impossible to even collect sufficient data to annotate. This warrants a need for effective detectors that can generalize well from small amounts of annotated data. We refer to the problem of learning detectors from limited labeled data as *few-shot detection*. For example, in *one-shot* detection, only one image is available with objects of interest annotated, and a detector needs to train on just this image and generalize. When presented with such small amounts of annotated data, traditional detectors tend to suffer from overfitting.

Inspired by the fact that humans can learn a new concept from limited training data, we aim to develop few-shot detection methods based on the principle of meta-learning [201]. In this dissertation, we first proposed few-shot detection algorithm: **Meta-RCNN**. Meta-RCNN is an end to end trainable meta object detector, which follows the episodic learning paradigm of meta-learning, where multiple few-shot tasks are simulated based on a give meta-train dataset. The whole model is meta-optimized through a novel class-specific attention module followed by a weight-shared binary classifier. This novel design significantly reduces parameter numbers, and thus is robust and suitable to few-shot detection problem. Enjoying the merit of the episodic learning paradigm, Meta-RCNN shows significantly improvement on public benchmarks.

However, the binary classifier in Meta-RCNN potentially makes it harder to learn discriminative feature representations for different objects. In few-shot detection problem, a more discriminative feature representation is required. Based on Meta-RCNN, we further designed a new meta-learning based framework **MCD** which follows the principle of contrastive learning, and is trained by hard negative sampling strategy. MCD is also learned under the episodic learning paradigm, and the contrastive learning module enables it learning a discriminative representations in a synergic manner. Furthermore, the hard negative sampling strategy further help the training process of MCD and increases the effectiveness. We will discuss Meta R-CNN in Chapter 6 and MCD in Chapter 7 in detail.

### 1.2.4 Other Real-world Problems

There are some other detection problems in real applications such as logo detection or video detection. Logo detection and recognition has been studied in computer vision and pattern recognition literature [151, 165, 83]. From a computer vision perspective, *logo recognition*, which can be viewed as a special case of image recognition, aims to recognize the logo name of an input image, and *logo detection* is often more challenging in that it not only needs to recognize the logo name but also need to find the locations of logo objects in the input image. Logo detection and recognition found a wide range of applications in many domains, such as product brand recognition for intellectual property protection in e-commerce platforms, vehicle logo recognition for intelligent transportation [151], product brand management on social media [41], etc. However, most existing studies are based on traditional computer vision and regular machine learning approaches, and very few has explored deep learning techniques. One reason is probably because of lacking large-scale datasets as deep learning methods are often very data-intensive. The other reason is logo objects are usually very small with complex contexts, which is hard for current algorithms to detect. In my research work a large scale logo datasets "LOGONet" [67] is collected and we conduct a systematic ablation study based on it. This work is more focused on engineering effort so we omit the detail of it.

## 1.3 Summary of Contributions

In this dissertation, we identified limitations of existing object detection algorithms in real-world problems. Specifically, we analyze three real-world problems: scale-variant detection, high-quality detection and few-shot detection. We proposed novel algorithms addressing these problems respectively, and make detection algorithms applicable to real world applications. We made following contributions in the field of applying detection algorithms into real-world problems:

**Scale-invariant Detection:** A novel framework of Feature Agglomeration Networks (FAN) is proposed for single stage face detection, which creates a new effective feature pyramid with rich semantics at all scales by introducing a new hierarchical agglomerative connection module to agglomerate multi-scale features via a hierarchical agglomerative manner, with a more effective Hierarchical Loss based training scheme to train the proposed FAN model in an end-to-end manner, which enables us to learn discriminative features of the feature pyramid effectively; Comprehensive experiments on several public Face Detection benchmarks have been conducted to evaluate the effectiveness of the proposed FAN framework. FAN detector shows promising results which not only achieves the state-of-the-art performances but also runs efficiently with real-time speed on GPU.

**High-quality Object Detection:** A novel framework of Bidirectional Pyramid Networks (BPN) for single-shot object detector that is designed directly towards high-quality detection. BPN consists of a novel Bidirection Feature Pyramid Structure that improves the vanilla Feature Pyramid and a novel Cascaded Anchor Refinement scheme to gradually improve the quality of predefined anchors which are often inaccurate at the beginning. Extensive experiments on PASCAL VOC and MSCOCO showed that the proposed method achieved the state-of-the-art results for high-quality object detection while maintaining the advantage of computational efficiency.

Although the anchor shapes are refined during training, BPN still requires initial manually designed anchors. Further we propose an anchor-free detector KPNet which directly predicts objects via learnable keypoints without manual design. A dynamic set of high-quality keypoints is automatically optimized from the image pixels of different types of objects, which learns rich feature representations. Extensive experiments on MSCOCO showed that the proposed KPNet achieved the state-of-the-art results.

**Few-shot Detection:** A novel framework of Meta-RCNN is proposed for few-shot detection based on meta-learning. The new proposed meta-optimization strategies make Meta-RCNN robust and is suitable for few-shot detection. Notably, it is based on vanilla Faster RCNN but all the components of the detector, the object classifier, the RPN and the bounding box regressor, can be meta-optimized. The whole model is optimized through a novel class-specific attention module. We conduct extensive experiments and obtain promising results.

To enhance the representation ability of Meta-RCNN, a novel framework of Meta Contrastive Detector (MCD) is proposed based on Meta-RCNN. MCD imposes contrastive loss between positive and negative samples to achieve more discriminative representation. In addition, we identify two types of negative samples that have different distributions and effects, and propose a new negative sampling strategy to further improve the training. The meta-contrastive learning framework with the new negative sampling strategy makes MCD a state-of-the-art detector in few-shot settings. We demonstrate the effectiveness of MCD on both Pascal VOC and MSCOCO datasets in few-shot settings.

## 1.4 Dissertation Structure

As shown in Figure 1.2, the remainder of this dissertation is organized as follows: Chapter 2 is the literature review which gives a comprehensive understanding of object detection based on deep learning. Next we have studied three real-world detection problems and thus we summarize our works into three parts: scale-invariant detection (Part **I**), high-quality detection (Part II) and few-shot detection (Part III).

In Part **I**, we have one chapter (Chapter 3) to study algorithms for real-world scale-invariant detection problem. We select face detection benchmarks to validate our method and we propose FAN with novel feature agglomeration blocks and hierarchical learning method, which achieves state-of-the-art results in face detection while still keeping real-time inference speed.

In Part **II**, we have two chapters (Chapter 4 and Chapter 5) to investigate the methods in high quality detection. In Chapter 4, the new proposed framework BPN tries to refine ill-defined anchors by cascaded way with more robust bidirectional feature pyramid blocks. BPN is a real-time detector and achieves state-of-the-art results in high quality scenarios. In Chapter 5, we point out the limitations of BPN of using manually defined-anchors and heuristic IoU-based matching strategy. A new anchor-free framework KPNet is proposed which automatically optimizes a dynamic set of high quality keypoints from image pixels, without manual design.

In Part **III**, we have two chapters (Chapter 6 and Chapter 7) to study few-shot detection by training detectors under the paradigm of meta-learning. In Chapter 6 we propose the Meta-RCNN which follows the episodic learning principle with a class-aware attention module. In Chapter 7 we present MCD assigns the ability to Meta-RCNN of learning discriminative feature representations.

Finally, Chapter 8 talks about the future direction of object detection and Chapter 9 concludes this dissertation.

# Chapter 2

# Literature Review

In this section, we present a literature review of the object detection based on deep learning. The goal of this section is to present a comprehensive understanding of deep learning based object detection algorithms. Fig. 2.1 shows a taxonomy of key methodologies to be covered in this section. We review various contribution-s in deep learning based object detection and categorize them into three groups: detection components, learning strategies, and applications & benchmarks. For detection components, we first introduce two detection settings: bounding box level (bbox-level) and pixel mask level (mask-level) localization. Bbox-level algorithms require to localize objects by rectangle bounding boxes, while more precise pixel-wise masks are required to segment objects in mask-level algorithms. Next, we summarize the representative frameworks of two detection families: two-stage detection and one-stage detection. Then we give a detailed survey of each detection component, including backbone architecture, proposal generation and feature learning. For learning strategies, we first highlight the importance of learning strategy of detection due to the difficulty of training detectors, and then introduce the optimization techniques for both training and testing stages in detail. Finally, we review some real-world object detection based applications including face detection, pedestrian detection, logo detection and video analysis. We also discuss publicly available and commonly used benchmarks and evaluation metrics for these detection tasks.

Finally we show the state-of-the-art results of different detection scenarios on public benchmarks over the recent years.

The rest of the section are organized as follows: The history of deep learning based detection algorithms is listed in Section 2.1. The problem settings of detection algorithms are listed in Section 2.2. The details of detector components are listed in Section 2.3. Then the learning strategies are presented in Section 2.4. Detection algorithms for real-world applications and benchmarks are provided in Section 2.5 and Section 2.6.

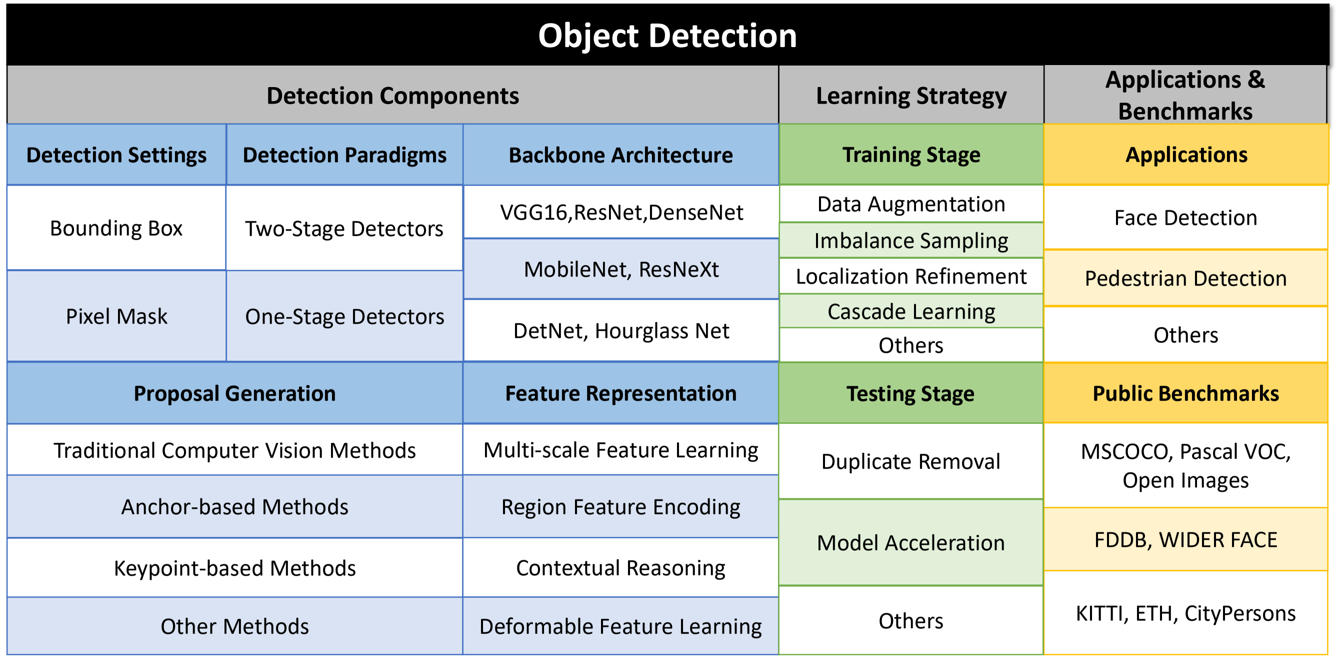


Figure 2.1: Taxonomy of key methodologies in this section. We categorize various contributions for deep learning based object detection into three major categories: Detection Components, Learning Strategies, Applications and Benchmarks. We review each of these categories in detail.

## 2.1 History of Detection Algorithms

In the early stages, before the deep learning era, the pipeline of object detection was divided into three steps: (i) proposal generation; (ii) feature vector extraction; and (iii) region classification. During proposal generation, the objective was to search locations in the image which may contain objects. These locations are also called regions of interest (roi). An intuitive idea is to scan the whole image with sliding windows [200, 202, 59, 26, 203]. In order to capture information about multi-scale and different aspect ratios of objects, input images were resized into different scales and multi-scale windows were used to slide through these images. During the second step, on each location of the image, a fixed-length feature vector was obtained from the sliding window, to capture discriminative semantic information of the region covered. This feature vector was commonly encoded by low-level visual descriptors such as SIFT (Scale Invariant Feature Transform) [128], Haar [116], HOG (Histogram of Gradients) [26] or SURF (Speeded Up Robust Features) [4], which showed a certain robustness to scale, illumination and rotation variance. Finally, in the third step, the region classifiers were learned to assign categorical labels to the covered regions. Commonly, support vector machines (SVM) [65] were used here due to their good performance on small scale training data. In addition, some classification techniques such as bagging [142], cascade learning [203] and adaboost [37] were used in region classification step, leading to further improvements in detection accuracy.

Most of the successful traditional methods for object detection focused on carefully designing feature descriptors to obtain embedding for a region of interest. With the help of good feature representations as well as robust region classifiers, impressive results [227, 32] were achieved on Pascal VOC dataset [31] (a publicly available dataset used for benchmarking object detection). Notably, deformable part based machines (DPMs) [33], a breakthrough detection algorithm, were 3-time winners on VOC challenges in 2007, 2008 and 2009. DPMs learn and integrate multiple

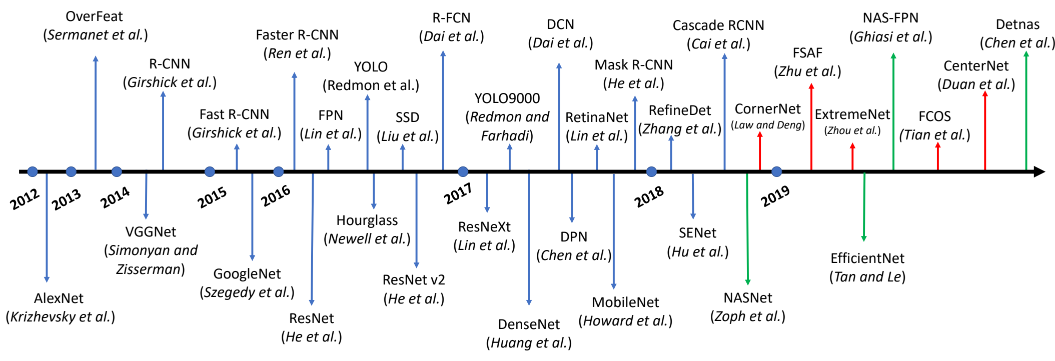


Figure 2.2: Major milestone in object detection research based on deep convolution neural networks since 2012. The trend in the last year has been designing object detectors based on anchor-free (in red) and AutoML (in green) techniques, which are potentially two important research directions in the future. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

part models with a deformable loss and mine hard negative examples with a latent SVM for discriminative training. However, during 2008 to 2012, the progress on Pascal VOC based on these traditional methods had become incremental, with minor gains from building complicated ensemble systems. This showed the limitations of these traditional detectors. Most prominently, these limitations included: (i) during proposal generation, a huge number of proposals were generated, and many of them were redundant; this resulted in a large number of false positives during classification. Moreover, window scales were designed manually and heuristically, and could not match the objects well; (ii) feature descriptors were hand-crafted based on low level visual cues [129, 141, 4], which made it difficult to capture representative semantic information in complex contexts. (iii) each step of the detection pipeline was designed and optimized separately, and thus could not obtain a global optimal solution for the whole system.

After the success of applying deep convolutional neural networks (DCNN) for image classification [96, 62], object detection also achieved remarkable progress based on deep learning techniques [162, 48]. The new deep learning based algorithms outperformed the traditional detection algorithms by huge margins. Deep convolutional neural network is a biologically-inspired structure for computing hierarchical features. An early attempt to build such a hierarchical and spatial-invariant model for image classification was “neocognitron” [39] proposed by Fukushima. However, this early attempt lacked effective optimization techniques for supervised learning. Based on this model, Lecun et al. [100] optimized a convolutional neural network by stochastic gradient descent (SGD) via back-propagation and showed competitive performance on digit recognition. After that, however, deep convolutional neural networks were not heavily explored, with support vector machines becoming more prominent. This was because deep learning had some limitations: (i) lack of large scale annotated training data, which caused overfitting; (ii) limited computation resources; and (iii) weak theoretical support compared to SVMs. In 2009, Jia et al. [27] collected a large scale annotated image dataset ImageNet which contained 1.2M high resolution images, making it possible to train deep models with large scale training data. With the development of computing resources on parallel computing systems (such as GPU clusters), in 2012 Krizhevsky et al. [96] trained a large deep convolutional model with ImageNet dataset and showed significant improvement on Large Scale Visual Recognition Challenge (ILSVRC) compared to all other approaches. After the success of applying DCNN for classification, deep learning techniques were quickly adapted to other vision tasks and showed promising results compared to the traditional methods. Figure 2.2 also illustrates the major developments and milestones of deep learning based object detection techniques after 2012.

In contrast to hand-crafted descriptors used in traditional detectors, deep convolutional neural networks generate hierarchical feature representations from raw pixels to high level semantic information, which is learned automatically from the training data and shows more discriminative expression capability in complex contexts. Furthermore, benefiting from the powerful learning capacity, a deep convolutional neural network can obtain a better feature representation with a larger dataset, while the learning capacity of traditional visual descriptors are fixed, and can not improve when more data becomes available. These properties made it possible to design object detection algorithms based on deep convolutional neural networks which could be optimized in an end-to-end manner, with more powerful feature representation capability.

## 2.2 Problem Settings

In this section, we present the formal problem setting for object detection based on deep learning. Object detection involves both recognition (e.g., "object classification") and localization (e.g., "location regression") tasks. An object detector needs to distinguish objects of certain target classes from backgrounds in the image with precise localization and correct categorical label prediction to each object instance. Bounding boxes or pixel masks are predicted to localize these target object instances.

More formally, assume we are given a collection of $N$ annotated images $\left\{x_1, x_2, ..., x_N\right\}$, and for $i^{th}$ image $x_i$, there are $M_i$ objects belonging to $C$ categories with annotations:

$$y_i = \left\{(c_1^i, b_1^i), (c_2^i, b_2^i), ..., (c_{M_i}^i, b_{M_i}^i)\right\} \tag{2.1}$$

where $c_j^i( c_j^i \in C)$ and $b_j^i$ (bounding box or pixel mask of the object) denote categorical and spatial labels of $j$. th object in $x_i$ respectively. The detector is $f$ parameterized by $\theta$. For $x_i$, the prediction $y_{\text{pred}}^i$ shares the same format as $y_i$:

$$y_{\text{pred}}^i = \left\{(c_{\text{pred}_1}^i, b_{\text{pred}_1}^i), (c_{\text{pred}_2}^i, b_{\text{pred}_2}^i), ...)\right\} \tag{2.2}$$

Finally a loss function $\ell$ is set to optimize detector as:

$$\ell(x, \theta) = \frac{1}{N}\sum_{i=1}^{N} \ell(y_{\text{pred}}^i, x_i, y_i; \theta) + \frac{\lambda}{2}\left\|\theta\right\|_2^2 \tag{2.3}$$

where the second term is a regularizer, with trade-off parameter $\lambda$. Different loss functions such as softmax loss [47] and focal loss [119] impact the final detection performance, and we will discuss these functions in Section 2.4.

At the time of evaluation, a metric called intersection-over-union (IoU) between objects and predictions is used to evaluate the quality of localization (we omit index $i$ here):

$$\mathrm{IoU}(b_{\text{pred}}, b_{\text{gt}}) = \frac{\mathrm{Area}(b_{\text{pred}} \bigcap b_{\text{gt}})}{\mathrm{Area}(b_{\text{pred}} \bigcup b_{\text{gt}})} \tag{2.4}$$

Here, $b_{\text{gt}}$ refers to the ground truth bbox or mask. An IoU threshold $\Omega$ is set to determine whether a prediction *tightly* covers the object or not (i.e. IoU $\geq \Omega$; commonly researchers set $\Omega = 0.5$). For object detection, a prediction with correct categorical label as well as successful localization prediction (meeting the IoU criteria) is considered as positive, otherwise it's a negative prediction:

$$\text{Prediction} = \begin{cases} \text{Positive} & c_{\text{pred}} = c_{\text{gt}} \text{ and } \mathrm{IoU}(b_{\text{pred}}, b_{\text{gt}}) > \Omega \\ \text{Negative} & \text{otherwise} \end{cases} \tag{2.5}$$

For generic object detection problem evaluation, mean average precision (mAP) over $C$ classes is used for evaluation, and in real world scenarios such as pedestrian detection, different evaluation metrics are used. The details of evaluation metric for different detection tasks will be discussed in Section 2.6. In addition to detection accuracy, inference speed is also an important metric to evaluate object detection algorithms. Specifically, if we wish to detect objects in a video stream (real-time detection), it is imperative to have a detector that can process this information quickly. Thus, the detector efficiency is also evaluated on Frame per second (FPS), i.e., how many images it can process per second. Commonly a detector that can achieve an inference speed of 20 FPS, is considered to be a real-time detector.

## 2.3 Detection Components

In this section, we introduce different components of object detection. The first is about the choice of object detection paradigm. We first introduce the concepts of two detection settings: bbox-level and mask-level algorithms. Then, We introduce two major object detection paradigms: two-stage detectors and one-stage detectors. Under these paradigms, detectors can use a variety of deep learning backbone architectures, proposal generators, and feature representation modules.

### 2.3.1 Detection Settings

There are two settings in object detection: (i) vanilla object detection (bbox-level localization) and (ii) instance segmentation (pixel-level or mask-level localization). Vanilla object detection has been more extensively studied and is considered as the traditional detection setting, where the goal is to localize objects by rectangle bounding boxes. In vanilla object detection algorithms, only bbox annotations are required, and in evaluation, the IoU between predicted bounding box with the ground truth is calculated to measure the performance. Instance segmentation is a relatively new setting and is based on traditional detection setting. Instance segmentation requires to segment each object by a pixel-wise mask instead of a rough rectangle bounding box. Due to more precise pixel-level prediction, instance segmentation is more sensitive to spatial misalignment, and thus has higher requirement to process the spatial information. The evaluation metric of instance segmentation is almost identical to the bbox-level detection, except that the IoU computation is performed on mask predictions. Though the two detection settings are slightly different, the main components introduced later can mostly be shared by the two settings.

### 2.3.2 Detection Paradigms

Current state-of-the-art object detectors with deep learning can be mainly divided into two major categories: two-stage detectors and one-stage detectors. For a two-stage detector, in the first stage, a sparse set of proposals is generated; and in the second stage, the feature vectors of generated proposals are encoded by deep convolutional neural networks followed by making the object class predictions. An one-stage detector does not have a separate stage for proposal generation (or learning a proposal generation). They typically consider all positions on the image as potential objects, and try to classify each region of interest as either background or a target object. Two-stage detectors often reported state-of-the-art results on many public benchmark datasets. However, they generally fall short in terms of lower inference speeds. One-stage detectors are much faster and more desired for real-time object detection applications, but have a relatively poor performance compared to the two-stage detectors.

#### Two-stage Detectors

Two-stage detectors split the detection task into two stages: (i) proposal generation; and (ii) making predictions for these proposals. During the proposal generation phase, the detector will try to identify regions in the image which may potentially be objects. The idea is to propose regions with a high recall, such that all objects in the image belong to at least one of these proposed region. In the second stage, a deep-learning based model is used to classify these proposals with the right categorical labels. The region may either be a background, or an object from one of the predefined class labels . Additionally, the model may refine the original localization suggested by the proposal generator. Next, we review some of the most influential efforts among two-stage detectors.

**R-CNN** [48] is a pioneering two-stage object detector proposed by Girshick et al. in 2014. Compared to the previous state-of-the-art methods based on a traditional detection framework SegDPM [35] with 40.4% mAP on Pascal VOC2010, R-CNN significantly improved the detection performance and obtained 53.7% mAP. The pipeline of R-CNN can be divided into three components: (i) proposal generation, (ii) feature extraction and (iii) region classification. For each image, R-CNN generates a sparse set of proposals (around 2000 proposals) via Selective Search [199], which is designed to reject regions that can easily be identified as background regions. Then, each proposal is cropped and resized into a fixed-size region and is encoded into a (e.g. 4096 dimensional) feature vector by a deep convolutional neural network, followed by a one-vs-all SVM classifier. Finally the bounding box regressors are learned using the extracted features as input in order to make the original proposals tightly bound the objects. Compared to traditional hand-crafted feature descriptors, deep neural networks generate hierarchical features and capture different scale information in different layers, and finally produce robust and discriminative features for classification. utilize the power of transfer learning, R-CNN adopts weights of convolutional networks pre-trained on ImageNet. The last fully connected layer (FC layer) is re-initialized for the detection task. The whole detector is then finetuned on the pre-trained model. This transfer of knowledge from the Imagenet dataset offers significant performance gains. In addition, R-CNN rejects huge number of easy negatives before training, which helps improve learning speed and reduce false positives.

However, R-CNN faces some critical shortcomings: (i) the features of each proposal were extracted by deep convolutional networks *separately* (i.e., computation was not shared), which led to heavily duplicated computations. Thus, R-CNN was extremely time-consuming for training and testing; (ii) the three steps of R-CNN (proposal generation, feature extraction and region classification) were independent components and the whole detection framework could not be optimized in an end-to-end manner, making it difficult to obtain global optimal solution; and (iii) Selective Search relied on low-level visual cues and thus struggled to generate high quality proposals in complex contexts. Moreover, it is unable to enjoy the benefits of GPU acceleration.

Inspired by the idea of spatial pyramid matching (SPM) [91], He et al. proposed **SPP-net** [61] to accelerate R-CNN as well as learn more discriminative features. Instead of cropping proposal regions and feeding into CNN model separately, SPP-net computes the feature map from the whole image using a deep convolutional network and extracts fixed-length feature vectors on the feature map by a Spatial Pyramid Pooling (SPP) layer. SPP partitions the feature map into an $N \times N$ grid, for multiple values of $N$ (thus allowing obtaining information at different scales), and performs pooling on each cell of the grid, to give a feature vector. The feature vectors obtained from each $N \times N$ grid are concatenated to give the representation for the region. The extracted features are fed into region SVM classifiers and bounding box regressors. In contrast to RCNN, SPP-layer can also work on images/regions at various scales and aspect ratios without resizing them. Thus, it does not suffer from information loss and unwanted geometric distortion.

SPP-net achieved better results and had a significantly faster inference speed compared to R-CNN. However, the training of SPP-net was still multi-stage and thus it could not be optimized end-to-end (and required extra cache memory to s-tore extracted features). In addition, SPP layer did not back-propagate gradients to convolutional kernels and thus all the parameters before the SPP layer were frozen. This significantly limited the learning capability of deep backbone architectures. Girshick et al. proposed **Fast R-CNN** [47], a multi-task learning detector which addressed these two limitations of SPP-net. Fast R-CNN (like SPP-Net) also computed a feature map for the whole image and extracted fixed-length region features on the feature map. Different from SPP-net, Fast R-CNN used ROI Pooling layer to extract region features. *ROI pooling* layer is a special case of SPP which on-ly takes a single scale (i.e., only one value of $N$ for the $N \times N$ grid) to partition the proposal into fixed number of divisions, and also backpropagated error signals to the convolution kernels. After feature extraction, feature vectors were fed into a sequence of fully connected layers before two sibling output layers: classification layer (cls) and regression layer (reg). Classification layer was responsible for generating softmax probabilities over C+1 classes (C classes plus one background class), while regression layer encoded 4 real-valued parameters to refine bounding boxes. In Fast RCNN, the feature extraction, region classification and bounding box regression steps can all be optimized end-to-end, without extra cache space to store features (unlike SPP Net). Fast R-CNN achieved a much better detection accuracy than R-CNN and SPP-net, and had a better training and inference speed.

Despite the progress in learning detectors, the proposal generation step still relied on traditional methods such as Selective Search [199] or Edge Boxes [256], which were based on low-level visual cues and could not be learned in a data-driven manner. To address this issue, **Faster R-CNN** [162] was developed which relied on a novel proposal generator: Region Proposal Network (RPN). This proposal generator could be learned via supervised learning methods. RPN is a fully convolutional network which takes an image of arbitrary size and generates a set of object proposals on each position of the feature map. The network slid over the feature map using an $n \times n$ sliding window, and generated a feature vector for each position. The feature vector was then fed into two sibling output branches, object classification layer (which classified whether the proposal was an object or not) and bounding box regression layer. These results were then fed into the final layer for the actual object classification and bounding box localization. RPN could be inserted into Fast R-CNN and thus the whole framework could be optimized in an end-to-end manner on training data. This way RPN enabled proposal generation in a data driven manner, and was also able to enjoy the discriminative power of deep backbone networks. Faster R-CNN was able to make predictions at 5FPS on GPU and achieved state-of-the-art results on many public benchmark datasets, such as Pascal VOC 2007, 2012 and MSCOCO. Currently, there are huge number of detector variants based on Faster R-CNN for different usage [10, 118, 95, 5].

Faster R-CNN computed feature map of the input image and extracted region features on the feature map, which shared feature extraction computation across different regions. However, the computation was not shared in the region classification step, where each feature vector still needed to go through a sequence of FC layers separately. Such extra computation could be extremely large as each image may have hundreds of proposals. Simply removing the fully connected layers would result in the drastic decline of detection performance, as the deep network would have reduced the spatial information of proposals. Dai et al. [24] proposed Region-based Fully Convolutional Networks (**R-FCN**) which shared the computation cost in the region classification step. R-FCN generated a Position Sensitive Score Map which encoded relative position information of different classes, and used a Position Sensitive ROI Pooling layer (PSROI Pooling) to extract spatial-aware region features by encoding each relative position of the target regions. The extracted feature vectors maintained spatial information and thus the detector achieved competitive results compared to Faster R-CNN without region-wise fully connected layer operations.

Another issue with Faster R-CNN was that it used a single deep layer feature map to make the final prediction. This made it difficult to detect objects at different scales. In particular, it was difficult to detect small objects. In DCNN feature representations, deep layer features are semantically-strong but spatially-weak, while shallow layer features are semantically-weak but spatially-strong. Lin et al. [118] exploited this property and proposed Feature Pyramid Networks (**FPN**) which combined deep layer features with shallow layer features to enable object detection in feature maps at different scales. The main idea was to strengthen the spatially strong shallow layer features with rich semantic information from the deeper layers. FPN achieved significant progress in detecting multi-scale objects and has been widely used in many other domains such as video detection [85, 57] and human pose recognition [156, 146].

Most instance segmentation algorithms are extended from vanilla object detection algorithms. Early methods [149, 148, 23] commonly generated segment proposals, followed by Fast RCNN for segments classification. Later, Dai et al. [23] proposed a multi-stage algorithm named "MNC" which divided the whole detection framework into multiple stages and predicted segmentation masks from the learned bounding box proposals, which were later categorized by region classifiers. These early works performed bbox and mask prediction in multiple stages. To make the whole process more flexible, He et al. [60] proposed **Mask R-CNN**, which predicted bounding boxes and segmentation masks in parallel based on the proposals and reported state-of-the-art results. Based on Mask R-CNN, Huang et al. [75] proposed a mask-quality aware framework, named Mask Scoring R-CNN, which learned the quality of the predicted masks and calibrated the misalignment between mask quality and mask confidence score.

Figure 2.3 gives an overview of the detection frameworks for several representative two-stage detectors.

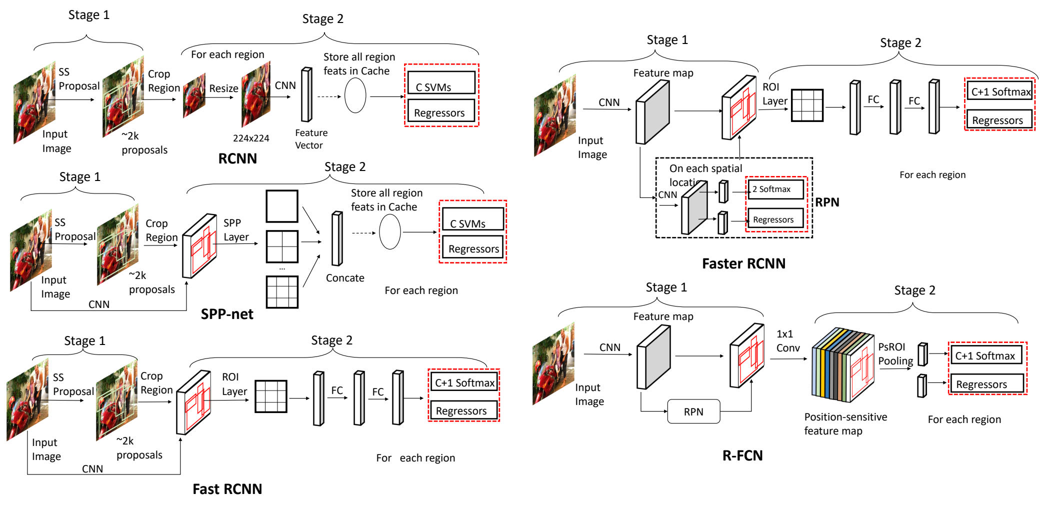


Figure 2.3: Overview of different two-stage detection frameworks for generic object detection. Red dotted rectangles denote the outputs that define the loss functions.

### One-stage Detectors

Different from two-stage detection algorithms which divide the detection pipeline into two parts: proposal generation and region classification; one-stage detectors do not have a separate stage for proposal generation (or learning a proposal generation). They typically consider all positions on the image as potential objects, and try to classify each region of interest as either background or a target object.

One of the early successful one-stage detectors based on deep learning was developed by Sermanet et al. [171] named **OverFeat**. OverFeat performed object detection by casting DCNN classifier into a fully convolutional object detector. Object detection can be viewed as a "multi-region classification" problem, and thus OverFeat extended the original classifier into detector by viewing the last FC layers as 1x1 convolutional layers to allow arbitrary input. The classification network output a grid of predictions on each region of the input to indicate the presence of an object. After identifying the objects, bounding box regressors were learned to refine the predicted regions based on the same DCNN features of classifier. In order to detect multi-scale objects, the input image was resized into multiple scales which were fed into the network. Finally, the predictions across all the scales were merged together. OverFeat showed significant speed strength compared with RCNN by sharing the computation of overlapping regions using convolutional layers, and only a single pass forward through the network was required. However, the training of classifiers and regressors were separated without being jointly optimized.

Later, Redmon et al. [157] developed a real-time detector called **YOLO** (You Only Look Once). YOLO considered object detection as a regression problem and spatially divided the whole image into fixed number of grid cells (e.g. using a $7 \times 7$ grid). Each cell was considered as a proposal to detect the presence of one or more objects. In the original implementation, each cell was considered to contain the center of (upto) two objects. For each cell, a prediction was made which comprised the following information: whether that location had an object, the bounding box coordinates and size (width and height), and the class of the object. The whole framework was a single network and it omitted proposal generation step which could be optimized in an end-to-end manner. Based on a carefully designed lightweight architecture, YOLO could make prediction at 45 FPS, and reach 155 FPS with a more simplified backbone. However, YOLO faced some challenges: (i) it could detect upto only two objects at a given location, which made it difficult to detect small objects and crowded objects [157]. (ii) only the last feature map was used for prediction, which was not suitable for predicting objects at multiple scales and aspect ratios.

In 2016, Liu et al. proposed another one-stage detector Single-Shot Mulibox Detector (**SSD**) [123] which addressed the limitations of YOLO. SSD also divided images into grid cells, but in each grid cell, a set of anchors with multiple scales and aspect-ratios were generated to discretize the output space of bounding boxes (unlike predicting from fixed grid cells adopted in YOLO). Each anchor was refined by 4-value offsets learned by the regressors and was assigned (C+1) categorical probabilities by the classifiers. In addition, SSD predicted objects on multiple feature maps, and each of these feature maps was responsible for detecting a certain scale of objects according to its receptive fields. In order to detect large objects and increase receptive fields, several extra convolutional feature maps were added to the original backbone architecture. The whole network was optimized with a weighted sum of localization loss and classification loss over all prediction maps via an end-to-end training scheme. The final prediction was made by merging all detection results from different feature maps. In order to avoid huge number of negative proposals dominating training gradients, hard negative mining was used to train the detector. Intensive data augmentation was also applied to improve detection accuracy. SSD achieved comparable detection accuracy with Faster R-CNN but enjoyed the ability to do real-time inference.

Without proposal generation to filter easy negative samples, the class imbalance between foreground and background is a severe problem in one-stage detector. Lin et al. [119] proposed a one-stage detector **RetinaNet** which addressed class imbalance problem in a more flexible manner. RetinaNet used focal loss which suppressed the gradients of easy negative samples instead of simply discarding them. Further, they used feature pyramid networks to detect multi-scale objects at different levels of feature maps. Their proposed focal loss outperformed naive hard negative mining strategy by large margins.

Redmon et al. proposed an improved YOLO version, **YOLOv2** [159] which significantly improved detection performance but still maintained real-time inference speed. YOLOv2 adopted a more powerful deep convolutional backbone architecture which was pre-trained on higher resolution images from ImageNet (from $224 \times 224$ to $448 \times 448$), and thus the weights learned were more sensitive to capturing fine-grained information. In addition, inspired by the anchor strategy used in SSD, YOLOv2 defined better anchor priors by k-means clustering from the training data (instead of setting manually). This helped in reducing optimizing difficulties in localization. Finally integrating with Batch Normalization layers [76] and multi-scale training techniques, YOLOv2 achieved state-of-the-art detection results at that time.

The previous approaches required designing anchor boxes manually to train a detector. Later a series of anchor-free object detectors were developed, where the goal was to predict keypoints of the bounding box, instead of trying to fit an object to an anchor. Law and Deng proposed a novel anchor-free framework **CornerNet** [98] which detected objects as a pair of corners. On each position of the feature map, class heatmaps, pair embeddings and corner offsets were predicted. Class heatmaps calculated the probabilities of being corners, and corner offsets were used to regress the corner location. And the pair embeddings served to group a pair of corners which belong to the same objects. Without relying on manually designed anchors to match objects, CornerNet obtained significant improvement on MSCOCO datasets. Later there were several other variants of keypoint detection based one-stage detectors [243, 29].

Figures 2.4 gives an overview of different detection frameworks for several representative one-stage detectors.

### 2.3.3 Backbone Architecture

R-CNN [48] showed adopting convolutional weights from models pre-trained on large scale image classification problem could provide richer semantic information

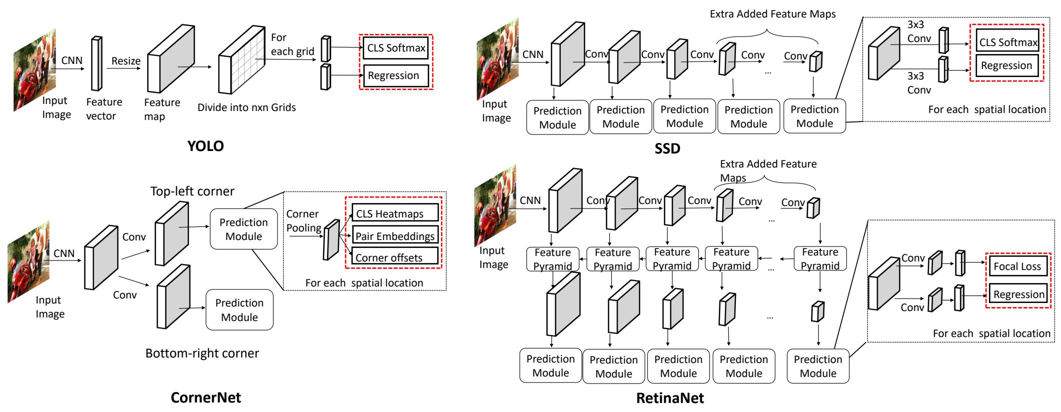


Figure 2.4: Overview of different one-stage detection frameworks for generic object detection. Red rectangles denotes the outputs that define the objective functions.

to train detectors and enhanced the detection performance. During the later years, this approach had become the default strategy for most object detectors. In this section, we will first briefly introduce the basic concept of deep convolutional neural networks and then review some architectures which are widely used for detection.

### Basic Architecture of A CNN

Deep convolutional neural network (DCNN) is a typical deep neural network and has proven extremely effective in visual understanding [100, 96]. Deep convolutional neural networks are commonly composed of a sequence of convolutional layers, pooling layers, nonlinear activation layers and fully connected layers (FC layers). Convolutional layer takes an image input and convolves over it by $n \times n$ kernels to generate a feature map. The generated feature map can be regarded as a multi-channel image and each channel represents different information about the image. Each pixel in the feature map (named neuron) is connected to a small portion of adjacent neurons from the previous map, which is called the receptive field. After generating feature maps, a non-linear activation layer is applied. Pooling layers are used to summarize the signals within the receptive fields, to enlarge receptive fields as well as reduce computation cost, .

With the combination of a sequence of convolutional layers, pooling layers and non-linear activation layers, the deep convolutional neural network is built. The whole network can be optimized via a defined loss function by gradient-based optimization method (stochastic gradient descent [164], Adam [90], etc.). A typical convolutional neural network is AlexNet [96], which contains five convolutional layers, three max-pooling layers and three fully connected layers. Each convolutional layer is followed by a ReLU [137] non-linear activation layer.

**CNN Backbone for Object Detection**

In this section, we will review some architectures which are widely used in object detection tasks with state-of-the-art results, such as VGG16 [162, 47], ResNet [62, 24], ResNeXt [119] and Hourglass [98].

VGG16 [181] was developed based on AlexNet. VGG16 is composed of five groups of convolutional layers and three FC layers. There are two convolutional layers in the first two groups and three convolutional layers in the next three groups. Between each group, a Max Pooling layer is applied to decrease spatial dimension. VGG16 showed that increasing depth of networks by stacking convolutional layers could increase the model's expression capability, and led to a better performance. However, increasing model depth to 20 layers by simply stacking convolutional layers led to optimization challenges with SGD. The performance declined significantly and was inferior to shallower models, even during the training stages. Based on this observation, He et al. [62] proposed ResNet which reduced optimization difficulties by introducing shortcut connections. Here, a layer could skip the nonlinear transformation and directly pass the values to the next layer as is (thus giving us an implicit identity layer). This is given as:

$$x_{l+1} = x_l + f_{l+1}(x_l, \theta) \tag{2.6}$$

where $x_l$ is the input feature in $l$-th layer and $f_{l+1}$ denotes operations on input $x_l$ such as convolution, normalization or non-linear activation. $f_{l+1}(x_l, \theta)$ is the residual function to $x_l$, so the feature map of any deep layer can be viewed as the sum of the activation of shallow layer and the residual function. Shortcut connection creates a highway which directly propagates the gradients from deep layers to shallow units and thus, significantly reduces training difficulty. With residual blocks effectively training networks, the model depth could be increased (e.g. from 16 to 152), allowing us to train very high capacity models. Later, He et al. [63] proposed a pre-activation variant of ResNet, named ResNet-v2. Their experiments showed appropriate ordering of the Batch Normalization [76] could further perform better than original ResNet. This simple but effective modification of ResNet made it possible to successfully train a network with more than 1000 layers, and still enjoyed improved performance due to the increase in depth. Huang et al. argued that although ResNet reduced the training difficulty via shortcut connection, it did not fully utilize features from previous layers. The original features in shallow layers were missing in element-wise operation and thus could not be directly used later. They proposed DenseNet [73], which retained the shallow layer features, and improved information flow, by concatenating the input with the residual output instead of element-wise addition:

$$x_{l+1} = x_l \circ f_{l+1}(x_l, \theta) \tag{2.7}$$

where $\circ$ denotes concatenation. Chen [16] et al. argued that in DenseNet, the majority of new exploited features from shallow layers were duplicated and incurred high computation cost. Integrating the advantages of both ResNet and DenseNet, they propose a Dual Path Network (DPN) which divides $x_l$ channels into two parts: $x_l^d$ and $x_l^r$. $x_l^d$ was used for dense connection computation and $x_l^r$ was used for element-wise summation, with unshared residual learning branch $f_{l+1}^d$ and $f_{l+1}^r$. The final result was the concatenated output of the two branches:

$$x_{l+1} = (x_l^r + f_{l+1}^r(x_l^r, \theta^r)) \circ (x_l^d \circ f_{l+1}^d(x_l^d, \theta^d)) \tag{2.8}$$

Based on ResNet, Xie et al. [215] proposed ResNeXt which considerably reduced computation and memory cost while maintaining comparable classification accuracy. ResNeXt adopted group convolution layers [96] which sparsely connects feature map channels to reduce computation cost. By increasing group number to keep computation cost consistent to the original ResNet, ResNeXt captures richer semantic feature representation from the training data and thus improves backbone accuracy. Later, Howard et al. [70] set the coordinates equal to number of channels of each feature map and developed MobileNet. MobileNet significantly reduced computation cost as well as number of parameters without significant loss in classification accuracy. This model was specifically designed for usage on a mobile platform.

In addition to increasing model depth, some efforts explored benefits from increasing model width to improve the learning capacity. Szegedy et al. proposed GoogleNet with an inception module [192] which applied different scale convolution kernels ($1 \times 1, 3 \times 3$ and $5 \times 5$) on the same feature map in a given layer. This way it captured multi-scale features and summarized these features together as an output feature map. Better versions of this model were developed later with different design of choice of convolution kernels [193], and introducing residual blocks [191].

The network structures introduced above were all designed for image classification. Typically these models trained on ImageNet are adopted as initialization of the model used for object detection. However, directly applying this pre-trained model from classification to detection is sub-optimal due to a potential conflict between classification and detection tasks. Specifically, (i) classification requires large receptive fields and wants to maintain spatial invariance. Thus multiple downsampling operation (such as pooling layer) are applied to decrease feature map resolution. The feature maps generated are low-resolution and spatially invariant and have large receptive fields. However, in detection, high-resolution spatial information is required to correctly localize objects; and (ii) classification makes predictions on a single feature map, while detection requires feature maps with multiple representations to detect objects at multiple scales. To bridge the difficulties between the two tasks, Li et al. introduced DetNet [113] which was designed specifically for detection. DetNet kept high resolution feature maps for prediction with dilated convolutions to increase receptive fields. In addition, DetNet detected objects on multi-scale feature maps, which provided richer information. DetNet was pre-trained on large scale classification dataset while the network structure was designed for detection.

Hourglass Network [139] is another architecture, which was not designed specifically for image classification. Hourglass Network first appeared in human pose recognition task [139], and was a fully convolutional structure with a sequence of hourglass modules. Hourglass module first downsampled the input image via a sequence of convolutional layer or pooling layer, and upsampled the feature map via deconvolutional operation. To avoid information loss in downsampling stage, skip connection were used between downsampling and upsampling features. Hourglass module could capture both local and global information and thus was very suitable for object detection. Currently Hourglass Network is widely used in state-of-the-art detection frameworks [98, 243, 29].

### 2.3.4 Proposal Generation

Proposal generation plays a very important role in the object detection framework. A proposal generator generates a set of rectangle bounding boxes, which are potentially objects. These proposals are then used for classification and localization refinement. We categorize proposal generation methods into four categories: traditional computer vision methods, anchor-based supervised learning methods, keypoint based methods and other methods. Notably, both one-stage detectors and two-stage detectors generate proposals, the main difference is two-stage detectors generates a sparse set of proposals with only foreground or background information, while one-stage detectors consider each region in the image as a potential proposal, and accordingly estimates the class and bounding box coordinates of potential objects at each location.

### Traditional Computer Vision Methods

These methods generate proposals in images using traditional computer vision methods based on low-level cues, such as edges, corners, color, etc. These techniques can be categorized into three principles: (i) computing the 'objectness score' of a candidate box; (ii) merging super-pixels from original images; (iii) generating multiple foreground and background segments;

*Objectness score* based methods predict an objectness score of each candidate box measuring how likely it may contain an object. Arbelaez et al. [1] assigned objectness score to proposals by classification based on visual cues such as color contrast, edge density and saliency. Rahtu et al.[153] revisited the idea of Arbelaez et al. [1] and introduced a more efficient cascaded learning method to rank the objectness score of candidate proposals.

*Superpixels merging* is based on merging superpixels generated from segmentation results. Selective Search [199] was a proposal generation algorithm based on merging super-pixels. It computed the multiple hierarchical segments generated by segmentation method [34], which were merged according to their visual factors (color, areas, etc.), and finally bounding boxes were placed on the merged segments. Manen et al. [134] proposed a similar idea to merge superpixels. The difference was that the weight of the merging function was learned and the merging process was randomized. Selective Search is widely used in many detection frameworks due to its efficiency and high recall compared to other traditional methods.

*Seed segmentation* starts with multiple seed regions, and for each seed, foreground and background segments are generated. To avoid building up hierarchical segmentation, CPMC [11] generated a set of overlapping segments initialized with diverse seeds. Each proposal segment was the solution of a binary (foreground or background) segmentation problem. Enreds and Hoiem [30] combined the idea of

Selective Search [199] and CPMC [11]. It started with super-pixels and merged them with new designed features. These merged segments were used as seeds to generate larger segments, which was similar to CPMC. However, producing high quality segmentation masks is very time-consuming and it's not applicable to large scale datasets.

The primary advantage of these traditional computer vision methods is that they are very simple and can generate proposals with high recall (e.g. on medium scale datasets such as Pascal VOC). However, these methods are mainly based on low level visual cues such as color or edges. They cannot be jointly optimized with the whole detection pipeline. Thus they are unable to exploit the power of large scale datasets to improve representation learning. On challenging datasets such as MSCOCO [120], traditional computer vision methods struggled to generate high quality proposals due to these limitations.

### Anchor-based Methods

One large family of supervised proposal generators is anchor-based methods. They generate proposals based on pre-defined anchors. Ren et al. proposed Region Proposal Network (RPN) [162] to generate proposals in a supervised way based on deep convolutional feature maps. The network slid over the entire feature map using $3 \times 3$ convolution filters. For each position, $k$ anchors (or initial estimates of bounding boxes) of varying size and aspect ratios were considered. These sizes and ratios allowed for matching objects at different scales in the entire image. Based on the ground truth bonding boxes, the object locations were matched with the most appropriate anchors to obtain the supervision signal for the anchor estimation. A $256-$dimensional feature vector was extracted from each anchor and was fed into t-wo sibling branches - classification layer and regression layer. Classification branch was responsible for modeling objectness score while regression branch encoded four real-values to refine location of the bounding box from the original anchor estimation. Based on the ground truth, each anchor was predicted to either be an object, or just background by the classification branch (See Fig. 2.5). Later, SS-D [123] adopted a similar idea of anchors in RPN by using multi-scale anchors to match objects. The main difference was that SSD assigned categorical probabilities to each anchor proposal, while RPN first evaluated whether the anchor proposal was foreground or background and performed the categorical classification in the next stage.

Despite promising performance, the anchor priors are manually designed with multiple scales and aspect ratios in a heuristic manner. These design choices may not be optimal, and different datasets would require different anchor design strategies. Many efforts have been made to improve the design choice of anchors. Zhang et al. proposed Single Shot Scale-invariant Face Detector (S3FD) [237] based on SSD with carefully designed anchors to match the objects. According to the effective receptive field [132] of different feature maps, different anchor priors were designed. Zhu et al. [248] introduced an anchor design method for matching small objects by enlarging input image size and reducing anchor strides. Xie et al. proposed Dimension-Decomposition Region Proposal Network (DeRPN) [102] which decomposed the dimension of anchor boxes based on RPN. DeRPN used an anchor string mechanism to independently match objects width and height. This helped match objects with large scale variance and reduced the searching space.

Ghodrati et al. developed DeepProposals [44] which predicted proposals on the low-resolution deeper layer feature map. These were then projected back onto the high-resolution shallow layer feature maps, where they are further refined. Redmon et al. [159] designed anchor priors by learning priors from the training data using k-means clustering. Later, Zhang et al. introduced Single-Shot Refinement Neural Network (RefineDet) [236] which refined the manually defined anchors in two steps. In the first step, RefineDet learned a set of localization offsets based on the original hand-designed anchors and these anchors were refined by the learned offsets. In the second stage, a new set of localization offsets were learned based on the refined anchors from the first step for further refinement. This cascaded optimization framework significantly improved the anchor quality and final prediction accuracy in a data-driven manner. Cai et al. proposed Cascade R-CNN [10] which adopted a similar idea as RefineDet by refining proposals in a cascaded way. Yang et al. [224] modeled anchors as functions implemented by neural networks which was computed from customized anchors. Their method MetaAnchor showed comprehensive improvement compared to other manually defined methods but the customized anchors were still designed manually.

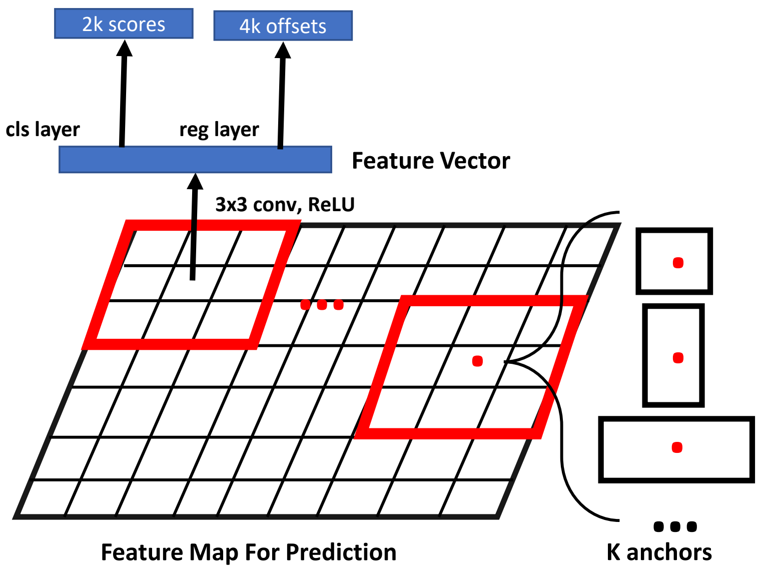


Figure 2.5: Diagram of RPN [162]. Each position of the feature map connects with a sliding windows, followed with two sibling branches.

### Keypoints-based Methods

Another proposal generation approach is based on keypoint detection, which can be divided into two families: corner-based methods and center-based methods. *Corner-based methods* predict bounding boxes by merging pairs of corners learned from the feature map. Denet [197] reformulated the object detection problem in a probabilistic way. For each point on the feature map, Denet modeled the distribution of being one of the 4 corner types of objects (top-left, top-right, bottom-left, bottom-right), and applied a naive bayesian classifiers over each corner of the objects to estimate the confidence score of a bounding box. This corner-based algorithm eliminated the design of anchors and became a more effective method to produce high quality proposals. Later based on Denet, Law and Deng proposed CornerNet [98] which directly modeled categorical information on corners. CornerNet modeled information of top-left and bottom-right corners with novel feature embedding methods and corner pooling layer to correctly match keypoints belonging to the same objects, obtaining state-of-the-art results on public benchmarks. For *center-based methods*, the probability of being the center of the objects is predicted on each position of the feature map, and the height and width are directly regressed without any anchor priors. Zhu et al. [247] presented a feature-selection-anchor-free (FSAF) framework which could be plugged into one-stage detectors with FPN structure. In FSAF, an online feature selection block is applied to train multi-level center-based branches attached in each level of the feature pyramid. During training, FSAF dynamically assigned each object to the most suitable feature level to train the center-based branch. Similar to FSAF, Zhou et al. proposed a new center-based framework [243] based on a single Hourglass network [98] without FPN structure. Furthermore, they applied center-based method into higher-level problems such as 3D-detection and human pose recognition, and all achieved state-of-the-art results. Duan et al. [29] proposed CenterNet, which combined the idea of center-based methods and corner-based methods. CenterNet first predicted bounding boxes by pairs of corners, and then predicted center probabilities of the initial prediction to reject easy negatives. CenterNet obtained significant improvements compared with baselines. These anchor-free methods form a promising research direction in the future.

### Other Methods

There are some other proposal generation algorithms which are not based on keypoints or anchors but also offer competitive performances. Lu et al. proposed

AZnet [131] which automatically focused on regions of high interest. AZnet adopted a search strategy that adaptively directed computation resources to sub-regions which were likely contain objects. For each region, AZnet predicted two values: zoom indicator and adjacency scores. Zoom indicator determined whether to further divide this region which may contain smaller objects and adjacency scores denoted its objectness. The starting point was the entire image and each divided sub-region is recursively processed in this way until the zoom indicator is too small. AZnet was better at matching sparse and small objects compared to RPN's anchor-object matching approach.

### 2.3.5 Feature Representation Learning

Feature Representation Learning is a critical component in the whole detection framework. Target objects lie in complex environments and have large variance in scale and aspect ratios. There is a need to train a robust and discriminative feature embedding of objects to obtain a good detection performance. In this section, we introduce feature representation learning strategies for object detection. Specifically, we identify three categories: multi-scale feature learning, contextual reasoning, and deformable feature learning.

#### Multi-scale Feature Learning

Typical object detection algorithms based on deep convolutional networks such as Fast R-CNN [47] and Faster R-CNN [162] use only a single layer's feature map to detect objects. However, detecting objects across large range of scales and aspect ratios is quite challenging on a single feature map. Deep convolutional networks learn hierarchical features in different layers which capture different scale information. Specifically, shallow layer features with spatial-rich information have higher resolution and smaller receptive fields and thus are more suitable for detecting small objects, while semantic-rich features in deep layers are more robust to illumination, translation and have larger receptive fields (but coarse resolutions), and are more suitable for detecting large objects. When detecting small objects, high resolution representations are required and the representation of these objects may not even be available in the deep layer features, making small object detection difficult. Some techniques such as dilated/atrous convolutions [25, 24] were proposed to avoid downsampling, and used the high resolution information even in the deeper layers. At the same time, detecting large objects in shallow layers are also non-optimal without large enough receptive fields. Thus, handling feature scale issues has become a fundamental research problem within object detection. There are four main paradigms addressing multi-scale feature learning problem: Image Pyramid, Prediction Pyramid, Integrated Features and Feature Pyramid. These are briefly illustrated in the Fig. 2.6.

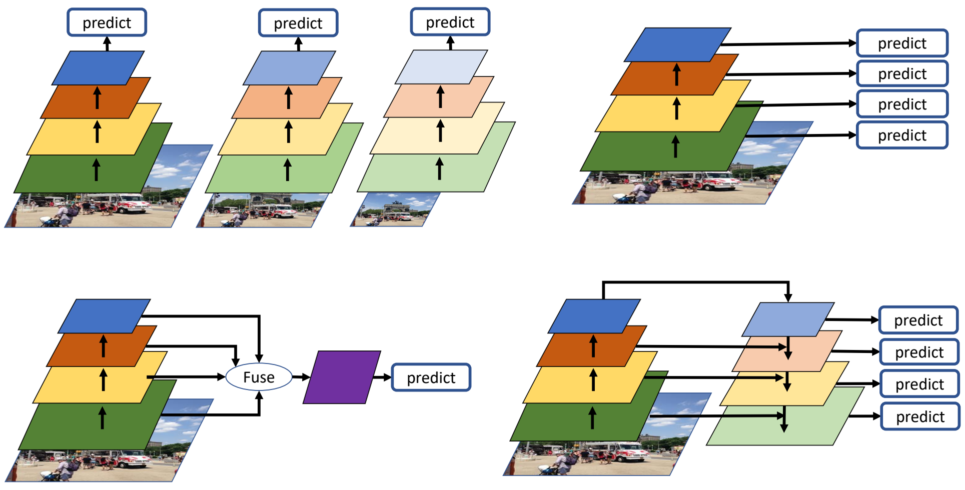


Figure 2.6: Four paradigms for multi-scale feature learning. Top Left: *Image Pyramid*, which learns multiple detectors from different scale images; Top Right: *Prediction Pyramid*, which predicts on multiple feature maps; Bottom Left: *Integrated Features*, which predicts on single feature map generated from multiple features; Bottom Right: *Feature Pyramid* which combines the structure of *Prediction Pyramid* and *Integrated Features*.

*Image pyramid*: An intuitive idea is to resize input images into a number of different scales (Image Pyramid) and to train multiple detectors, each of which is responsible for a certain range of scales [182, 72, 220, 126]. During testing, images are resized to different scales followed by multiple detectors and the detection

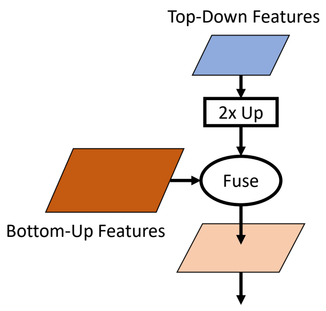


Figure 2.7: General framework for feature combination. Top-down features are 2 times up-sampled and fuse with bottom-up features. The fuse methods can be element-wise sum, multiplication, concatenation and so on. Convolution and normalization layers can be inserted in to this general framework to enhance semantic information and reduce memory cost.

results are merged. This can be computationally expensive. Liu et al. [126] first learned a light-weight scale-aware network to resize images such that all objects were in a similar scale. This was followed by learning a single scale detector. Singh et. al. [182] conducted comprehensive experiments on small object detection. They argued that learning a single scale-robust detector to handle all scale objects was much more difficult than learning scale-dependent detectors with image pyramids. In their work, they proposed a novel framework Scale Normalization for Image Pyramids (SNIP) [182] which trained multiple scale-dependent detectors and each of them was responsible for a certain scale objects.

*Integrated features*: Another approach is to construct a single feature map by combining features in multiple layers and making final predictions based on the new constructed map [180, 5, 95, 205, 88, 167]. By fusing spatially rich shallow layer features and semantic-rich deep layer features, the new constructed features contain rich information and thus can detect objects at different scales. These combinations are commonly achieved by using skip connections [62]. Feature normalization is required as feature norms of different layers have a high variance. Bell et al. proposed Inside-Outside Network (ION) [5] which cropped region features from different layers via ROI Pooling [47], and combined these multi-scale region features for the final prediction. Kong et. al. proposed HyperNet [95] which adopted a similar idea as IoN. They carefully designed high resolution hyper feature maps by integrating intermediate and shallow layer features to generate proposals and detect objects. Deconvolutional layers were used to up-sample deep layer feature maps and batch normalization layers were used to normalize input blobs in their work. The constructed hyper feature maps could also implicitly encode contextual information from different layers. Inspired by fine-grained classification algorithms which integrate high-order representation instead of exploiting simple first-order representations of object proposals, Wang et al. proposed a novel framework Multi-scale Location-aware Kernel Representation (MLKP) [205] which captured high-order statistics of proposal features and generated more discriminative feature representations efficiently. The combined feature representation was more descriptive and provides both semantic and spatial information for both classification and localization.

*Prediction pyramid*: Liu et al.'s SSD [123] combined coarse and fine features from multiple layers together. In SSD, predictions were made from multiple layers, where each layer was responsible for a certain scale of objects. Later, many efforts [9, 174, 122] followed this principle to detect multi-scale objects. Yang et al. [220] also exploited appropriate feature maps to generate certain scale of object proposals and these feature maps were fed into multiple scale-dependent classifiers to predict objects. In their work, cascaded rejection classifiers were learned to reject easy background proposals in early stages to accelerate detection speed. Multi-scale Deep Convolutional Neural Network (MSCNN) [9] applied deconvolutional layers on multiple feature maps to improve their resolutions, and later these refined feature maps were used to make predictions. Liu et al. proposed a Receptive Field Block Net (RFBNet) [122] to enhance the robustness and receptive fields via a receptive field block (RFB block). RFB block adopted similar ideas as the inception module [192] which captured features from multiple scale and receptive fields via multiple branches with different convolution kernels and finally merged them together.

*Feature pyramid*: To combine the advantage of Integrated Features and Prediction Pyramid, Lin et al. proposed Feature Pyramid Network (FPN) [118] which integrated different scale features with lateral connections in a top-down fashion to build a set of scale invariant feature maps, and multiple scale-dependent classifiers were learned on these feature pyramids. Specifically, the deep semantic-rich features were used to strengthen the shallow spatially-rich features. These top-down and lateral features were combined by element-wise summation or concatenation, with small convolutions reducing the dimensions. FPN showed significant improvement in object detection, as well as other applications, and achieved state-of-the art results in learning multi-scale features. Many variants of FPN were later developed [160, 78, 160, 242, 38, 236, 212, 104, 92, 240, 114, 101, 22], with modifications to the feature pyramid block (see Fig. 2.7). Kong et al. [94] and Zhang et. al. [236] built scale invariant feature maps with lateral connections. Different from FPN which generated region proposals followed by categorical classifiers, their methods omitted proposal generation and thus were more efficient than original FPN. Ren et al. [160] and Jeong et al. [78] developed a novel structure which gradually and selectively encoded contextual information between different layer features. Inspired by super resolution tasks [117, 176], Zhou et al. [242] developed high resolution feature maps using a novel transform block which explicitly explored the inter-scale consistency nature across multiple detection scales.

### Region Feature Encoding

For two-stage detectors, region feature encoding is a critical step to extract features from proposals into fixed length feature vectors. In R-CNN, Girshick et al. [48] cropped region proposals from the whole image and resized the cropped regions into fixed sized patches ($224 \times 224$) via bilinear interpolation, followed by a deep convolution feature extractor. Their method encoded high resolution region features but the computation was expensive.

Later Girshick et al. [47] and Ren [162] proposed ROI Pooling layer to encode region features. ROI Pooling divided each region into $n \times n$ cells (e.g. $7 \times 7$ by default) and only the neuron with the maximum signal would go ahead in the feedforward stage. This is similar to max-pooling, but across (potentially) different sized regions. ROI Pooling extracted features from the down-sampled feature map and as a result struggled to handle small objects. Dai [23] proposed ROI Warping layer which encoded region features via bilinear interpolation. Due to the down-sampling operation in DCNN, there can be a misalignment of the object position in the original image and the downsampled feature maps, which RoI Pooling and RoI Warping layers are not able to handle. Instead of quantizing grids border as ROI Warping and ROI Pooling do, He et al. [60] proposed ROI Align layer which addressed the quantization issue by bilinear interpolation at fractionally sampled positions within each grid. Based on ROI Align, Jiang et al. [80] presented Precise ROI Pooing (PrROI Pooling), which avoided any quantization of coordinates and had a continuous gradient on bounding box coordinates.

In order to enhance spatial information of the downsampled region features, Dai et al. [24] proposed Position Sensitive ROI Pooing (PSROI Pooling) which kept relative spatial information of downsampled features. Each channel of generated region feature map only corresponded to a subset channels of input region according to its relative spatial position. Based on PSROI Pooling, Zhai et al. [230] presented feature selective networks to learn robust region features by exploiting disparities among sub-region and aspect ratios. The proposed network encoded sub-region and aspect ratio information which were selectively pooled to refine initial region features by a light-weight head.

Later, more algorithms were proposed to well encode region features from different viewpoints. Zhu et al. proposed CoupleNet [254] which extracted region features by combining outputs generated from both ROI Pooling layer and PSROI Pooling layer. ROI Pooling layer extracted global region information but struggled for objects with high occlusion while PSROI Pooling layer focused more on local information. CoupleNet enhanced features generated from ROI Pooling and PSROI

Pooling by element-wise summation and generated more powerful features. Later Dai et al. proposed Deformable ROI Pooling [25] which generalized aligned RoI pooling by learning an offset for each grid and adding it to the grid center. The sub-grid start with a regular ROI Pooling layer to extract initial region features and the extracted features were used to regress offset by an auxiliary network. Deformable ROI Pooling can automatically model the image content without being constrained by fixed receptive fields.

### Contextual Reasoning

Contextual information plays an important role in object detection. Objects often tend to appear in specific environments and sometimes also coexist with other objects. For each example, birds commonly fly in the sky. Effectively using contextual information can help improve detection performance, especially for detecting objects with insufficient cues (small object, occlusion etc.) Learning the relationship between objects with their surrounding context can improve detector's ability to understand the scenario. For traditional object detection algorithms, there have been several efforts exploring context [40], but for object detection based on deep learning, context has not been extensively explored. This is because convolutional networks implicitly already capture contextual information from hierarchical feature representations. However, some recent efforts [62, 144, 21, 253, 60, 23, 60, 15, 45, 9] still try to exploit contextual information. Some works [19] have even shown that in some cases context information may even harm the detection performance. In this section we review contextual reasoning for object detection from two aspects: *global context* and *region context*.

*Global context reasoning* refers to learning from the context in the whole image. Unlike traditional detectors which attempt to classify specific regions in the image as objects, the idea here is to use the contextual information (i.e., information from the rest of the image) to classify a particular region of interest. For example, detecting a baseball ball from an image can be challenging for a traditional detector (as it may be confused with balls from other sports); but if the contextual information from the rest of the image is used (e.g. baseball field, players, bat), it becomes easier to identify the baseball ball object.

Some representative efforts include ION [5], DeepId [144] and improved version of Faster R-CNN [62]. In ION, Bell et al. used recurrent neural network to encode contextual information across the whole image from four directions. Ouyang et al. [144] learned a categorical score for each image which is used as contextual features concatenated with the object detection results. He et al. [62] extracted feature embedding of the entire image and concatenate it with region features to improve detection results. In addition, some methods [253, 60, 23, 241, 239, 178, 103] exploit global contextual information via semantic segmentation. Due to precise pixel-level annotation, segmentation feature maps capture strong spatial information. He et al. [60] and Dai et al. [23] learn unified instance segmentation framework and optimize the detector with pixel-level supervision. They jointly optimized detection and segmentation objectives as a multi-task optimization. Though segmentation can significantly improve detection performance, obtaining the pixel-level annotation is very expensive. Zhao et al. [241] optimized detectors with pseudo segmentation annotation and showed promising results. Zhang et al.'s work Detection with Enriched Semantics (DES) [239], introduced contextual information by learning a segmentation mask without segemtation annotations. It also jointly optimized object detection and segmentation objectives and enriched original feature map with a more discriminative feature map.

*Region context reasoning* encodes contextual information surrounding regions and learns interactions between the objects with their surrounding area. Directly modeling different locations and categories objects relations with the contextual is very challenging. Chen et al. proposed Spatial Memory Network (SMN) [15] which introduced a spatial memory based module. The spatial memory module captured instance-level contexts by assembling object instances back into a pseudo "image" representations which were later used for object relations reasoning. Liu et al. proposed Structure Inference Net (SIN) [127] which formulated object detection as a graph inference problem by considering scene contextual information and object relationships. In SIN, each object was treated as a graph node and the relationship between different objects were regarded as graph edges. Hu et al. [71] proposed a lightweight framework relation network which formulated the interaction between different objects between their appearance and image locations. The new proposed framework did not need additional annotation and showed improvements in object detection performance. Based on Hu et al., Gu et al. [53] proposed a fully learnable object detector which proposed a general viewpoint that unified existing region feature extraction methods. Their proposed method removed heuristic choices in ROI pooling methods and automatically select the most significant parts, including contexts beyond proposals. Another method to encode contextual information is to implicitly encode region features by adding image features surrounding region proposals and a large number of approaches have been proposed based on this idea [45, 9, 217, 18, 229, 108]. In addition to encode features from region proposals, Gidaris et al. [45] extracted features from a number of different sub-regions of the original object proposals (border regions, central regions, contextual regions etc.) and concatenated these features with the original region features. Similar to their method, [9] extracted local contexts by enlarging the proposal window size and concatenating these features with the original ones. Zeng et al. [229] proposed Gated Bi-Directional CNN (GBDNet) which extracted features from multi-scale subregions. Notably, GBDNet learned a gated function to control the transmission of different region information because not all contextual information is helpful for detection.

### Deformable Feature Learning

A good detector should be robust to nonrigid deformation of objects. Before the deep learning era, Deformable Part based Models (DPMs) [32] had been successfully used for object detection. DPMs represented objects by multiple component parts using a deformable coding method, making the detector robust to nonrigid object transformation. In order to enable detectors based on deep learning to model deformations of object parts, many researchers have developed detection frameworks to explicitly model object parts [25, 144, 216, 49]. DeepIDNet [144] developed a deformable-aware pooling layer to encode the deformation information across different object categories. Dai et al. [25] and Zhu et al. [216] designed deformable convolutional layers which automatically learned the auxiliary position offsets to augment information sampled in regular sampling locations of the feature map.

## 2.4 Learning Strategy

In contrast to image classification, object detection requires optimizing both localization and classification tasks, which makes it more difficult to train robust detectors. In addition, there are several issues that need to be addressed, such as imbalance sampling, localization, acceleration etc. Thus there is a need to develop innovative learning strategies to train effective and efficient detectors. In this section, we review some of the learning strategies for object detection.

### 2.4.1 Training Stage

In this section, we review the learning strategies for training object detectors. Specifically we discuss, data augmentation, imbalance sampling, cascade learning, localization refinement and some other learning strategies.

**Data Augmentation.**

Data augmentation is important for nearly all deep learning methods as they are often data-hungry and more training data leads to better results. In object detection, in order to increase training data as well as generate training patches with multiple visual properties, Horizontal flips of training images is used in training Faster R-CNN detector [47]. A more intensive data augmentation strategy is used in one-stage detectors including rotation, random crops, expanding and color jittering [123, 9, 183]. This data augmentation strategy has shown significant improvement in detection accuracy.

### Imbalance Sampling

In object detection, imbalance of negative and positive samples is a critical issue. That is, most of the regions of interest estimated as proposals are in fact just background images. Very few of them are positive instances (or objects). This results in problem of imbalance while training detectors. Specifically, two issues arise, which need to be addressed: class imbalance and difficulty imbalance. The class imbalance issue is that most candidate proposals belong to the background and only a few of proposals contain objects. This results in the background proposals dominating the gradients during training. The difficulty imbalance is closely related to the first issue, where due to the class imbalance, it becomes much easier to classify most of the background proposals easily, while the objects become harder to classify. A variety of strategies have been developed to tackle the class imbalance issue. Two-stage detectors such as R-CNN and Fast R-CNN will first reject majority of negative samples and keep 2000 proposals for further classification. In Fast R-CNN [47], negative samples were randomly sampled from these 2k proposals and the ratio of positive and negative was fixed as 1:3 in each mini-batch, to further reduce the adverse effects of class imbalance. Random sample can address class imbalance issue but are not able to fully utilize information from negative proposals. Some negative proposals may contain rich context information about the images, and some hard proposals can help to improve detection accuracy. To address this, Liu et al. [123] proposed hard negative sampling strategy which fixed the foreground and background ratio but sampled most difficult negative proposals for updating the model. Specifically, negative proposals with higher classification loss were selected for training.

To address difficulty imbalance, most sampling strategies are based on carefully designed loss functions. For obejct detection, a *multi-class* classifier is learned over C+1 categories (C target categories plus one background category). Assume the region is labeled with ground truth class $u$, and $p$ is the output discrete probability distribution over C+1 classes ($p = \{p_0, ..., p_C\}$). The loss function is given by:

$$L_{\text{cls}}(p, u) = -\log p_u \tag{2.9}$$

Lin et al. proposed a novel focal loss[119] which suppressed signals from easy samples. Instead of discarding all easy samples, they assigned an importance weight to each sample w.r.t its loss value as:

$$L_{\text{FL}} = -\alpha(1 - p_u)^{\gamma} \log(p_u) \tag{2.10}$$

where $\alpha$ and $\gamma$ were parameters to control the importance weight. The gradient signals of easy samples got suppressed which led the training process to focus more on hard proposals. Li et al. [8] adopt a similar idea from focal loss and propose a novel gradient harmonizing mechanism (GHM). The new proposed GHM not only suppressed easy proposals but also avoided negative impact of outliers. Shrivastava et al. [179] proposed an online hard example mining strategy which was based on a similar principle as Liu et al.'s SSD [123] to automatically select hard examples for training. Different from Liu et al., online hard negative mining only considered difficulty information but ignored categorical information, which meant the ratio of foreground and background was not fixed in each mini-batch. They argued that difficult samples played a more important role than class imbalance in object detection task.

### Localization Refinement

An object detector must provide a tight localization prediction (bbox or mask) for each object. To do this, many efforts refine the preliminary proposal prediction to improve the localization. Precise localization is challenging because predictions are commonly focused on the most discriminative part of the objects, and not necessarily the region containing the object. In some scenarios, the detection algorithms are required to make high quality predictions (high IoU threshold) See Fig. 2.8 for an illustration of how a detector may fail in a high IoU threshold regime. A general approach for localization refinement is to generate high quality proposals (See Sec 2.3.4). In this section, we will review some other methods for localization refinement. In R-CNN framework, the L-2 auxiliary bounding box regressors were learned to refine localizations, and in Fast R-CNN, the smooth L1 regressors were learned via an end-to-end training scheme as:

$$L_{\text{reg}}(t^c, v) = \sum_{i \in \{x,y,w,h\}} \text{SmoothL1}(t_i^c - v_i) \tag{2.11}$$

$$\text{SmoothL1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \tag{2.12}$$

where the predicted offset is given by $t^c = (t_x^c, t_y^c, t_w^c, t_h^c)$ for each target class, and $v$ denotes ground truth of object bounding boxes($v = (v_x, v_y, v_w, v_h)$). $x, y, w, h$ denote bounding box center, width and height respectively.

Beyond the default localization refinement, some methods learn auxiliary models to further refine localizations. Gidaris et al. [45] introduced an iterative bounding box regression method, where an R-CNN was applied to refine learned predictions. Here the predictions were refined multiple times. Gidaris et al. [46] proposed LocNet which modeled the distribution of each bounding box and refined the learned predictions. Both these approaches required a separate component in the detection pipeline, and prevent joint optimization.

Some other efforts [228, 130] focus on designing a unified framework with modified objective functions. In MultiPath Network, Zagoruyko et al. [228] devel-

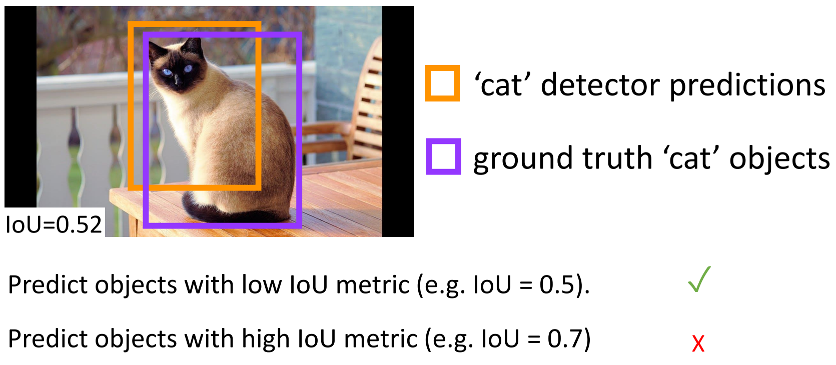

'cat' detector predictions
ground truth 'cat' objects
IoU=0.52
Predict objects with low IoU metric (e.g. IoU = 0.5). ✓
Predict objects with high IoU metric (e.g. IoU = 0.7) x

Figure 2.8: Example of failure case of detection in high IoU threshold. Purple box is ground truth and yellow box is prediction. In low IoU requirement scenario, this prediction is correct while in high IoU threshold, it's a false positive due to insufficient overlap with objects. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

oped an ensemble of classifiers which were optimized with an integral loss targeting various quality metrics. Each classifier was optimized for a specific IoU threshold and the final prediction results were merged from these classifiers. Tychsen et al. proposed Fitness-NMS [198] which learned novel fitness score function of IoU between proposals and objects. They argued that existing detectors aimed to find *qualified* predictions instead of *best* predictions and thus highly quality and low quality proposals received equal importance. Fitness-IoU assigned higher importance to highly overlapped proposals. They also derived a bounding box regression loss based on a set of IoU upper bounds to maximum the IoU of predictions with objects. Inspired by CornerNet [98] and DeNet [197], Lu et al. [130] proposed a Grid R-CNN which replaced linear bounding box regressor with the principle of locating corner keypoints corner-based mechanism.

### Cascade Learning

Cascade learning is a coarse-to-fine learning strategy which collects information from the output of the given classifiers to build stronger classifiers in a cascaded manner. Cascade learning strategy was first used by Viola and Jones [202] to train the robust face detectors. In their models, a lightweight detector first rejects the majority easy negatives and feeds hard proposals to train detectors in next stage. For deep learning based detection algorithms, Yang et al. [219] proposed CRAFT (Cascade Region-proposal-network And FasT-rcnn) which learned RPN and region classifiers with a cascaded learning strategy. CRAFTS first learned a standard RPN followed by a two-class Fast RCNN which rejected the majority easy negatives. The remaining samples were used to build the cascade region classifiers which consisted of two Fast RCNNs. Yang et al. [220] introduced layer-wise cascade classifiers for different scale objects in different layers. Multiple classifiers were placed on different feature maps and classifiers on shallower layers would reject easy negatives. The remaining samples would be fed into deeper layers for classification. RefineDet [236] and Cascade R-CNN [10] utilized cascade learning methods in refining object locations. They built multi-stage bounding box regressors and bounding box predictions were refined in each stage trained with different quality metrics. Cheng et al. [19] observed the failure cases of Faster RCNN, and noticed that even though the localization of objects was good, there were several classification errors. They attributed this to sub-optimal feature representation due to sharing of features and joint multi-task optimization, for classification and regression; and they also argued that the large receptive field of Faster RCNN induce too much noise in the detection process. They found that vanilla RCNN was robust to these issues. Thus, they built a cascade detection system based on Faster RCNN and RCNN to complement each other. Specifically, A set of initial predictions were obtained from a well trained Faster RCNN, and these predictions were used to train RCNN to refine the results.

#### Others

There are some other learning strategies which offer interesting directions, but have not yet been extensively explored. We split these approaches into four categories: adversarial learning, training from scratch and knowledge distillation.

*Adversarial learning.* Adversarial learning has shown significant advances in generative models. The most famous work applying adversarial learning is generative adversarial network (GAN) [52] where a generator is competing with a discriminator. The generator tries to model data distribution by generating fake images using a noise vector input and use these fake images to confuse the discriminator, while the discriminator competes with the generator to identify the real images from fake images. GAN and its variants [250, 152, 7] have shown effectiveness in many domains and have also found applications in object detection. Li et al. [106] proposed a new framework Perceptual GAN for small object detection. The learnable generator learned high-resolution feature representations of small objects via an adversarial scheme. Specifically, its generator learned to transfer low-resolution small region features into high-resolution features and competed with the discriminator which identified real high-resolution features. Finally the generator learned to generate high quality features for small objects. Wang et al. [207] proposed A-Fast-R-CNN which was trained by generated adversarial examples. They argued the difficult samples were on long tail so they introduced two novel blocks which automatically generated features with occlusion and deformation. Specifically, a learned mask was generated on region features followed by region classifiers. In this case, the detectors could receive more adversarial examples and thus become more robust.

*Training from scratch.* Modern object detectors heavily rely on pre-trained classification models on ImageNet, however, the bias of loss functions and data distribution between classification and detection can have an adversarial impact on the performance. Finetuning on detection task can relieve this issue, but cannot fully get rid of the bias. Besides, transferring a classification model for detection in a new domain can lead to more challenges (from RGB to MRI data etc.). Due to these reasons, there is a need to train detectors from scratch, instead of relying on pretrained models. The main difficulty of training detectors from scratch is the training data of object detection is often insufficient and may lead to overfitting. Different from image classification, object detection requires bounding box level annotations and thus, annotating a large scale detection dataset requires much more effort and time (ImageNet has 1000 categories for image classification while only 200 of them have detection annotations).

There are some works [174, 81, 251] exploring training object detectors from scratch. Shen et al. [174] first proposed a novel framework DSOD (Deeply Supervised Object Detectors) to train detectors from scratch. They argued deep supervision with a densely connected network structure could significantly reduce optimization difficulties. Based on DSOD, Shen et al. [175] proposed a gated recurrent feature pyramid which dynamically adjusted supervision intensities of intermediate layers for objects with different scales. They defined a recurrent feature pyramid structure to squeeze both spatial and semantic information into a single prediction layer, which further reduced parameter numbers leading to faster convergence. In addition, the gate-control structure on feature pyramids adaptively adjusted the supervision at different scales based on the size of objects. Their method was more powerful than original DSOD. However, later He et al. [81] validated the difficulty of training detectors from scratch on MSCOCO and found that the vanilla detectors could obtain a competitive performance with at least 10K annotated images. Their findings proved no specific structure was required for training from scratch which contradicted the previous work.

*Knowledge distillation.* Knowledge distillation is a training strategy which distills the knowledge in an ensemble of models into a single model via teacher-student training scheme. This learning strategy was first used in image classification [66]. In object detection, some works [109, 19] also investigate this training scheme to improve detection performance. Li et al. [109] proposed a light weight detector whose optimization was carefully guided by a heavy but powerful detector. This light detector could achieve comparable detection accuracy by distilling knowledge from the heavy one, meanwhile having faster inference speed. Cheng et al. [19] proposed a Faster R-CNN based detector which was optimized via teacher-student training scheme. An R-CNN model is used as teacher network to guide the training process. Their framework showed improvement in detection accuracy compared with traditional single model optimization strategy.

### 2.4.2 Testing Stage

Object detection algorithms make a dense set of predictions and thus these predictions cannot be directly used for evaluation due to heavy duplication. In addition, some other learning strategies are required to further improve the detection accuracy. These strategies improve the quality of prediction or accelerate the inference speed. In this section, we introduce these strategies in testing stage including duplicate removal, model acceleration and other effective techniques.

#### Duplicate Removal

Non maximum suppression (NMS) is an integral part of object detection to remove duplicate false positive predictions (See Figure 2.9). Object detection algorithms make a dense set of predictions with several duplicate predictions. For one-stage detection algorithms which generate a dense set of candidate proposals such as SSD [123] or DSSD (Deconvolutional Single Shot Detector) [38], the proposals surrounding the same object may have similar confidence scores, leading to false positives. For two-stage detection algorithms which generates a sparse set of proposals, the bounding box regressors will pull these proposals close to the same object and thus lead to the same problem. The duplicate predictions are regarded as false positives and will receive penalties in evaluation, so NMS is needed to remove these duplicate predictions. Specifically, for each category, the prediction boxes are sorted according to the confidence score and the box with highest score is selected. This box is denoted as $M$. Then IoU of other boxes with $M$ is calculated, and if the IoU value is larger than a predefined threshold $\Omega_{\text{test}}$, these boxes will be removed. This process is repeated for all remaining predictions. More formally, the confidence score of box $B$ which overlaps with $M$ larger than $\Omega_{\text{test}}$ will be set to zero:

$$\text{Score}_B = \begin{cases} \text{Score}_B & \text{IoU}(B, M) < \Omega_{\text{test}} \\ 0 & \text{IoU}(B, M) \geq \Omega_{\text{test}} \end{cases} \tag{2.13}$$

However, if an object just lies within $\Omega_{\text{test}}$ of $M$, NMS will result in a missing prediction, and this scenario is very common in clustered object detection. Navaneeth et al. [6] introduced a new algorithm Soft-NMS to address this issue. Instead of directly eliminating the prediction $B$, Soft-NMS decayed the confidence score of $B$ as a continuous function $F$($F$ can be linear function or guassian function) of its overlaps with $M$. This is given by:

$$\text{Score}_B = \begin{cases} \text{Score}_B & \text{IoU}(B, M) < \Omega_{\text{test}} \\ F(\text{IoU}(B, M)) & \text{IoU}(B, M) \geq \Omega_{\text{test}} \end{cases} \tag{2.14}$$

Soft-NMS avoided eliminating prediction of clustered objects and showed improvement in many common benchmarks. Hosong et al [68] introduced a network architecture designed to perform NMS based on confidence scores and bounding boxes, which was optimized separately from detector training in a supervised way. They argued the reason for duplicate predictions was that the detector deliberately encouraged multiple high score detections per object instead of rewarding one high score. Based on this, they designed the network following two motivations: (i) a loss penalizing double detections to push detectors to predict exactly one precise detection per object; (ii) joint processing of detections nearby to give the detector information whether an object is detected more than once. The new proposed model did not discard detections but instead reformulated NMS as a re-scoring task that sought to decrease the score of detections that cover objects that already have been detected.

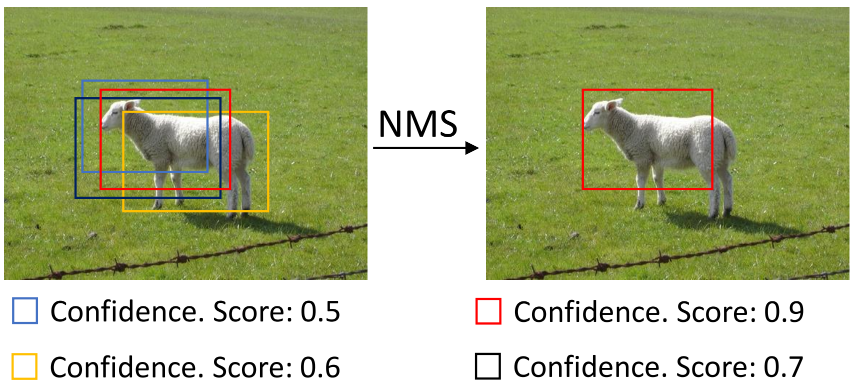


Figure 2.9: Duplicate predictions are eliminated by NMS operation. The most-confident box is kept, and all other boxes surrounding it will be removed.

### Model Acceleration

Application of object detection for real world application requires the algorithms to function in an efficient manner. Thus, evaluating detectors on efficiency metrics is important. Although current state-of-the-art algorithms [74, 62] can achieve very strong results on public datasets, their inference speeds make it difficult to apply them into real applications. In this section we review several works on accelerating detectors. Two-stage detectors are usually slower than one-stage detectors because they have two stages - one proposal generation and one region classification, which makes them computationally more time consuming than one-stage detectors which directly use one network for both proposal generation and region classification. R-FCN [24] built spatially-sensitive feature maps and extracted features with position sensitive ROI Pooling to share computation costs. However, the number of channels of spatially-sensitive feature maps significantly increased with the number of categories. Li et al. [112] proposed a new framework Light Head R-CNN which significantly reduced the number of channels in the final feature map (from 1024 to 16) instead of sharing all computation. Thus, though computation was not shared across regions, but the cost could be neglected.

From the aspect of backbone architecture, a major computation cost in object detection is feature extraction [162]. A simple idea to accelerate detection speed is to replace the detection backbone with a more efficient backbone, e.g., MobileNet [70, 169] was an efficient CNN model with depth-wise convolution layers which was also adopted into many works such as [211] and [111]. P-VANet [88] was proposed as a new network structure with CReLu [172] layer to reduce non-linear computation and accelerated inference speed. Another approach is to optimize models off-line, such as model compression and quantization [89, 64, 51, 121, 213, 55, 56] on the learned models. Finally, NVIDIA Corporation[1] released an acceleration toolkit TensorRT[2] which optimized the computation of learned models for deployment and thus significantly sped up the inference.

### Others

Other learning strategies in testing stage mainly comprise the transformation of input image to improve the detection accuracy. Image pyramids [62, 236] are a widely used technique to improve detection results, which build a hierarchical image set at different scales and make predictions on all of these images. The final detection results are merged from the predictions of each image. Zhang et al. [237, 236] used a more extensive image pyramid structure to handle different scale objects. They resized the testing image to different scales and each scale was responsible for a certain scale range of objects. Horizontal Flipping [60, 236] was also used in the testing stage and also showed improvement. These learning strategies largely improved the capability of detector to handle different scale objects and thus were widely used in public detection competitions. However, they also increase computation cost and thus were not suitable for real world applications.

[1]`https://www.nvidia.com/en-us/`
[2]`https://developer.nvidia.com/tensorrt`

## 2.5 Applications

Object detection is a fundamental computer vision task and there are many real world applications based on this task. Different from generic object detection, these real world applications commonly have their own specific properties and thus carefully-designed detection algorithms are required. In this section, we will introduce several real world applications: face detection and few-shot detection.

### 2.5.1 Face Detection

Face detection is a classical computer vision problem to detect human faces in the images, which is often the first step towards many real-world applications with human beings, such as face verification, face alignment and face recognition. There are some critical differences between face detection and generic detection: (i) the range of scale for objects in face detection is much larger than objects in generic detection. Moreover occlusion and blurred cases are more common in face detection; (ii) Face objects contain strong structural information, and there is only one target category in face detection. Considering these properties of face detection, directly applying generic detection algorithms is not an optimal solution as there could be some priors that can exploited to improve face detection.

In early stages of research before the deep learning era, face detection [203, 143, 154, 166] was mainly based on sliding windows, and dense image grids were encoded by hand-crafted features followed by training classifiers to find and locate objects. Notably, Viola and Jones [203] proposed a pioneering cascaded classifiers using AdaBoost with Haar features for face detection and obtained excellent performance with high real time prediction speed. After the progresses of deep learning in image classification, face detectors based on deep learning significantly outperformed traditional face detectors [187, 125, 195, 20, 107].

Current face detection algorithms based on deep learning are mainly extended from generic detection frameworks such as Fast R-CNN and SSD. These algorithms focus more on learning robust feature representations. In order to handle extreme scale variance, multi-scale feature learning methods discussed before have been widely used in face detection. Sun et al. [187] proposed a Fast R-CNN based framework which integrated multi-scale features for prediction and converted the resulting detection bounding boxes into ellipses as the regions of human faces are more elliptical than rectangular. Zhang et al. [237] proposed one-stage S3FD which found faces on different feature maps to detect faces at a large range of scales. They made predictions on larger feature maps to capture small-scale face information. Notably, a set of anchors were carefully designed according to empirical receptive fields and thus provided a better match to the faces. Based on S3FD, Zhang et al. [232] proposed a novel network structure to capture multi-scale features in different stages. The new proposed feature agglomerate structure integrated features at different scales in a hierarchical way. Moreover, a hierarchical loss was proposed to reduce the training difficulties. Single Stage Headless Face Detector (SSH) [138] was another one-stage face detector which combined different scale features for prediction. Hu et al. [72] gave a detailed analysis of small face detection and proposed a light weight face detector consisting of multiple RPNs, each of which was responsible for a certain range of scales. Their method could effectively handle face scale variance but it was slow for real world usage. Unlike this method, Hao et al. [58] proposed a Scale Aware Face network which addresses scale issues without incurring significant computation costs. They learned a scale aware network which modeled the scale distribution of faces in a given image and guided zoom-in or zoom-out operations to make sure that the faces were in desirable scale. The resized image was fed into a single scale light weight face detector. Wang et al. [204] followed RetinaNet [119] and utilized more dense anchors to handle faces in a large range of scales. Moreover, they proposed an attention function to account for context information, and to highlight the discriminative features. Zhang et al. [233] proposed a deep cascaded multi-task face detector with cascaded structure (MTCNN). MTCNN had three stages of carefully designed CNN models to predict faces in a coarse-to-fine style. Further, they also proposed a new online hard negative mining strategy to improve the result. Samangouei et al. [168] proposed a Face MegNet which allowed information flow of small faces without any skip connections by placing a set of deconvolution layers before RPN and ROI Pooling to build up finer face representations.

In addition to multi-scale feature learning, some frameworks were focused on contextual information. Face objects have strong physical relationships with the surrounding contexts (commonly appearing with human bodies) and thus encoding contextual information became an effective way to improve detection accuracy. Zhang et al. [231] proposed FDNet based on ResNet with larger deformable convolutional kernels to capture image context. Zhu et al. [249] proposed a Contextual Multi-Scale Region-based Convolution Neural Network (CMS-RCNN) in which multi-scale information was grouped both in region proposal and ROI detection to deal with faces at various range of scale. In addition, contextual information around faces is also considered in training detectors. Notably, Tang et al. [195] proposed a state-of-the-art context assisted single shot face detector, named PyramidBox to handle the hard face detection problem. Observing the importance of the contex-t, they improved the utilization of contextual information in the following three aspects: (i) first, a novel context anchor was designed to supervise high-level contextual feature learning by a semi-supervised method, dubbed as PyramidAnchors; (ii) the Low-level Feature Pyramid Network was developed to combine adequate high-level context semantic features and low-level facial features together, which also allowed the PyramidBox to predict faces at all scales in a single shot; and (iii) they introduced a context sensitive structure to increase the capacity of prediction network to improve the final accuracy of output. In addition, they used the method of data-anchor-sampling to augment the training samples across different s-cales, which increased the diversity of training data for smaller faces. Yu et al.[226] introduced a context pyramid maxout mechanism to explore image contexts and devised an efficient anchor based cascade framework for face detection which optimized anchor-based detector in cascaded manner. Zhang et al. [234] proposed a two-stream contextual CNN to adaptively capture body part information. In addition, they proposed to filter easy non-face regions in the shallow layers and leave difficult samples to deeper layers.

Beyond efforts on designing scale-robust or context-assistant detectors, Wang et al. [204] developed a framework from the perspective of loss function design. Based on vanilla Faster R-CNN framework, they replaced original softmax loss with a center loss which encouraged detectors to reduce the large intra-class variance in face detection. They explored multiple technologies in improving Faster R-CNN such as fixed-ratio online hard negative mining, multi-scale training and multi-scale testing, which made vanilla Faster R-CNN adaptable to face detection. Later, Wang et al. [209] proposed Face R-FCN which was based on vanilla R-FCN. Face R-FCN distinguished the contribution of different facial parts and introduced a novel position-sensitive average pooling to re-weight the response on final score maps. This method achieved state-of-the-art results on many public benchmarks such as FDDB [77] and WIDER FACE [222].

### 2.5.2 Few-shot Detection

Few-shot detection means training a detector with only a few objects annotated. Deep learning based detection algorithms requires a huge amount of annotated training data to finetune an immense number of parameters. Without sufficient training data, the detectors cannot guarantee satisfactory results. And in this case, algorithms for few-shot detection are required to address this problem. Before introducing few-shot detection algorithms, in this section, we summarize the existing work about meta learning for classification, which is beyond the area of detection but it is the basic of this work.

Few-shot learning has been widely explored in image classification. One of the promising methods are mainly based on meta learning. [155] optimized the base-model via an LSTM-based meta-learner which simulates traditional SGD optimization method. [36] proposed MAML which learns a good feature initialization which can adapt to a new task in only one gradient step update. Based on MAML, [115] proposed Meta-SGD which learns a set of learnable parameters to control gradient step of different tasks. Learning initialization is potentially a very general idea for few-shot learning however, the training process can be unstable [3] especially for complex problems such as detection. [201] proposed Matching Networks based on a non-parametric principle by learning a differentiable K-Nearest Neighbour model, and [184] proposed Prototypical Networks using the similar principle. [161] extended this idea to semi-supervised learning by self-learning from the unlabeled data. [190] proposed a relation network to automatically define the optimal distance metric. These metric-learning based methods are easy to train and effective for few-shot classification. However, directly adapting these techniques for detection is very challenging as just replacing the object classification branch of a detector with a meta-learner is not sufficient, and training the RPN under a meta-learning paradigm is non-trivial.

In contrast to classification, few-shot detection has received less attention. [28] addressed few-shot detection using large-scale unlabeled data. Their model is based on a semi-supervised method which extracts knowledge from unlabeled dataset to enrich training dataset by self-paced learning and multi-modal learning. However, their method may be misled by the incorrect predictions from initial model and also requires re-training the model for every new task. [13] proposed a Low-shot Transfer Detector (LSTD) using regularization to transfer knowledge from source domain to target domain by minimizing the gap between the two domains. Recently, there have been some concurrent efforts in applying meta-learning for object detection. [170] proposed RepMet as a meta-learning based few-shot detection which replaces the fully connected classification layer of a standard detector with modified prototypical network. However, its suffers from two critical limitations in that RPN and bounding box regression are not tailored for few-shot challenges, and it often fails in distinguishing object classes of interest from background (including object classes not of interest). [84] proposed Meta-YOLO by applying meta-learning with YOLO. They optimize the few-shot detector by re-weighting the channels of global features with support images. [210] proposed MetaDet as a meta-learning framework for few-shot detection. In MetaDet, they disentangle the learning process of class-agnostic parts and class-specific parts, and learn a meta model to predict class-specific parameters from few-shot data.

### 2.5.3 Others

There are some other real applications with object detection techniques, such as logo detection and video object detection.

Logo detection is an important research topic in e-commerce systems. Compared to generic detection, logo instance is much smaller with strong non-rigid transformation. Further, there are few logo detection baselines available. To address this issue, Su et al. [185] adopted the learning principle of webly data learning which automatically mined information from noisy web images and learns models with limited annotated data. Su et al. [186] described an image synthesising method to successfully learn a detector with limited logo instances. Hoi et al. [67] collected a large scale logo dataset from an e-commerce website and conducted a comprehensive analysis on the problem logo detection.

Existing detection algorithms are mainly designed for still images and are sub-optimal for directly applying in videos for object detection. To detect objects in videos, there are two major differences from generic detection: temporal and contextual information. The location and appearance of objects in video should be temporally consistent between adjacent frames. Moreover, a video consists of hundreds of frames and thus contains far richer contextual information compared to a single still image. Han et al. [57] proposed a Seq-NMS which associates detection results of still images into sequences. Boxes of the same sequence are re-scored to the average score across frames, and other boxes along the sequence are suppressed by NMS. Kang et al. proposed Tubelets with Convolutional Neural Networks (T-CNN) [85] which was extended from Faster RCNN and incorporated the temporal and contextual information from tubelets (box sequence over time). T-CNN propagated the detection results to the adjacent frames by optical flow, and generated tubelets by applying tracking algorithms from high-confidence bounding boxes. The boxes along the tubelets were re-scored based on tubelets classification.

There are also many other real-world applications based on object detection such as vehicle detection [245, 42, 133], traffic-sign detection [255, 150] and skeleton detection [87, 173].

## 2.6 Detection Benchmarks

In this section we will show some common benchmarks of generic object detection and face detection. We will first present some widely used datasets for each task and then introduce the evaluation metrics.

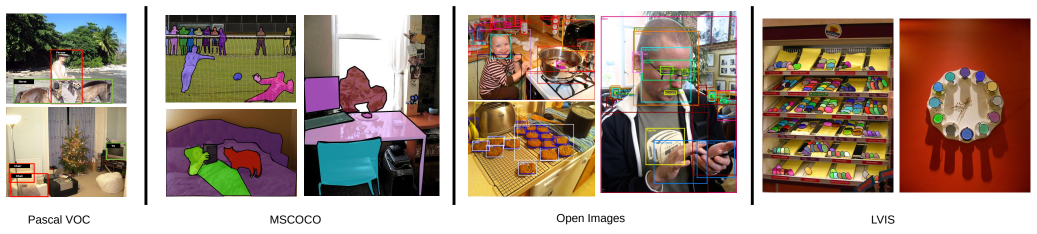


Figure 2.10: Some examples of Pascal VOC, MSCOCO, Open Images and LVIS.

### 2.6.1 Generic Detection Benchmarks

*Pascal VOC2007* [31] is a mid scale dataset for object detection with 20 categories. There are three image splits in VOC2007: training, validation and test with 2501, 2510 and 5011 images respectively.

*Pascal VOC2012* [31] is a mid scale dataset for object detection which shares the same 20 categories with Pascal VOC2007. There are three image splits in

VOC2012: training, validation and test with 5717, 5823 and 10,991 images respectively. The annotation information of VOC2012 test set is not available.

*MSCOCO* [120] is a large scale dataset for with 80 categories. There are three image splits in MSCOCO: training, validation and test with 118,287, 5000 and 40,670 images respectively. The annotation information of MSCOCO test set is not available.

*Open Images* [97] contains 1.9M images with 15M objects of 600 categories. The 500 most frequent categories are used to evaluate detection benchmarks, and more than 70% of these categories have over 1000 training samples.

*LVIS* [54] is a new collected benchmark with 164,000 images and 1000+ categories. It is a new dataset without any existing results so we leave the details of LVIS in future work section (Section 2.7).

*ImageNet* [27] is also a important dataset with 200 categories. However, the scale of ImageNet is huge and the object scale range is similar to VOC datasets, so it is not a commonly used benchmarks for detection algorithms.

*Evaluation metrics:* The details of evaluation metrics are shown in Tab. 2.1, both detection accuracy and inference speed are used to evaluate detection algorithms. For detection accuracy, mean Average Precision (mAP) is used as evaluation metric for all these challenges. The mAP is the mean value of AP, which is calculated separately for each class based on recall and precision. Assume the detector returns a set of predictions, we sample top $\gamma$ predictions by confidence in decreasing order, which is denoted as $D_\gamma$. Next we calculate the number of true positive ($TP_\gamma$) and false positive ($FP_\gamma$) from sampled $D_\gamma$ by the metric introduced in Section 2.2. Based on $TP_\gamma$ and $FP_\gamma$, recall ($R_\gamma$) and precision ($P_\gamma$) are easy to obtain. AP is the region area under the curve of precision and recall, which is also easy to compute by varying the value of parameter $\gamma$. Finally mAP is computed by averaging the value of AP across all classes. For VOC2012, VOC2007 and ImageNet, IoU threshold of mAP is set to 0.5, and for MSCOCO, more comprehensive evaluation metrics are applied. There are six evaluation scores which demonstrates different capability

| Alias | Meaning | Definition and description | |
|---|---|---|---|
| FPS | Frame per second | The number of images processed per second. | |
| $\Omega$ | IoU threshold | The IoU threshold to evaluate localization. | |
| $D_\gamma$ | All Predictions | Top $\gamma$ predictions returned by the detectors by confidence in decreasing order. | |
| $TP_\gamma$ | True Positive | Correct predictions from sampled predictions $D_\gamma$. | |
| $FP_\gamma$ | False Positive | False predictions from sampled predictions $D_\gamma$ . | |
| $P_\gamma$ | Precision | The fraction of $TP_\gamma$ out of $D_\gamma$. | |
| $R_\gamma$ | Recall | The fraction of $TP_\gamma$ out of all positive samples. | |
| AP | Average Precision | Region area under curve of $R_\gamma$ and $P_\gamma$ by varying the value of parameter $\gamma$. | |
| mAP | mean AP | Average score of AP across all classes. | |
| TPR | True Positive Rate | The fraction of positive rate over false positives. | |
| FPPI | FP Per Image | The fraction of false positive for each image. | |
| MR | log-average missing rate | Average miss rate over different FPPI rates evenly spaced in log-space | |
| **Generic Object Detection** | | | |
| mAP | mean Average Precision | VOC2007 | mAP at 0.50 IoU threshold over all 20 classes. |
| | | VOC2012 | mAP at 0.50 IoU threshold over all 20 classes. |
| | | OpenImages | mAP at 0.50 IoU threshold over 500 most frequent classes. |
| | | MSCOCO | • $AP_{coco}$: mAP averaged over ten $\Omega$: $\{0.5 : 0.05 : 0.95\}$;<br>• $AP_{50}$: mAP at 0.50 IoU threshold;<br>• $AP_{75}$: mAP at 0.75 IoU threshold;<br>• $AP_S$: $AP_{coco}$ for small objects of area smaller than $32^2$;<br>• $AP_M$: $AP_{coco}$ for objects of area between $32^2$ and $96^2$;<br>• $AP_L$: $AP_{coco}$ for large objects of area bigger than $96^2$; |
| **Face Detection** | | | |
| mAP | mean Average Precision | Pascal Face | mAP at 0.50 IoU threshold. |
| | | AFW | mAP at 0.50 IoU threshold. |
| | | WIDER FACE | • $mAP_{easy}$: mAP for easy level faces;<br>• $mAP_{mid}$: mAP for mid level faces;<br>• $mAP_{hard}$: mAP for hard level faces; |
| TPR | True Positive Rate | FDDB | • $TPR_{dis}$ with 1k FP at 0.50 IoU threshold, with bbox level.<br>• $TPR_{cont}$ with 1k FP at 0.50 IoU threshold, with eclipse level. |

Table 2.1: Summary of common evaluation metrics for various detection tasks including generic object detection, face detection and pedestrian detection.

of detection algorithms, including performance on different IoU thresholds and on different scale objects. Some examples of listed datasets (Pascal VOC, MSCOCO, Open Images and LVIS) are shown in Fig. 2.10.

### 2.6.2 Face Detection Benchmarks

In this section, we introduce several widely used face detection datasets (WIDER FACE, FDDB and Pascal Face) and the commonly used evaluation metrics.

*WIDER FACE* [222]. WIDER FACE has totally 32,203 images with about 400k faces for a large range of scales. It has three subsets: 40% for training, 10% for validation, and 50% for test. The annotations of training and validation sets are online available. According to the difficulty of detection tasks, it has three splits: Easy, Medium and Hard.

*FDDB* [77]. The Face Detection Data set and Benchmark (FDDB) is a well-known benchmark with 5171 faces in 2845 images. Commonly face detectors will first be trained on a large scale dataset (WIDERFACE etc. ) and tested on FDDB.

*PASCAL FACE* [31]. This dataset was collected from PASCAL person layout test set, with 1335 labeled faces in 851 images. Similar to FDDB, it's commonly used as test set only.

*Evaluation metrics.* As Tab. 2.1 shown, the evaluation metric for WIDER FACE and PASCAL FACE is mean average precision (mAP) with IoU threshold as 0.5, and for WIDER FACE the results of each difficulty level will be reported. For FDDB, true positive rate (TPR) at 1k false positives are used for evaluation. There are two annotation types to evaluate FDDB dataset: bounding box level and eclipse level.

## 2.7 Concluding Remarks

In this section, we give a comprehensive understanding of recent advances in deep learning techniques for object detection tasks. The main contents are divided into three major categories: object detector components, machine learning strategies, real-world applications and benchmark evaluations. We have reviewed a large body of representative articles in recent literature, and presented the contributions on this important topic in a structured and systematic manner. We hope this section can give readers a comprehensive understanding of object detection with deep learning.

# Part I

# Scale-invariant Detection

# Chapter 3

# Feature Agglomeration Networks with Application to Face Detection

## 3.1 Introduction

In real-world scenarios, the scale variance of objects is significantly larger than the variance in existing benchmarks for generic object detection. Applying generic detection frameworks into face detection directly fails to guarantee satisfactory results. In this dissertation, we select face detection as our benchmark to evaluate algorithms for scale-invariant detection. Face detection is a real world computer vision problem to detect human face in the wild, and is often the first key step towards face related applications, such as face alignment, face verification, face recognition, face tracking and facial expression analysis, etc. Despite being studied extensively, detecting faces in the wild remains an open research problem due to various challenges with real-world faces, such as varied scales of faces.

In general, many previous state-of-the-art face detectors inherited a lot of successful techniques from generic object detection, especially for the family of region-based CNN (R-CNN) methods and their variants [162, 47, 5]. Among the family of R-CNN based face detectors, there are two major categories of detection frameworks: (i) two-stage detectors (a.k.a. "proposal-based"), such as Fast R-CNN [47],

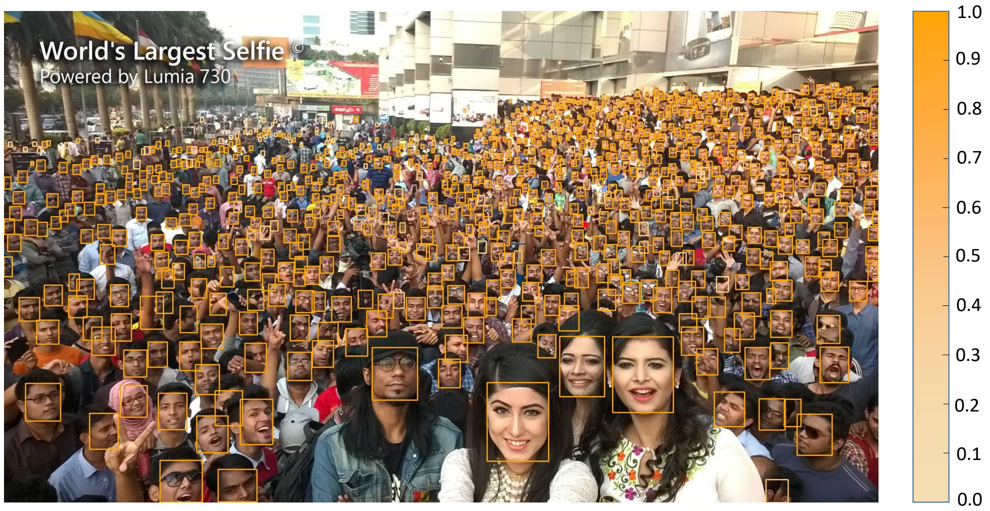


Figure 3.1: Example of face detection with the proposed method. In the above image, the proposed method can find 858 faces out of 1000 facial images present. The detection confidence scores are also given by the color bar as shown on the right. Best viewed in color.

Faster R-CNN [162], etc; and (ii) single-stage detectors (a.k.a. "proposal-free") , such as Region Proposal Networks (RPN) [162], Single-Shot Multibox Detector (SSD) [123], etc. The single-stage detection framework enjoys much higher inference efficiency, and thus has attracted increasing attention recently due to the high demand of real-time face detectors in real applications.

Despite enjoying significant computational advantages, single-stage detectors are not always effective in detecting faces of different scales and their performance can drop dramatically when handling small faces. In order to build a robust detector that can detect faces with a large range of scales, there are two major routes for improvement. One way is to train multi-shot single-scale detectors by using the idea of image pyramid to train multiple separate single-scale detectors each of which is tuned for one specific scale (e.g., the HR detector in [72] trained multiple scale-specific RPN detectors). However, such approach with the image pyramid is computationally expensive since it has to pass a very deep network multiple times during inference. Another way is to train a single-shot multi-scale detector by exploiting multi-scale representations in the feature hierarchy of a deep convolutional network, requiring only a single pass to the network during inference. For example, the Single-shot multi-scale Face Detector (S3FD) in [237] follows the second approach by extending SSD [123] for face detection.

Despite achieving promising performance, S3FD shares the similar drawback of SSD-style detection frameworks, where each of multi-scale feature maps is used alone for prediction and thus a high-resolution semantically weak feature map may fail to perform accurate predictions. Inspired by the recent success of Feature Pyramid Networks (FPN) [118] for generic object detection, we propose a novel detection framework of “Feature Agglomeration Networks” (FAN) to overcome the drawback of the single stage face detector “S3FD” by attempting to combine low-resolution semantically strong features with high-resolution semantically weak features. In particular, FAN aims to create a feature pyramid with rich semantics at all scales to boost the prediction performance of high-resolution feature maps using rich contextual cues from low-resolution semantically strong features.

However, unlike the existing FPN for generic object detection that creates feature pyramid using the skip connection module, we propose a novel “Agglomerative Connection” module to create a new feature pyramid for FAN. Moreover, we introduce a new Hierarchical Loss to train the FAN model effectively in an end-to-end approach. We conduct extensive experiments on several public face detection benchmarks, in which our results show that FAN is more effective than a naive application of FPN [118] for face detection (though FPN has yet to be explored for face detection), and the proposed Hierarchical Loss is also critical to train the proposed FAN model.

As a summary, the main contributions of this work include the following

- We propose a novel framework of Feature Agglomeration Networks (FAN) for single stage face detection, which creates a new effective feature pyramid with rich semantics at all scales by introducing a new hierarchical agglomerative connection module to agglomerate multi-scale features via a hierarchical agglomerative manner in which effective receptive field of each feature map can be easily controlled through hierarchical design;

- We introduce an effective Hierarchical Loss based training scheme to train the proposed FAN model in an end-to-end manner, which enables us to learn discriminative features of the feature pyramid effectively;
- We conducted comprehensive experiments on several public Face Detection benchmarks to evaluate the effectiveness of the proposed FAN framework, in which promising results show that our FAN detector not only achieves the state-of-the-art performances but also runs efficiently with real-time speed on GPU.

## 3.2 Feature Agglomeration Networks

In this section, we present the proposed Feature Agglomeration Networks (FAN) framework for face detection.

### 3.2.1 General Architecture

Our goal is to create an effective feature hierarchy with rich semantics at all levels to achieve robust multi-feature detection. To this end, we propose a novel hierarchical feature agglomeration structure which agglomerates adjacent features sequentially to construct a Feature Agglomeration Network (FAN) for detection. Figure 3.2 shows an example of the proposed FAN with 3-level feature hierarchies.

The proposed FAN framework is general-purpose and applicable to any types of detectors and CNN architectures. In this work, without loss of generality, we consider the widely used VGG16 model as the backbone CNN architecture and SSD as the single stage detector. Suppose detection is performed on $n$ layers of feature maps (ranging from index $1$ to $n$), then the network structure in $m$-level

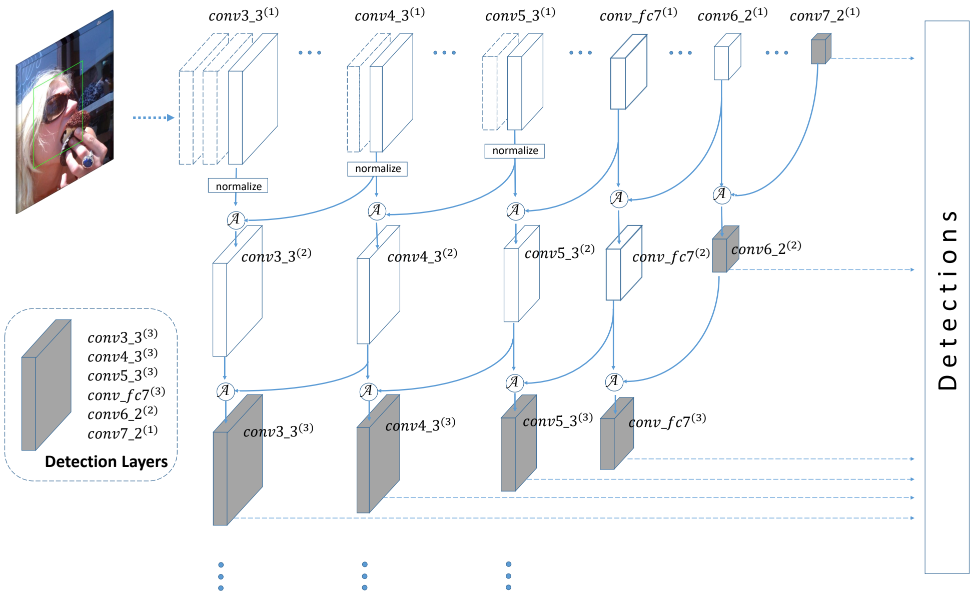


Figure 3.2: The network architecture of the proposed "Feature Agglomeration Networks" (FAN). Here demonstrates an example of three-level FAN architecture using the VGG-16 variant as the backbone CNN network.

FAN ($m \leq n$) can be mathematically defined as follows:

$$\phi_l^k = \mathcal{F}_l(\phi_{l-1}^k) \qquad k = 1 \tag{3.1}$$

$$\phi_l^k = \mathcal{A}_l(\phi_l^{k-1}, \phi_{l+1}^{k-1}) \quad k = 2, .., m \tag{3.2}$$

where $\phi_l^k$ denotes the feature maps in the $l$-th layer and the $k$-th hierarchy. Specifically, for $k = 1$, i.e., the first-level hierarchy, $\phi_l^k$ is the original feature maps in vanilla SSD, and $\mathcal{F}_l(\cdot)$ is the non-linear function to transform the feature maps from $l$-th layer to $(l+1)$-th layer, which consists of Convolution, ReLU and Pooling layers, etc. For $k > 1$, Eq.(3.2) denotes that $\phi_l^k$ is generated through an agglomeration function $\mathcal{A}_l$ to agglomerate two adjacent-layer feature maps in the same hierarchy $\phi_l^{k-1}$, $\phi_{l+1}^{k-1}$. This is called an "Agglomerative Connection" building block as shown in Figure 3.3, denoted as $\mathcal{A}$-block for short.

**Agglomerative Connection block**

Specifically, each $\mathcal{A}$-block consists of two input feature maps, a shallower $\phi_l^{k-1}$ and a deeper $\phi_{l+1}^{k-1}$. We first use a $1 \times 1$ convolution to change the channel of the shallower feature $\phi_l^{k-1}$ to a fixed number $N$ (e.g., 256). Then the dimensionality of the deeper feature $\phi_{l+1}^{k-1}$ is reduced via a $1 \times 1$ convolution to $\frac{1}{8}$ of $N$ (e.g., 32) followed by a $2\times$ bilinear upsampling in order to achieve the same size as $\phi_l^{k-1}$. The final agglomerative feature $\phi_l^k$ is obtained by the concatenation of these two features.

The main difference between our Agglomerative Connection block and commonly used skip connection block as FPN is that we firstly apply dimension reduction ($1 \times 1$ Convolution) on deeper feature maps, then Concatenation instead of Summation is used to merge two features. Since summation requires two features with the same channel numbers, our design enjoy the merits of controlling the proportion of contextual info from deeper layers.

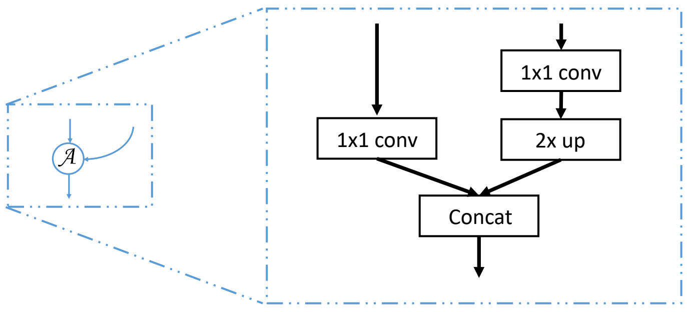


Figure 3.3: The Agglomerative Connection block ($\mathcal{A}$-block).

The final detection exploits the $m$-th hierarchy of feature maps, e.g., if $m = 3$, the detection layers of features are $\{\text{conv3\_3}^{(3)}, \text{conv4\_3}^{(3)}, \text{conv5\_3}^{(3)}, \text{conv\_fc7}^{(3)}, \text{conv6\_2}^{(2)}, \text{conv7\_2}^{(1)}\}$, and the detection result denotes as

$$Result \;=\; \mathcal{D}(\phi_l^m, \phi_{l+1}^m, ..., \phi_{l+n-1}^m) \tag{3.3}$$

where $\mathcal{D}$ denotes the final detection process including bounding box regression and class prediction followed by Non-Maximum Suppression to obtain the final detection results. To make the notation consistent, we use superscript $m$ to denote all the feature maps in the $m$-th level hierarchy. However, $\mathrm{conv7\_2}^{(3)}$ and $\mathrm{conv7\_2}^{(2)}$ is actually identical to $\mathrm{conv7\_2}^{(1)}$, and $\mathrm{conv6\_2}^{(3)}$ is identical to $\mathrm{conv6\_2}^{(2)}$, etc.

It is worth noting that the proposed network degenerates to vanilla SSD detector if the agglomeration operation Eq.(3.2) is excluded and the first hierarchy is used for detection, i.e., $m = 1$ in Eq.(3.3). The insight behind hierarchical agglomerative design is that in vanilla SSD, shallower features which are important to detect small faces are semantically weak in feature representation. The $\mathcal{A}$-block hierarchically aggregates semantics information from deeper layers to form a stronger set of hierarchical multi-feature maps. The ratio of deeper feature and shallower feature in one $\mathcal{A}$-block is set to $\frac{1}{8}$ which ensures that the shallower feature dominates the composition and the deeper (semantically stronger) feature generally plays a role of providing extra contextual cues. Besides, we note that the receptive field largely impacts the performance of detecting small faces. As shown in [72], too large receptive field can hurt the performance and so as if it is too small. The superiority of our design is that the agglomerative connection only incorporates semantics from a deeper layer feature, and thus we can easily control the receptive field of each feature map through our hierarchical design. This is in contrast to FPN [118] where a feature map incorporates information from all the deeper layers. We found our new design had the attributes to achieve stable and effective training.

### 3.2.2 Detailed Configurations

In this section, we discuss more details about the proposed FAN framework, which can be designed in a flexible way. In practice, assuming that detection is performed on $n$ feature maps of a CNN model (specifically n=6 in VGG-16 as used in our experiment), we can design a FAN structure with $m$-level hierarchies where $m \leq 6$. Figure 3.2 showed a 3-level hierarchy FAN structure. The detection layers of feature maps $\{\text{conv3\_3}^{(3)}, \text{conv4\_3}^{(3)}, \text{conv5\_3}^{(3)}, \text{conv\_}fc7^{(3)}, \text{conv6\_2}^{(2)}, \text{conv7\_2}^{(1)}\}$ have strides of $\{4, 8, 16, 32, 64, 128\}$, respectively. We follow the settings of [237], each of the six detection feature maps is associated with a specific scale anchor $\{16, 32, 64, 128, 256, 512\}$ to detect corresponding scale faces. The aspect ratio of each anchor is 1:1 since the size of a face is roughly 1:1.

For anchor-based detectors, we need to match each anchor as a positive or negative sample according to the ground truth bounding boxes. We adopt the following matching strategy: (i) for each face, the anchor with best jaccard overlap is matched; and (ii) each anchor is matched to the face that has jaccard overlap larger than $0.35$. Max-out background label for the lowest feature map $\text{conv3\_3}^{(3)}$ is also adopted [237]. Specifically, for each anchor of $\text{conv3\_3}^{(3)}$, $M = 3$ scores are predicted and then the highest score is chosen as the final background score.

## 3.2.3 Hierarchical Loss

In order to train the proposed FAN model effectively, we propose a new loss function called the hierarchical loss defined on the proposed FAN structure. The key idea is to define a loss function that accounts for all the hierarchies of feature maps, and meanwhile allows to train the entire network effectively in an end-to-end approach. To this end, we propose the hierarchical loss as follows

$$\text{HL}(\phi_l^m, ..., \phi_{l+n-1}^m) = \sum_{i=1}^{m} \omega_i L(\phi_l^i, ..., \phi_{l+n-1}^i) \tag{3.4}$$

where $\omega_i$ is a weight parameter for the loss of the $i$-th hierarchy. $\text{L}(\phi_l^i, ..., \phi_{l+n-1}^i)$ accounts for the loss on the $i$-th hierarchy, which is defined as follows

$$\text{L}(\phi_l^i, ..., \phi_{l+n-1}^i) = \frac{1}{N_i} \sum_{j=1}^{N_i} \Big( \lambda L_{cls}(y_j, y_j^*) + L_{loc}(\mathbf{p}_j, \mathbf{p}_j^*) \Big) \tag{3.5}$$

where $y_j$ denotes if the corresponding anchor is a face or not, $y_j^* \in \{0, 1\}$ is the ground truth label, $\mathbf{p}_j = [p_x, p_y, p_w, p_h]_j$ denotes the 4 coordinates of a predicted bounding box, $\mathbf{p}_j^*$ denotes the ground-truth box, $N_i$ denotes the total number of matched bounding boxes, and $\lambda$ is a parameter to tradeoff between classification loss and localization loss. In particular, the classification loss is based on a standard softmax loss, i.e.,

$$L_{cls}(y_j, y_j^*) = y_j^* \log(y_j) + (1 - y_j^*) \log(1 - y_j) \tag{3.6}$$

and the localization of bounding box is based on a standard regression loss proposed in [47] defined as follows

$$L_{loc}(\mathbf{p}_j, \mathbf{p}_j^*) = \sum_{k \in \{x,y,w,h\}} \mathrm{smooth}_{L_1}(\mathbf{p}_j - \mathbf{p}_j^*)_k \tag{3.7}$$

where

$$\mathrm{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1, \\ |x| - 0.5 & \text{otherwise.} \end{cases} \tag{3.8}$$

Using the proposed hierarchical loss, we can train the FAN detector end-to-end. Specifically, during training, all the losses are simultaneously computed, and the gradients are back propagated to each hierarchy of feature maps respectively.

**Remarks.** In contrast to the standard loss, the proposed hierarchical loss enjoys some key advantages. On one hand, the use of hierarchical loss plays a crucial role in training the FAN model robustly and effectively. This is because, in contract to the network of vanilla SSD, FAN has more newly added parameters for optimization, which is not easy to be directly trained with the existing loss in vanilla SSD training. With multiple hierarchies, hierarchical loss gradually increases the power of feature maps representation, and thus allows us to supervise the training process hierarchically to obtain more robust features in which lower level losses can also be seen as a kind of regularization to guide better training. On the other hand, compared with the standard single loss, after the model has been trained, the use of hierarchical loss does not incur extra computation cost during inference.

### 3.2.4 Training

Next we introduce more details of our training method.

**Training data and data augmentation.** Our model was trained on $12,880$ images of the WIDER FACE training set [222]. We use several data augmentation strategies as follows. First of all, we follow a similar color distortion strategy [123] to preprocess training images, e.g., brightness, contrast, hue, saturation, etc. Random crop is adopted to generate training images [237]. Specifically, to keep the face aspect ratio which is important due to our anchor scale design, instead of directly resizing the whole image to a squared patch (e.g., we use $640 \times 640$ as input size for training), we first crop a squared patch from original image whose scale ranges from $0.3$ to $1$ of the short size of original image. The overlapped part of the face box is discarded if and only if its center is out of the sampled patch. After random cropping, the final patch is resized to $640 \times 640$ and horizontally flipped with probability of $0.5$.

**Hard negative mining.** After the anchor matching, most of the anchors will be assigned as negative samples, which will result in a significant imbalance between positive and negative samples. Instead of using all negative samples for training, we use an online hard negative mining strategy [123] during training. In particular, all negative anchors are sorted by their classification loss values, and then the top ones are selected as negative samples to ensure the ratio between negative and positive anchors is at most $3:1$.

**Other implementation details.** In our experiments, we choose $\lambda = 3$ in Eq.(3.5) and the weight parameter $\omega_i$ in Eq.(3.4) as uniform for simplicity. The training starts from fine-tuning VGG16 backbone network using SGD optimizer with momentum of 0.9, weight decay of 0.0005, and a total batch size of 12 on two GPUs. The newly added layers are initialized with “xavier”, the initial learning rate is $10^{-3}$ and becomes 10 times smaller at iteration $80k$ and $100k$. The training ended at $120k$ iterations. Our implementation is based on Caffe [79].

## 3.3 Experiments

In this section, we conduct extensive experiments and ablation studies to evaluate the effectiveness of the proposed FAN framework in two folds. First, we examine the impact of several key components including the proposed hierarchical agglomeration connection module, the layer-wise hierarchical loss, and other techniques used in our solution. Second, we compare the proposed FAN face detector with the state-of-the-art face detectors on several popular face detection benchmarks and finally evaluate the inference speed of the proposed face detector.

### 3.3.1 Model Analysis.

**Dataset.** We conduct model analysis on the WIDER FACE dataset [222], which has 32,203 images with about 400k faces for a large range of scales. It has three subsets: 40% for training, 10% for validation, and 50% for test. The annotations of training and validation sets are online available. According to the difficulty of detection tasks, it has three splits: Easy, Medium and Hard. The evaluation metric is mean average precision (mAP) with Interception-of-Union (IoU) threshold as 0.5. We train FAN on the training set of WIDER FACE, and evaluate it on the validation set. For WIDER FACE evaluation, if without multi-scale inference in Table 5.2 and Table 3.2, the results are obtained by single scale testing in which the shorter size of image is resized to 768 while keeping image aspect ratio.

| | S3FD | | | | | | FAN (ours) |
|---|---|---|---|---|---|---|---|
| 1-Level Agglomeration | √ | √ | | | | | |
| 2-Level Agglomeration | | | √ | | √ | | |
| 3-Level Agglomeration | | | | √ | | √ | √ |
| Hierarchical Loss? | | | | | √ | √ | √ |
| Multi-scale inference? | | √ | | | | | √ |
| WIDER FACE mAP (Easy) | 92.9 | 93.7 | 93.8 | 93.0 | 93.8 | 94.0 | **94.8** |
| WIDER FACE mAP (Medium) | 91.8 | 92.5 | 92.6 | 91.9 | 92.7 | 92.9 | **93.8** |
| WIDER FACE mAP (Hard) | 83.4 | 85.9 | 85.8 | 85.3 | 86.0 | 86.4 | **87.6** |

Table 3.1: Ablation studies of FAN. All settings are trained on the training set of WIDER FACE and then tested on the validation set.

**Baseline.** We adopt the closely related detector S3FD [237] as the baseline to validate the effectiveness of our technique. S3FD achieved the previous state-of-the-art results on several well-known face detection benchmarks. It inherited the standard SSD framework with carefully designed scale-aware anchors according to effective receptive fields. We follow the same experimental setup in [237].

**Hierarchical Agglomerative Connection.** We first validate the contribution of *Agglomerative Connection* module with different hierarchical levels. When hierarchical level equals to 1, FAN degenerates to Vanilla S3FD. In Table 5.2, FAN with 2-level Agglomerative Connection outperforms baseline with a large margin in all three difficulty levels. More specifically, we notice the performance of 2-level FAN with single-scale inference is even comparable to S3FD with multi-scale inference, which validates the effectiveness of the agglomerative connection module. As increasing the hierarchical level from 2 to 3, the results become slightly worse than before, but still outperforms the baseline consistently. We argue this is because the complexity of FAN increases as the hierarchical level becomes deeper, traditional learning scheme suffers from the training. As the hierarchies increasing, i.e., $m = 4$ or higher, the improvement gradually become saturated. Next we will show that Hierarchical Loss is necessary in effectively training FAN with high hierarchical-level Agglomerative Connection.

**Hierarchical Loss.** We optimize 2-level FAN and 3-level FAN with Hierarchical Loss. In Table 5.2, the performance of 3-level FAN with Hierarchical Loss gains significant improvement compared with its single loss setting in all difficulty levels (+1.0% in Easy, +1.0% in Medium and +1.1% in Hard), while 2-level FAN with Hierarchical Loss also gains slight improvement. We argue this is because optimization for 2-level FAN is not very difficult so that traditional optimization methods can still handle. In Hierarchical Loss optimization scheme, 3-level FAN outperforms 2-level FAN consistently, which indicates high level Agglomerative Connection is crucial to improve detection accuracy with Hierarchical Loss optimization method.

**Robust Feature Learning.** We compare agglomerative connection with skip connection which was widely used in building feature pyramids. We build "S3FD w/ FPN" based on S3FD with skip-connection in top-down structure as FPN does [118]. In Table 3.2, compared with vanilla S3FD, "S3FD w/ FPN" gains improvement in Hard level, which validates the efficacy of feature pyramid for improving shallow features. Our FAN outperforms "S3FD w/ FPN" with large margin, which indicates the superiority of our agglomerative connection over the skip connection.

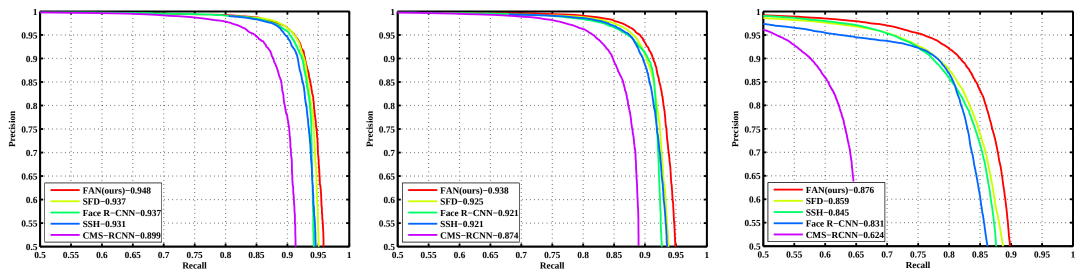
Moreover, to further validate the robust features learned by FAN, we remove all the Agglomerative Connection Module in 3-level FAN trained with the Hierarchical Loss, which shares the same network structure as the Vanilla S3FD. We use this model ("S3FD w/ HL ") to make inference as S3FD does. The results in Table 3.2 show this "truncated" model achieves a high detection accuracy and outperforms both Vanilla S3FD and "S3FD w/ FPN". This proves that the proposed structure of FAN enables us to learn robust and discriminative features.

| Loss | Easy | Medium | Hard |
|---|---|---|---|
| Vanilla S3FD | 92.9 | 91.8 | 83.4 |
| S3FD w/ FPN | 92.9 | 91.8 | 84.7 |
| S3FD w/ HL | 93.4 | 92.4 | 85.2 |
| FAN | **94.0** | **92.9** | **86.4** |

Table 3.2: Evaluation of our FAN with agglomerative connection and hierarchical loss (HL) for learning discriminative features in contrast to vanilla S3FD and a simple FPN with skip connection.

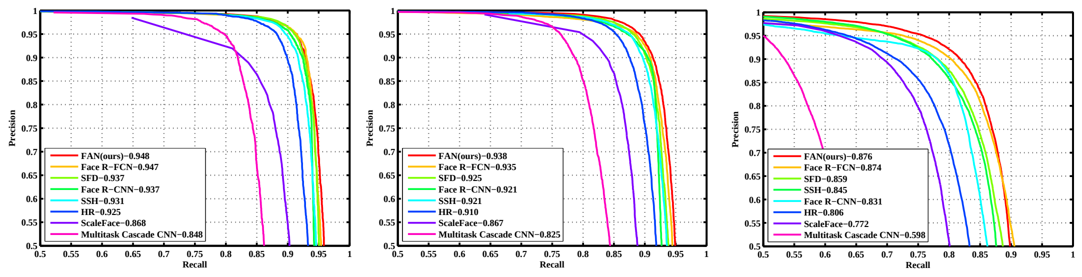
**Multi-scale Inference.** Multi-scale testing is a widely used technique in object detection, which can further boost the detection accuracy especially for small objects. In Table 5.2, both vanilla S3FD and FAN gain improvements of detection accuracy. We conduct multi-scale method during inference with fixing the aspect ratio of images.

**Comparisons with the State-of-the-Art.** We use a 3-level FAN network trained by the Hierarchical Loss as our final face detector to compare with various state-of-the-art detectors on the WIDER FACE datasets. Figure 3.4 and Figure 3.5 show the precision-recall curves and Table 3.3 summarizes the overall results on the WIDER



(a) Easy (b) Medium (c) Hard

Figure 3.4: Evaluation of VGG-based methods on the validation set of WIDER FACE



(a) Easy (b) Medium (c) Hard

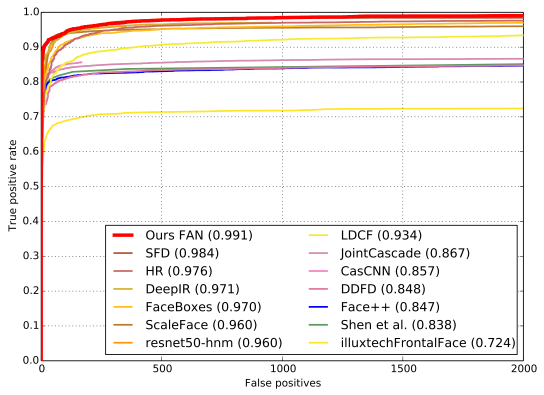
Figure 3.5: Evaluation of various state-of-the-art methods on the validation set of WIDER FACE

| Algorithms | Backbone | Easy | Med | Hard |
|---|---|---|---|---|
| ACF-WIDER [140] | - | 65.9 | 54.1 | 27.3 |
| LDCF+ [140] | - | 79.0 | 76.9 | 52.2 |
| ScaleFace [223] | ResNet50 | 86.8 | 86.7 | 77.2 |
| HR [72] | ResNet101 | 92.5 | 91.0 | 80.6 |
| Face R-FCN [209] | ResNet101 | 94.7 | 93.5 | 87.4 |
| CMS-RCNN [249] | VGG16 | 89.9 | 87.4 | 62.4 |
| SSD-face [237] | VGG16 | 92.1 | 89.5 | 71.6 |
| RPN-face [237] | VGG16 | 91.0 | 88.2 | 73.7 |
| Face-RCNN [204] | VGG19 | 93.7 | 92.1 | 83.1 |
| SSH [138] | VGG16 | 93.1 | 92.1 | 84.5 |
| S3FD[237] | VGG16 | 93.7 | 92.5 | 85.9 |
| FAN(ours) | VGG16 | **94.8** | **93.8** | **87.6** |

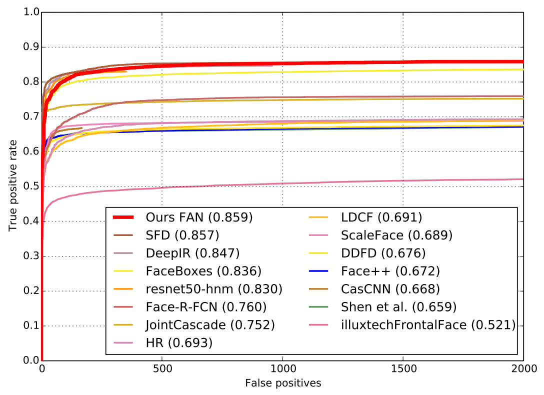
Table 3.3: Evaluation (mAP) on the validation set of WIDER FACE.

Face validation set.

FAN outperforms all VGG-based detectors with large margin, especially in Hard difficulty level. Compared with ResNet-based detectors which utilize much stronger backbone architecture, FAN still reaches the state-of-the-art results, while enjoying a clear advantage of high-inference speed. WIDER FACE is a very challenging face benchmark and the results strongly prove the effectiveness of FAN in handling high scale variances, especially for small faces.



(a) FDDB Discrete ROC Curves



(b) FDDB Continuous ROC Curves

Figure 3.6: Evaluation on FDDB face detection benchmarks.

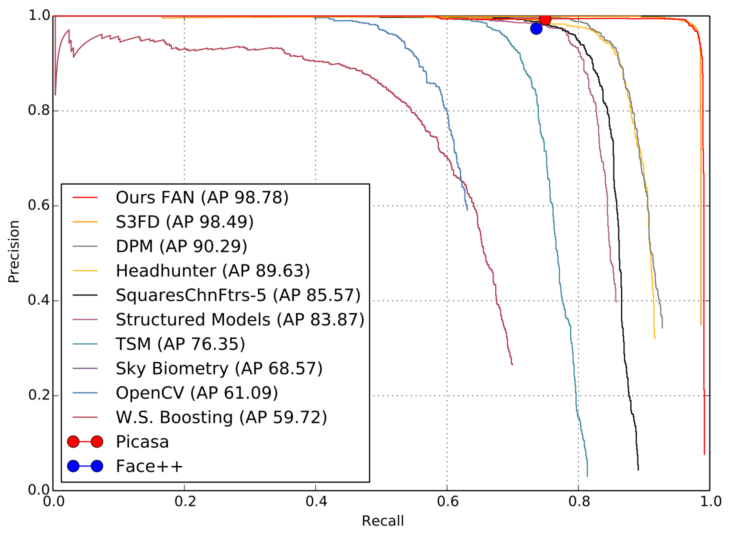


Figure 3.7: Evaluation on PASCAL FACE detection benchmarks.

### 3.3.2 Evaluation on Other Public Face Benchmarks

**FDDB.** The Face Detection Data set and Benchmark (FDDB)[77] is a well-known benchmark with 5,171 faces in 2,845 images. We compare our FAN detector trained on the WIDER FACE training set with other published results on FDDB. Figure 3.6

shows the evaluation results, in which our FAN detector achieves the state-of-the-art performance on both discrete and continuous ROC curves.

**PASCAL FACE.** This dataset was collected from PASCAL person layout test set [31], with 1,335 labeled faces in 851 images. Figure 3.7 shows the evaluation results of the precision-recall curves. Among all the existing methods, the proposed FAN achieved the best mAP (98.78%), which outperforms the previous state-of-the-art detectors S3FD (98.45%) and SSH (98.27%)[1][138], and significantly beats the other submitted methods [252, 221, 135, 82, 12].

### 3.3.3 Inference Time

Our FAN detector is a single-stage detector and thus enjoys high inference speed. FAN runs 32 FPS in GTX 1080ti and CuDNN v5.1 with a VGA-resolution input image. The majority of the time cost is spent on the VGG16 backbone network, while the Agglomerative Connection module is computationally efficient and has little extra cost.

## 3.4 Discussion

This work proposed a novel framework of "Feature Agglomeration Networks" (FAN) for building single stage face detectors. The proposed FAN based face detector not only achieves the state-of-the-art performance in various common face detection benchmarks, but also runs very fast and enjoys real-time inference speed on GPU. FAN introduces two key novel components: (i) the proposed feature "Agglomerative Connection" module agglomerates multi-scale features and contextual information by hierarchical structure, which effectively handles scale variance in face detection; and (ii) the proposed Hierarchical Loss allows to train the FAN model robustly in an end-to-end manner. Our future direction is to extend and apply the Feature Agglomeration Networks (FAN) framework for more computer vision

[1]We cannot plot their curve as their result file is not available.

tasks, including generic object detection or specialized object detection tasks in other domains, such as pedestrian detection, car detection, etc.

# Part II

# High-quality Detection

# Chapter 4

# Bidirectional Pyramid Networks for High-quality Object Detection

## 4.1 Introduction

Most existing object detectors are designed for achieving localization with relatively low-quality precision (e.g. Intersection over Union (IoU) threshold of 0.5 is considered good enough). When the goal is to achieve higher quality localization precision (IoU>0.5), the detection performance often drops significantly [10]. However, real-world applications such as autopilot, requires detectors with high localization ability, where the goal is to achieve higher quality localization precision (IoU=0.7 etc.). Most existing frameworks match objects by a set of pre-defined anchors, and sample positive anchor candidates based on whether their IoU with the objects larger than the threshold (0.5 by default). In this case, the ill-designed anchors significantly hurt the detector performance in high-quality evaluation metric. A naive solution to address this issue is to increase the IoU threshold when selecting positive samples (e.g., from 0.5 to 0.7) during training, such that the detector is trained on only high-quality examples. Unfortunately, such a strategy will lead to very few (positive) training samples, and will consequently lead to overfitting, especially for single-shot SSD-like detectors. In addition, most object detectors aim to use the strength of deep features for object localization. This can have adverse effects as deep features (while being semantically rich) lack detailed information about the spatial location of the objects.

In this work, we aim to develop a novel high-quality single-shot detector. We follow the family of single-stage SSD-like detectors, and design an approach that makes it amenable for high-quality detection. We identify two critical drawbacks of SSD-like detectors for learning high-quality detectors: first, the single-shot feature representations may not be discriminative and robust enough for precise localization; and second, the singe-stage detection scheme relies on the predefined anchors which are very rigid and often inaccurate. To overcome these drawbacks for high-quality object detection tasks, in this chapter, we propose a novel single-shot detection framework named "Bidirectional Pyramid Networks" (BPN). Specifically, BPN uses a novel Bidirectional Pyramid Structure, that boosts the vanilla feature pyramid [118] by reinforcing it with a Reverse Feature Pyramid to fuse both deep and shallow features to learn more effective and robust representations. Unlike Feature Pyramid Network (FPN) which aims to enhance the shallow features with semantically rich deep features, the Reverse FPN aims to enhance the deep features with spatially rich shallow features, thereby improving the representation for better localization. BPN is also augmented with a novel Anchor Refinement scheme that learns to gradually improve the quality of predefined anchors which are often inaccurate at the beginning. Specifically, we train the bounding box regressors at different levels of qualtiy (IoU thresholds), and in an incremental manner, feed the bounding box predictions of a specific quality into the predictions of the next higher quality. We conducted extensive experiments on PASCAL VOC and MSCOCO showed that the proposed method achieved the state-of-the-art results for high-quality object detection while still maintaining the advantage of computational efficiency of single shot detectors.

## 4.2 Single-Shot Detector for High-Quality Detection

To train a detector, predefined anchors are often used. These anchors are generated densely or sparsely across the image, and the goal is to predict the class of object and the appropriate corrections to the original anchor localization. Each anchor is assigned to some object class label (including background) according to the anchor's Jaccard overlap score with ground-truth objects, a.k.a. "Intersection over Union" (IoU). When an anchor matches with the object for a given threshold, it is termed as a positive anchor. These positive anchors serve as ground truth for training. For objects that do not meet this threshold with any anchor, the best anchor is assigned as a positive anchor during the training stage. Our aim is to devise a new single-shot detector for high-quality object detection tasks by overcoming the drawbacks of state-of-the-art detectors. We tackle this challenge from both *feature representation* and *anchor-refining* perspectives. Existing single-shot object detectors, feature representations may not be discriminate and robust enough for precise localization, as they rely primarily on the deep layer features which while being semantically-rich, lack spatial information. We propose to strengthen deep layer features with spatially rich shallow feature to improve the localization performance. Second, for many state-of-the-art detectors, a group of anchors are often generated/pre-defined on the feature maps densely or sparsely, followed by location regression and object classification prediction. Due to the scale variance of the objects, and several downsampling steps from the original image, the manually designed anchors will often not be able to find a good match with the ground truth object locations. This issue becomes more prominent when we aim to train high-quality detectors with a high IoU threshold (e.g., 0.7) since the number of positive anchors would decrease significantly as IoU increases. This would consequently result in poor detection performance due to overfitting. Thus, we propose a novel anchor refinement procedure to improve the localization prediction.

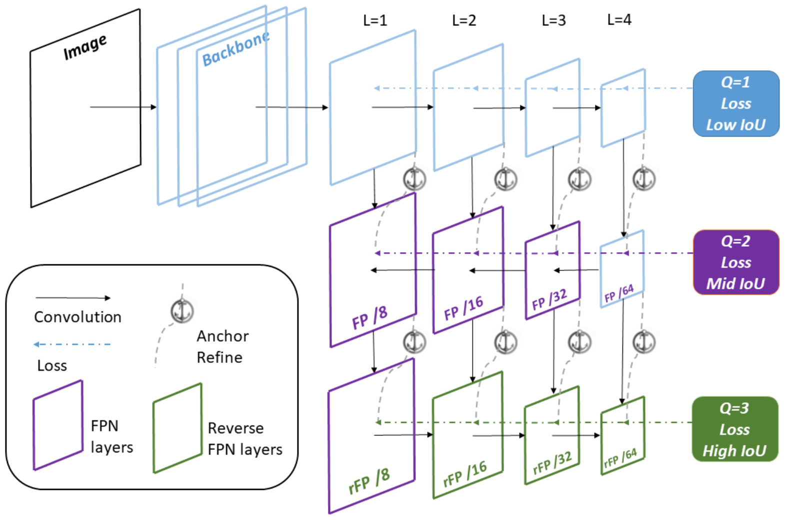


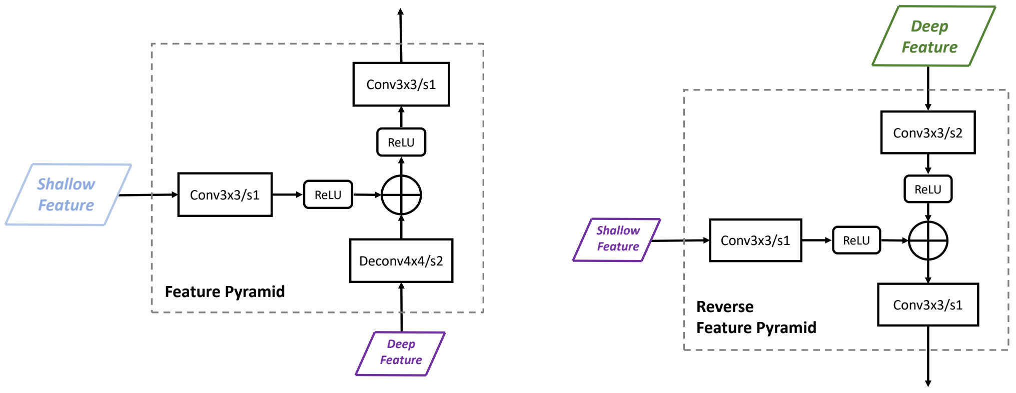
Figure 4.1: The proposed framework of Bidirectional Pyramid Networks (BPN) for single-shot high-quality detection. *FP* denotes Feature Pyramid building block, and *rFP* denotes the Reverse Feature Pyramid building block. Bidirectional Feature Pyramid block generates more robust and discriminative feature map and the Anchor Refinement (*AR*) is utilized for relocating anchors, each level of which is responsible for a certain quality of detection. Training sample quality improves as the Anchor Refinement progresses (with higher IoU).

### 4.2.1 Framework of Bidirectional Pyramid Networks

We propose a novel framework called Bidirectional Pyramid Networks (BPN) to overcome the above drawbacks of SSD-style detectors, with the aim of developing a high-quality object detector. To address the weak feature representation issue of SSD-style detectors, we adapt the structure Feature Pyramid Networks (FPN) [118] and develop a novel Bidirectional Feature Pyramid structure that significantly boosts the effectiveness of Feature Pyramid(FP) structure. To address the issue of anchor quality, the key idea is to devise an effective yet efficient multi-level learning scheme to refine the quality of the anchors. We have classifiers and regressors at multiple levels, and for each level we train the classifier and regressor to refine anchors, before training the classifiers and regressors in the next level. Figure 4.1 gives an overview of the proposed single-shot Bidirectional Pyramid Networks (BPN) for high-quality object detection, where the backbone network (as shown in the blue branch of Figure 4.1) can be any CNN network, such as Alexnet [96], GoogleNet [192], VGG [181], ResNet [62], etc. For simplicity, we choose VGG-16 and ResNet-101 as backbone networks.

Similar to typical single-shot detectors, at the lowest quality level with the default IoU=0.5, the proposed BPN detector makes the prediction based on the predefined anchors. Then, the features are further enhanced by the Bidirectional Feature Pyramid which aggregates features from different depths. It consists of standard feature pyramids in a bottom-up fashion (the purple branch of Figure 4.1) and reverse feature pyramid in a top-down fashion (the green branch of Figure 4.1). These three-level branches not only aggregate multi-level features to provide robust feature representations, but also enable multi-quality training. For the joint training with multiple quality levels, the Anchor Refinement scheme with multi-level learning optimizes anchors from the previous level/branch and sends them to the next level/branch.

The above two key components, Bidirectional Feature Pyramid and Anchor Refinement, are seamlessly integrated in the proposed framework and can be trained end-to-end to achieve high-quality detection in a synergic manner. In the following, we present the detailed functioning of these components.



(a) Feature Pyramid (b) Reverse Feature Pyramid

Figure 4.2: The proposed Bidirectional Feature Pyramid Network Structure

### 4.2.2 Bidirectional Feature Pyramid Structure

We denote the index of feature maps for prediction as $L$, where $L \in \{1, 2, 3, 4\}$ in our setting, and the levels of quality $Q \in \{1, 2, 3, \ldots\}$ with the corresponding IoU thresholds as $\text{IoU}(Q) \in \{0.5, 0.6, 0.7, ...\}$. The feature map in depth $L$ for quality $Q$ prediction is denoted as $F_L^Q$, and anchors for training quality $Q$ detector in depth $L$ are denoted as $A_L^Q$. Specifically for this work, we choose three types of detectors with different quality levels: *Low*, *Mid* and *High* with the corresponding IoU threshold as 0.5, 0.6 and 0.7 respectively (See Figure 4.1 for details).

In order to improve the power of feature representation of SSD-style detectors, we apply Feature Pyramids (FP) [118], which exploits the inherent multi-scale and pyramidal hierarchy of deep convolutional networks to construct the representation of feature pyramids. Specifically, FPN fuses semantically-strong deep layer features with shallow features which are semantically-weak but spatially-strong. The idea is to strengthen the features by helping them with stronger semantic information. We propose to augment this structure via a reverse Feature Pyramid (rFP), where the deep features are strengthened by the spatially strong shallow features.

Reverse Feature Pyramid has several strengths. First, the deep feature representations are enhanced to for better localization of large objects in the high-quality scenario; second, compared to stacked CNN for image classification, rFP reduces the *distance* from shallow features to deep features by using much fewer convolution filters and thus more effectively preserves spatial information. Finally, the lateral connections *reuse* different shallow layer features to reduce information attenuation from shallow features to deep features. We demonstrate this concept in Figure 4.2. Specifically, Figure 4.2(a) is the vanilla Feature Pyramid building block that fuses features in a bottom-up manner with lateral connections. It is worth noting that there is no strengthening of the deepest feature layer from the Feature Pyramid (the right diagram of Figure4.1). Thus, we further build the Reverse Feature Pyramid by top-down aggregation (as shown in Figure 4.2 (b)) with lateral connections to enhance deep layer features with rich spatial information.

The formulations of Feature Pyramid (FP) and reverse Feature Pyramid (rFP) can be represented as:

$$\begin{aligned}\text{FP}: \quad F_L^Q &= \text{Deconv}_{s2}(F_{L+1}^Q) \oplus \text{Conv}(F_L^{Q-1}) \qquad (4.1)\\ \text{rFP}: \quad F_L^Q &= \text{Conv}_{s2}(F_{L-1}^Q) \oplus \text{Conv}(F_L^{Q-1}) \qquad (4.2)\end{aligned}$$

where $\text{Deconv}_{s2}$ denotes the deconvolution operation for feature map up-sampling with stride 2 and $\text{Conv}$ denotes convolution operation. $\oplus$ denotes element-wise summation. In this work, we use $3 \times 3$ convolution kernels with $256$ channels to build the Feature Pyramid and Reverse Feature Pyramid in our BPN detector.

### 4.2.3 Anchor Refinement

In order to both increase the number of positive anchors during training and improve their quality, we propose the Anchor Refinement (“AR”). We denote the anchors used at quality $Q$, depth $L$ as $\text{AR}_L^Q$. In particular, AR has two parts: location regressor $\text{Reg}_L^Q$ and a categorical classifier $\text{Cls}_L^Q$. At each level of quality, regressors receive the processed anchors from the previous level of quality for further optimization ($A_L^1$ is the set of manually defined anchor):

$$A_L^Q = Reg^Q(A_L^{Q-1}; F_L^Q), \quad Q = 2, 3, \ldots, L = 1, 2, \ldots \qquad (4.3)$$

A set of offsets is learned from the regressors to adjust the location of the predicted bounding boxes. Different from vanilla SSD, these bounding boxes are conditioned on the refined anchors and are be used as new anchors in next stage.

Categorical classifiers learn to predict categorical confidence scores and assign them to these anchors:

$$C_L^Q = Cls^Q(F_L^Q), \quad Q = 1, 2, 3 \ldots, L = 1, 2, \ldots \qquad (4.4)$$

Thus, the training loss at quality level $Q$ can be written as:

$$\begin{aligned}\ell^{Q} =& \frac{1}{N_{\text{Q}}} * \sum_{L}\sum_{i} \Big( \ell^{Q}_{\text{Cls}}(\{C^{Q}_{L_i}\}, \{t_{L_i}\}) \\ &+ \lambda * \ell^{Q}_{\text{Reg}}(\{A^{Q}_{L_i}\}, \{g_{L_i}\}) \Big)\end{aligned} \tag{4.5}$$

where $N_{\text{Q}}$ is the positive sample number at quality level $Q$, $L_i$ is the index of anchor in depth $L$ feature map within a mini-batch, $t_{L_i}$ is the ground truth class label of anchor $L_i$, $g_{L_i}$ is the ground truth location and size of anchor $L_i$, $\lambda$ is the balance weighting parameter which is simply set to 1 in our settings. $L^{Q}_{\text{Cls}}(.)$ is softmax loss function over multiple classes confidences and $L^{Q}_{\text{Reg}}(.)$ is the Smooth L1-loss which is also used in [123]. The total training loss is the summation of losses at all the quality levels:

$$\ell_{\text{BPN}} = \sum_{Q} \ell^{Q} \tag{4.6}$$

### 4.2.4 Implementation Details

**CNN Backbone Architecture:** We choose VGG16 [181] and ResNet-101 [62] pre-trained on ImageNet as the backbone networks in our experiments. For VGG16, we follow [123] to transform the last two fully-connected layers “fc6” and “fc7” to convolutional layers “conv_fc6” and “conv_fc7” via reducing parameters. To increase receptive fields and capture large objects, we attached two additional convolution layers after the VGG16 (denoted as conv6_1 and conv6_2). Due to different scale norm in different feature maps, we re-scale the norms of the first two feature blocks to 10 and 8 respectively. For ResNet-101, we added one extra residual block “res6” at the end of the network.

**Data Augmentation:** We adopt the augmentation strategies in [123] to make the detectors robust to objects with the changes in scale and color. Specifically, images are randomly expanded or cropped with additional photometric distortion to generate additional training samples.

**Feature Blocks for Prediction:** In order to detect objects at different scales, we use multiple feature maps for prediction. The vanilla convolution feature blocks in backbone are used for low-quality detection, feature pyramid blocks are used for mid-quality detection, and the reverse feature pyramid blocks are used for high-quality detection. We use four feature blocks with stride 8, 16, 32 and 64 pixels in training each quality detector. In VGG16, conv4_3, conv5_3, conv_fc7, conv6_2 and their corresponding feature pyramid blocks FP3, FP4, FP5 and FP6, and reverse feature pyramid blocks rFP3, rFP4, rFP5 and rFP6 are used, while in ResNet-101, res3b3, res4b22, res5c, res6 and their corresponding feature pyramid blocks and reverse feature pyramid blocks are used.

**Anchor Design:** Originally a group of anchors are pre-designed manually. For each prediction feature block, one scale-specific set of anchors with three aspect ratios isssociated. In our approach, we set the scale of anchors as 4 times that of the feature map stride and set the aspect ratios as 0.5, 1.0 and 2.0 to cover different scales of objects. We first match each object to the anchor box with the best overlap score, and then match the anchor boxes to any ground truth with overlap higher than the quality thresholds.

**Optimization:** We use "Xavier" method in [50] to randomly initialize the parameters in extra added layers in VGG16 and ResNet-101. We set the mini-batch size as 32 in training and the whole network is optimized via the SGD optimizer (momentum=0.9, weight decay=0.005, and initial learning rate=0.001). The training strategy varies a bit for different datasets. For PASCAL VOC dataset, the models are completely finetuned for 120k iterations and we decrease the learning rate to $10^{-4}$ and $10^{-5}$ after 80k and 100k iterations, respectively. For MSCOCO, the models are finetuned for 400k iterations and we decrease the learning rate to $10^{-4}$ and $10^{-5}$ after 280k and 360k iterations, respectively. All the detectors were trained and optimized end-to-end.

**Sampling Strategy:** The ratio of positive and negative anchors are imbalanced after the anchor matching step, so proper sampling strategy is necessary to address this imbalance. We sample a subset of negative anchors to keep the ratio of positive and negative anchors as 1:3 in training process. To achieve faster convergence, instead of randomly sampling negative anchors, we sort the negative anchors according to the loss sufferred by them and select the hardest ones for training. Different IoU thresholds are used for different quality levels. We use three quality levels (low, mid and high) for IoU as 0.5, 0.6 and 0.7 respectively.

**Inference:** During the inference phase, the anchor refinement different quality stage makes prediction and send the refined anchors to the next quality stage. We take the predictions from AR in all quality stages to ensure they are suitable for all the low-, mid- and high-quality detection.

## 4.3 Experiments

We conduct extensive experiments on two publicly available benchmark datasets: Pascal VOC and MSCOCO. The evaluation metric for the detector performance is mean average precision which is widely used in evaluating object detection.

### 4.3.1 Pascal VOC

We use Pascal VOC2007 trainval set and Pascal VOC2012 trainval set as our training set, and VOC2007 test set as testing set. There are 16k images for training and 5k images for testing. All models are based on VGG16 architecture as ResNet-101 has limited benefits for this dataset [38]. We train BPN with two resolutions of the input ($320 \times 320$ and $512 \times 512$) and compare them with the state-of-the-art methods on low, mid and high-quality detection scenarios (IoU thresholds as 0.5, 0.6 and 0.7 respectively).

We show the comparison of performance of our proposed method BPN320 and BPN512 against several state of the art two-stage and one-stage baseline detectors in Table 4.1. BPN320 obtains an accuracy of 80.3%, 75.5% and 66.1% in low, mid and high-quality detection scenario respectively, which outperforms many detectors (e.g., SSD320, Faster RCNN, etc.). BPN512 achieves the state-of-the-art results of 82.2%, 77.6% and 68.3% for three scenarios respectively. Notably, BPN has clear advantage in high-quality detection scenario(IoU=0.7). BPN is one-stage detector, and can thus be used for real-time inference. BPN320 can perform inference at 32.4fps while BPN512 at 18.9fps on a Titan XP GPU.

| Method | Backbone | Input size | FPS | mAP (%) | | |
|---|---|---|---|---|---|---|
| | | | | IoU@0.5 | IoU@0.6 | IoU@0.7 |
| *Two-stage Detectors:* | | | | | | |
| Fast R-CNN [47] | VGG-16 | $\sim 1000 \times 600$ | 0.5 | 70.0 | 62.4 | 49.4 |
| Faster R-CNN [162] | VGG-16 | $\sim 1000 \times 600$ | 7 | 73.2 | 67.7 | 54.4 |
| OHEM [179] | VGG-16 | $\sim 1000 \times 600$ | 7 | 74.6 | 68.9 | 55.9 |
| HyperNet [95] | VGG-16 | $\sim 1000 \times 600$ | 0.88 | 76.3 | - | - |
| Faster R-CNN [62] | ResNet-101 | $\sim 1000 \times 600$ | 2.4 | 76.4 | 69.5 | 57.3 |
| ION [5] | VGG-16 | $\sim 1000 \times 600$ | 1.25 | 76.5 | - | - |
| LocNet [46] | VGG-16 | $\sim 1000 \times 600$ | - | 77.5 | - | 64.5 |
| R-FCN [24] | ResNet-101 | $\sim 1000 \times 600$ | 9 | 80.5 | 73.2 | 61.8 |
| R-FCN Cascade [10] | ResNet-101 | $\sim 1000 \times 600$ | 7 | 81.0 | 75.8 | 66.7 |
| CoupleNet [254] | ResNet-101 | $\sim 1000 \times 600$ | 8.2 | 81.7 | 76.6 | 66.8 |
| *One-stage Detectors:* | | | | | | |
| RON384 [94] | VGG-16 | $384 \times 384$ | 15 | 75.4 | 66.8 | 54.2 |
| SSD300 [123] | VGG-16 | $300 \times 300$ | 46 | 77.3 | 72.3 | 61.3 |
| DSOD300 [174] | DS/64-192-48-1 | $300 \times 300$ | 17.4 | 77.7 | 73.4 | 63.6 |
| YOLOv2 [158] | Darknet-19 | $544 \times 544$ | 40 | 78.6 | 69.1 | 56.5 |
| SSD512 [123] | VGG-16 | $512 \times 512$ | 19 | 79.8 | 74.7 | 64.0 |
| RefineDet320 [236] | VGG-16 | $320 \times 320$ | 40.3 | 80.0 | 74.2 | 63.6 |
| RefineDet512 [236] | VGG-16 | $512 \times 512$ | 24.1 | 81.8 | 76.9 | 66.0 |
| RFBNet300 [122] | VGG-16 | $300 \times 300$ | 83.0 | 80.7 | 75.5 | 65.5 |
| RFBNet512 [122] | VGG-16 | $512 \times 512$ | 38.0 | 82.2 | - | - |
| BPN320(ours) | VGG-16 | $320 \times 320$ | 32.4 | 80.3 | 75.5 | 66.1 |
| BPN512(ours) | VGG-16 | $512 \times 512$ | 18.9 | **82.2** | **77.6** | **68.3** |

Table 4.1: Detection results on PASCAL VOC dataset. All the methods were trained on VOC2007 and VOC2012 `trainval` sets and tested on VOC2007 `test` set.

### 4.3.2 Ablation Study

In this section, we conduct a series of ablation studies to analyze the impact of different components of BPN. We use VOC2007 and VOC2012 `trainval` set as our training set and test on VOC2007 `test` set. We use mean average precision on three different IoU thresholds (0.5, 0.6 and 0.7) as our evaluation metric. The results are shown in Table 4.2.

**Bidirectional Feature Pyramid:** To validate the effectiveness of the Bidirectional

Feature Pyramid, we remove all Anchor Refinement components from BPN leaving only one classifier, and compare this model (called as BPN w / o AR) with vanilla SSD and SSD+FP. Bidirectional Feature Pyramid is built based on vanilla SSD and all three models are fine-tuned with IoU threshold as 0.5. In Table 4.2, we can see that SSD+FP outperforms vanilla SSD because deep semantic features boost feature representations. Further, BPN w / o AR outperforms SSD+FP in all quality scenarios, demonstrating its effectiveness.

**Levels of AR:** We aim to validate if the level of AR is important for training high-quality detectors. We show the results in Table 4.2. Firstly, a vanilla SSD was trained with 0.7 IoU threshold. This model (row 2) performs much worse than the baseline (row 1) trained with 0.5 IoU threshold in all three quality levels, which validates that insufficient positive training samples causes overfitting. Second, we keep a single level of AR block on SSD+FP (called "SSD+FP+AR"), and train this model with 0.5 IoU threshold. We can see that the detection results improve significantly compared with "BPN w/o AR" in low and mid quality scenarios, and is similar in the high-quality scenario (63.6% vs 63.4% ). We further train "SSD+FP+AR" with 0.7 IoU threshold and this model (row 6) also suffers from overfitting issues but it is less severe compared to vanilla SSD. This shows that Anchor Refinement can boost detection performance by refining anchor quality. However, a single level of AR was not enough to boost the performance of the model. Finally, to the above model, we add one more level AR blocks and jointly optimize AR with different quality settings (0.5,0.5,0.7) and (0.5,0.6,0.7), which utilize high quality anchors for training. These two models (row 7 and row 8) further improve the performance significantly especially for high-quality scenario (IoU=0.6 and IoU=0.7, etc.). In summmary, single level of AR is effective in addressing overfitting issues with SS-D, and multi-level of AR are critical for enhancing the detection performance in high-quality scenarios.

**Proposal Quality Improved by Anchor Refinement:** In this section, we validate the effectiveness of the Anchor Refinement blocks to improve the anchor quality.

| | Training IoU | mAP@IoU=0.5 | mAP@IoU=0.6 | mAP@IoU=0.7 |
|---|---|---|---|---|
| SSD | (0.5, - , - ) | 76.3 | 71.0 | 60.4 |
| SSD | (0.7, - , - ) | 68.4 | 61.9 | 50.8 |
| SSD+FP | ( - ,0.5, - ) | 77.4 | 72.1 | 61.6 |
| BPN w / o AR | ( - , - ,0.5) | 78.1 | 72.7 | 63.4 |
| SSD+FP+AR | (0.5, 0.5, - ) | 80.0 | 74.2 | 63.6 |
| SSD+FP+AR | (0.5, 0.7, - ) | 78.1 | 73.7 | 63.1 |
| BPN | (0.5, 0.5, 0.7) | 80.0 | 75.1 | 65.4 |
| BPN | (0.5, 0.6, 0.7) | **80.3** | **75.5** | **66.1** |

Table 4.2: Detection results on PASCAL VOC dataset. For VOC 2007, all methods are trained on VOC 2007 and VOC 2012 `trainval` sets and tested on VOC 2007 `test` set. Original SSD uses six feature maps for prediction, while we use four feature maps to be consistent with BPN, so the detection result of SSD here is a bit lower. "Training IoU" denotes IoU thresholds trained for different stages ("-" means no classifier in this stage). Bold fonts indicate the best mAP.

In Figure 4.3, we count the number of positive anchors per image for training under different IoU thresholds for SSD, SSD+FP+AR and BPN. For SSD, anchors are generated manually and only a few anchors matched objects under high IoU threshold metric, which makes it hard to train effective detectors. For SSD+FP+AR, anchors have been refined by AR once, and the number of positive anchors increases significantly under all IoU thresholds. Further in BPN where anchors are refined by AR twice, more high quality anchors are generated on more robust feature maps. Notably, after being refined by AR we have sufficient positive training samples even under high IoU metrics, so that we could conduct gradually increasing training positive IoU thresholds (0.5, 0.6 and 0.7). These results show that our AR blocks can gradually improve anchor qualities and generate more positive anchors for training.

### 4.3.3 MSCOCO

In addition to PASCAL VOC, we also evaluate BPN on MSCOCO [120]. COCO contains 80 classes objects and about 120k images in `trainval` set. We use `trainval35k` set for training and test on `test-dev` set. Table 4.3 shows the results on MS COCO test-dev set. BPN320 with VGG-16 achieves 29.6% AP and when using larger input image size 512, the detection accuracy of BPN512 reaches

33.1%, which is better than all other VGG16-based methods. Notably, we notice in high-quality detection metric $AP_{75}$, BPN is clearly better than other detectors. As the objects in COCO dataset are of various scales, we also applied multi-scale testing based on BPN320 and BPN512 to reduce the impact of input size. The improved version BPN320++ and BPN512++ achieve 35.4% and 37.9% AP, which is the state-of-the-art performance among one-stage detectors. Different from Pascal VOC, using a deeper backbone such as ResNet could further improve detection accuracy compared to VGG16. Thus we report BPN512 with ResNet-101. Single BPN512 achieves 37.6% AP and when using multi-scale and flip horizontal inference, it improves to 42.3% AP, which is the state-of-the-art performance among one-stage detectors. Notably, BPN512++ achieves 46.3% on $AP_{75}$, which outperforms all other one-stage detectors significantly under high-quality metric.

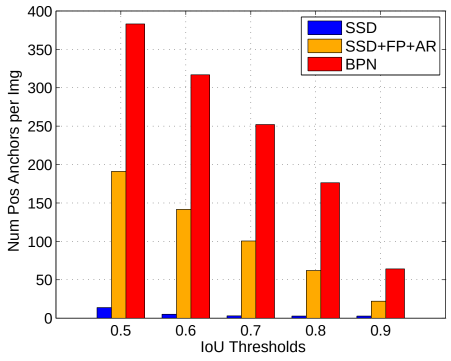


Figure 4.3: Average *positive* anchor number per image by different approaches under different “IoU Threshold” metric.

## 4.4 Discussion

In this work, we proposed a novel single-stage detector framework Bidirectional Feature Pyramid Networks (BPN) for high-quality object detection. It comprises two novel major components: a Bidirectional Feature Pyramid structure for more effective and robust feature representations and an Anchor Refinement component to gradually refine the quality of predesigned anchors for more effective training.

| Method | Backbone | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|
| *two-stage:* | | | | | | | |
| Fast R-CNN [47] | VGG-16 | 19.7 | 35.9 | - | - | - | - |
| Faster R-CNN [162] | VGG-16 | 21.9 | 42.7 | - | - | - | - |
| OHEM [179] | VGG-16 | 22.6 | 42.5 | 22.2 | 5.0 | 23.7 | 37.9 |
| ION [5] | VGG-16 | 23.6 | 43.2 | 23.6 | 6.4 | 24.1 | 38.3 |
| OHEM++ [179] | VGG-16 | 25.5 | 45.9 | 26.1 | 7.4 | 27.7 | 40.3 |
| R-FCN [24] | ResNet-101 | 29.9 | 51.9 | - | 10.8 | 32.8 | 45.0 |
| CoupleNet [254] | ResNet-101 | 34.4 | 54.8 | 37.2 | 13.4 | 38.1 | 50.8 |
| Faster R-CNN by G-RMI [74] | Inception-ResNet-v2[191] | 34.7 | 55.5 | 36.7 | 13.5 | 38.1 | 52.0 |
| Faster R-CNN+++ [62] | ResNet-101-C4 | 34.9 | 55.7 | 37.4 | 15.6 | 38.7 | 50.9 |
| Faster R-CNN w FPN [118] | ResNet-101-FPN | 36.2 | 59.1 | 39.0 | 18.2 | 39.0 | 48.2 |
| FRCNN w Cascade RCNN [10] | VGG16 | 26.9 | 44.3 | 27.8 | 8.3 | 28.2 | 41.1 |
| R-FCN w Cascade RCNN [10] | ResNet-50 | 30.9 | 49.9 | 32.6 | 10.5 | 33.1 | 46.9 |
| R-FCN w Cascade RCNN [10] | ResNet-101 | 33.3 | 52.6 | 35.2 | 12.1 | 36.2 | 49.3 |
| Regionlets [217] | ResNet-101 | 39.3 | 59.8 | - | 21.7 | 43.7 | 50.9 |
| Mask-RCNN [60] | ResNeXt-101 | 39.8 | 62.3 | 43.4 | 22.1 | 43.2 | 51.2 |
| Soft-NMS [6] | Aligned-Inception-ResNet | 40.9 | 62.8 | - | 23.3 | 43.6 | 53.3 |
| Fitness NMS [198] | ResNet-101 | 41.8 | 60.9 | 44.9 | 21.5 | 45.0 | 57.5 |
| Cascade RCNN w FPN [10] | ResNet-101 | 42.8 | 62.1 | 46.3 | 23.7 | 45.5 | 55.2 |
| *one-stage:* | | | | | | | |
| YOLOv2 [158] | DarkNet-19[158] | 21.6 | 44.0 | 19.2 | 5.0 | 22.4 | 35.5 |
| SSD300 [123] | VGG-16 | 25.1 | 43.1 | 25.8 | 6.6 | 25.9 | 41.4 |
| RON384++ [94] | VGG-16 | 27.4 | 49.5 | 27.1 | - | - | - |
| SSD321 [38] | ResNet-101 | 28.0 | 45.4 | 29.3 | 6.2 | 28.3 | 49.3 |
| DSSD321 [38] | ResNet-101 | 28.0 | 46.1 | 29.2 | 7.4 | 28.1 | 47.6 |
| SSD512 [123] | VGG-16 | 28.8 | 48.5 | 30.3 | 10.9 | 31.8 | 43.5 |
| SSD513 [38] | ResNet-101 | 31.2 | 50.4 | 33.3 | 10.2 | 34.5 | 49.8 |
| DSSD513 [38] | ResNet-101 | 33.2 | 53.3 | 35.2 | 13.0 | 35.4 | 51.1 |
| RefineDet320 [236] | VGG-16 | 29.4 | 49.2 | 31.3 | 10.0 | 32.0 | 44.4 |
| RefineDet512 [236] | VGG-16 | 33.0 | 54.5 | 35.5 | 16.3 | 36.3 | 44.3 |
| RefineDet320 [236] | ResNet-101 | 32.0 | 51.4 | 34.2 | 10.5 | 34.7 | 50.4 |
| RefineDet512 [236] | ResNet-101 | 36.4 | 57.5 | 39.5 | 16.6 | 39.9 | 51.4 |
| FoveaBox [93] | ResNeXt-101 | 42.1 | 61.9 | 45.2 | 24.9 | 46.8 | 55.6 |
| CenterNet-HG [243] | Hourglass-104 | 42.1 | 61.1 | 45.9 | 24.1 | 45.5 | 52.8 |
| CornerNet511 [98] | Hourglass-104 | 40.5 | 56.5 | 43.1 | 19.4 | 42.7 | 53.9 |
| CornerNet511++ [98] | Hourglass-104 | 42.1 | 57.8 | 45.3 | 20.8 | 44.8 | 56.7 |
| BPN320 | VGG-16 | 29.6 | 48.4 | 32.3 | 9.6 | 32.5 | 44.3 |
| BPN512 | VGG-16 | 33.1 | 53.1 | 36.3 | 15.7 | 37.0 | 44.2 |
| BPN320++ | VGG-16 | 35.4 | 55.3 | 38.5 | 19.0 | 37.9 | 47.0 |
| BPN512++ | VGG-16 | 37.9 | 58.0 | 41.5 | 21.9 | 41.1 | 48.1 |
| BPN512 | ResNet-101 | 37.6 | 59.1 | 40.5 | 18.7 | 42.2 | 50.8 |
| BPN512++ | ResNet-101 | 42.3 | 62.8 | 46.3 | 25.7 | 46.1 | 53.2 |

Table 4.3: Detection results on MS COCO `test-dev` set. Bold fonts indicate the best performance.

The proposed method achieves state-of-the-art results on Pascal VOC and MSCOCO dataset while enjoying real-time inference speed.

# Chapter 5

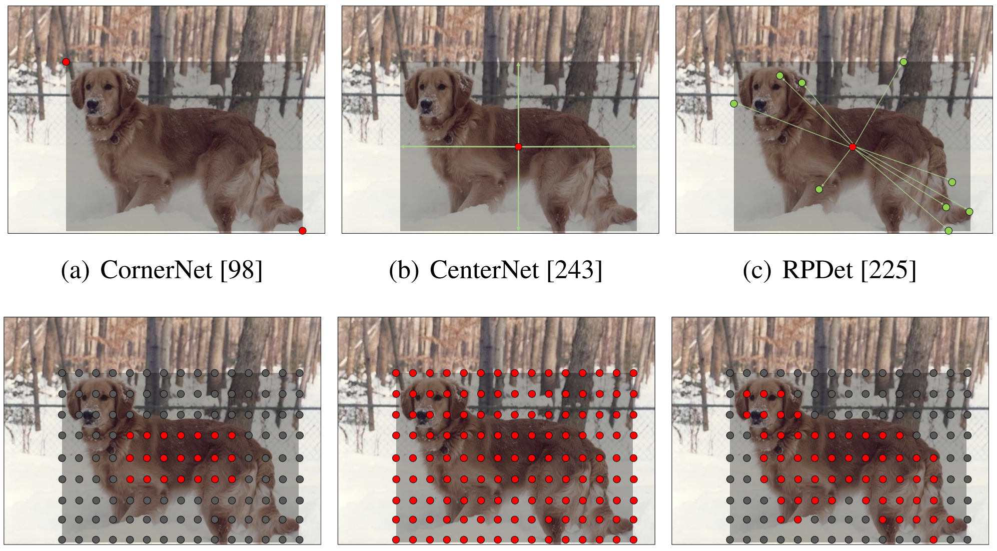
# KPNet: Learning to Optimize Keypoints for Anchor-free Detectors

## 5.1 Introduction

In Chapter 4 we have introduced one high-quality detection framework BPN. Like the mainstream state-of-the-art detectors, BPN is also an anchor based detection method, which heavily rely on the design and selection of appropriate *anchor boxes*. For most existing anchor-based methods, the difficulties of learning high-quality detector is the pre-designed anchors cannot match objects with sufficient IoU, and thus the learned models produce low-quality results. BPN proposes anchor refinement to optimize the shape of anchors to better match objects, and has shown improvement. However, it still requires manually designed anchors for initialization.

Unlike the anchor-based detectors, the anchor-free detectors have emerged recently as a promising direction for object detection that eliminates the need of manually designing anchor boxes [247, 196, 98, 29]. In literature, a variety of anchor-free object detectors have been proposed based on different object modeling strategies and heuristics. For example, CornerNet [98] was proposed for detecting objects by a pair of corner points. Instead of using two corners, CenterNet [243] was proposed by modeling an object as one center point of its bounding box. Besides, there are also a number of other anchor-free detectors that extend the ideas of either Corners based or Centerness based or various other different keypoint design strategies to improve the detection performance.

While anchor-free detectors have been actively explored, we argue that many, if not all, popular anchor-free detectors can be essentially viewed as a special form of keypoint based detectors that adopt different forms of designing/selecting keypoints from a unified keypoint-based detection perspective. Figure 1 (a)-(e) gives an example that compares five popular anchor-free detectors from the view of keypoint based detectors. For example, other than CornerNet and CenterNet, RPDet [225] uses a fixed set of 9 keypoints sampled from the center, FSAF [247] samples a set of multiple keypoints from the center region, while FCOS [196] uses many keypoints by treating every pixel as a keypoint.


(a) CornerNet [98] (b) CenterNet [243] (c) RPDet [225]

(d) FSAF [247] (e) FCOS [196] (f) KPNet (ours)

Figure 5.1: Comparison of different anchor-free object detection methods.

From the view of keypoint based object detection, the popular anchor-free detectors adopt different keypoint design strategies, including some pre-defined keypoints such as CornerNet [98] and CenterNet [243], or a fixed number of keypoints sampled from a pre-defined layout such as RPDet [225], or sampling keypoints from a pre-defineed center region such as FSAF [247], or simply sampling many keypoints by treating all pixels within an object as keypoints such as FCOS [196].

We argue that the existing heuristic keypoint design and sampling strategies may be sub-optimal and do not fully exploit the potential of keypoint based detection techniques. Based on the above motivation, in this chapter, we propose a new keypoint based object detector named “KPNet”, which is able to learn a dynamic set of keypoints automatically without heuristic keypoint designs. Figure 1 (f) illustrates the idea of the proposed KPNet compared with the other anchor-free detectors. As the example shown in the figure, the set of keypoints to be selected can be learned and optimized dynamically from image pixels with different types of objects, which is able to greatly enhance the power of the keypoint based detectors while eliminating manual keypoint design effort.

As a summary, the key contributions of this work include:

- We introduce a unified view of keypoint based object detection for understanding popular anchor-free object detectors, in which many popular anchor-free object detectors can be viewed as a special form of keypoint based detectors with different keypoint design strategies;

- We propose a new anchor-free object detector named “KPNet” which eliminates the heuristic keypoint design and is capable of learning a dynamic set of keypoints automatically from image pixels and performs inference of detection efficiently.

- We conduct experiments to evaluate the performance of our KPNet detector on the COCO benchmark, in which our promising results show that KPNet outperforms all the existing anchor-free detectors, and is able to achieve highly competitive results better or on par with the state-of-the-art two-stage anchor-based detectors on COCO test-dev ($48.3\%$ AP with DCNv2-ResNeXt-101 backbone on COCO test-dev under single-model single-scale settings). Our source code and models will be released publiclly upon acceptance.

The rest of this paper is organized as follows. Section 5.2 reviews two major categories of related work in deep-learning based object detection: popular anchor-based detectors and recent anchor-free detectors. Section 5.3 presents the proposed KPNet detector in detail. Section 5.4 discusses our experimental results and analysis, and Section 5.5 concludes this chapter.

## 5.2 Preliminaries

In this section, we briefly review two major groups of related work in the literature of deep-learning based object detection approaches: the mainstream family of anchor-based object detection methods and the emerging family of anchor-free object detection methods.

### 5.2.1 Anchor-based Object Detection

The methods in this group represent the mainstream detectors widely used in many real-world applications. They can be broadly categorized into two groups: two-stage detectors [162, 48, 47] and one-stage detectors [123, 236, 119]. Two-stage detectors often consist of two stages: (i) region proposal generation, and (ii) regioin proposal classification and regression. For example, one of most popoular two-stage detectors is Faster R-CNN [162] that uses a Region Proposal Network (RPN) to generate regions of interests in the first stage and then send the region proposals down the pipeline for object classification and bounding-box regression. Faster R-CNN has resulted in many various extensions and improvements in literature [118, 95, 254, 25, 24]. Single-stage detectors, also known as single-shot detectors, do not need the proposal generation and simply treat object detection as a simple classification and regression problem by taking an input image and directly learning the class probabilities and bounding box coordinates of objects via convolutional networks. Popular single-stage detectors include YOLO [157], SS-D [123], and RetinaNet [119]. Typically, single-stage detectors are less accurate than two-stage detectors, but are much faster and thus more amenable to real-time inference needs. In general, anchor-based detectors suffer from some critical limitations, including requiring heuristic design of anchors, poor alignment of anchors with ground truth objects, and incurring a large number of false positives when anchors are not carefully designed.

### 5.2.2 Anchor-free Object Detection

From a unified view of keypoint based detectors, we can categorize the existing anchor-free object detection methods into three groups according to different keypoint design and selection strategies: 1) Single Center keypoint, 2) Two Corner keypoints, and 3) Multiple keypoints based detectors. We review some representative works in each group below.

**Single center keypoint:** A representative anchor-free detector is CenterNet [243] that models an object by the center point of its bounding box, and uses keypoint estimation to find center points and regresses to all other object properties, such as size, 3D location, orientation, and even pose.

**Two corner keypoints:** CornerNet [98] models each object by a pair of corner keypoints, which eliminates the need of anchor boxes and is perhaps the first anchor-free detector that achieved the state-of-the-art single-stage object detection accuracy. There was also some extension of CornerNet to improve its efficiency towards real-time applications such as CornerNet-Lite [99].

**Multiple keypoints:** There was another version of CenterNet [29], which models an object by a triplet of keypoints, including one center and two corner keypoints. ExtremeNet [244] models an object by a set of five keypoints, including one center and four extreme points ( (top-most, leftmost, bottom-most, right-most) of an object based on a standard keypoint estimation network. RPDet [225] represents an object by a fixed set of 9 keypoints sampled from the center of an object and can be refined progressively during the training process. In addition, there are also some approaches that sample many keypoints from the center region of an object, such as FSAS [247] and FoveaBox [93]. Finally, some approaches such as FCOS [196] and SAPD [246] treat all the pixels within an object as candidate keypoints during training and improve them by some separate post-processing.

Unlike the above keypoint based detectors that either use predefiend keypoints or sample keypoints from fixed layouts, our approach learns a dynamical set of keypoints automatically from image pixels.

## 5.3 KPNet

### 5.3.1 Overview

We now present the proposed keypoint based detection network (KPNet), a new anchor-free detector that is capable of learning dynamic keypoints with respect to different objects automatically. Unlike many popular keypoint based detectors that use heuristic keypoint design strategies, we argue that the set of keypoints should not be fixed, and the optimal set of keypoints can vary across different types of object. Our main he idea for designing the proposed keypoint based object detector is to minimize the human heuristic design of keypoints and attempt to learn the optimal set of keypoints automatically from data. To this end, instead of assuming any predefined corners or center, we assume any pixel/location of an image could be considered a potential candidate of keypoint, and our goal is to develop an efficient end-to-end learning scheme to find out a compact set of relevant and high-quality keypoints.

Figure 1 gives an overview of the architecture of the proposed KPNet. An input image is passed through a CNN network to produce a set of feature maps. Note that following the CNN backbone, we also apply the Feature Pyramid Network (FPN) that can better handle scale variances. Based on the feature maps, a Fully Convolutional layer is applied to perform the pixel-wise objectness prediction, which predicts the objectness likelihood of a pixel with respect to a particular object category. The objectness map obtained from the objectness prediction module will be used as an input to the keypoins prediction module to predict a dynamic set of high-quality keypoints. Finally, after the final set of dynamic keypoints are obtained, the keypoints's bounding boxes together with their objectness and likelihood scores are passed to NMS to obtain the final detection result.

Next we will discuss several key modules of the proposed framework, including pixel-wise objectness prediction, keypoint prediction, overall detector training, inferences, and other implementation details.

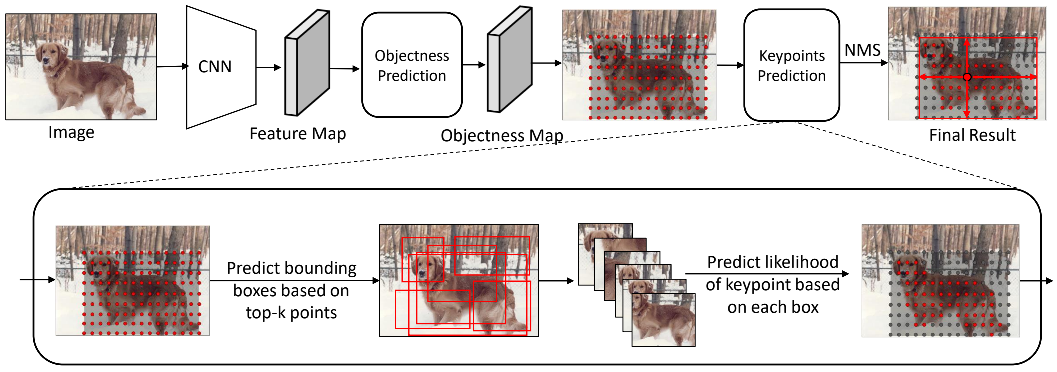


Image
CNN
Feature Map
Objectness Prediction
Objectness Map
Keypoints Prediction
NMS
Final Result
Predict bounding boxes based on top-k points
Predict likelihood of keypoint based on each box

Figure 5.2: The architecture of our KPNet detector. Following the CNN backbone, we also apply the Feature Pyramid Network (FPN). The objectness prediction module is a pixel-wise prediction to predict the likelihood of a relevant object in the location of the pixel. The keypoints prediction module takes the pixel-wise objectness map from the objectness prediction module as the input and learns to predict a dynamic set of high-quality keypoints towards the final detection.

## 5.3.2 Pixel-wise Objectness Prediction

The input to this module is a set of feature maps from a CNN backbone, denoted by $F_i \in \mathbb{R}^{H\times W\times C}$ the feature map at layer $i$ of a CNN with a total of $C$ classes. We view any pixel/location $(x, y)$ of the feature map as a potential candidate for keypoints, and the objectness prediction module aims to predict how likely a particular location of the feature map is relevant to some particular class of objects. This idea follows the principle of fully convolutional networks (FCN) for semantic segmentation and has also been used previously in FCOS [196].

Specifically, we predict the objectness of a particular location $(x, y)$ by a real vector $(\mathbf{c}_{x,y}, \mathbf{b}_{x,y})$, where $\mathbf{c}_{x,y} \in (0, 1)^C$ denotes a $C$-dimensional vector of object class prediction scores for location $(x, y)$, and $\mathbf{b}_{x,y} = (l, t, r, b)$ is a 4D vector representing the bounding box to be regressed at $(x, y)$, namely the left, top, right, bottom distances from the location to the four sides of the bounding box.

### 5.3.3 Keypoint Prediction

The previous objectness scores provide us some useful clue for removing irrelevant pixels/locations with low class objectness scores (e.g., if the max class score of $\mathbf{c}_{x,y}$ is lower than a threshold). However, the objectness score may not be strong enough to choose a small compact set of high-quality keypoints. In this module, we aim to learn a separate keypoint predictor that is able to predict the quality of being a representative keypoint for a particular location. Specifically, we denote by $\mathbf{p}_{x,y} \in (0, 1)$ a keypoint likelihood score that indicates if the location $(x, y)$ is a qualified keypoint for the class $C_j$. We defer the discussion for training the keypoint predictor later.

### 5.3.4 Detector Training

#### Training Losses for Objectness Prediction

We train the objectness prediction using both classification loss and bounding box regression loss. We define $\mathcal{Q} = (x, y)$ is the set of candidate locations that fall into the ground-truth bounding boxes. The classification loss is defined as

$$L_{\text{cls}} = \frac{1}{|\mathcal{Q}|} \sum_{(x,y)} L_{\text{focal}}(\mathbf{c}_{x,y}, \mathbf{c}^*_{x,y}) \tag{5.1}$$

where $L_{\text{focal}}$ is based on the focal loss [119] and $\mathbf{c}^*_{x,y} \in [0, 1]^C$ denotes the ground-truth class labels at location $(x, y)$. For bounding box regression, we only consider a location that falls into any ground-truth box, and we define the regression loss as

$$L_{\text{loc}} = \frac{1}{|\mathcal{Q}|} \sum_{(x,y)\in\mathcal{Q}} L_{\text{GIoU}}(\mathbf{b}_{x,y}, \mathbf{b}^*_{x,y}) \tag{5.2}$$

where $L_{\text{GIoU}}$ is based on the GIoU loss [163], $\mathbf{b}^*_{x,y}$ is a 4D vector $(l, t, r, b)$ representing the ground-truth bounding box of class $i$ with respect to the location $(x, y)$.

### Training Loss for Keypoint Prediction

Consider a location $(x, y)$ of a feature map, our goal is to train a model to predict $p_{x,y} \in (0, 1)$ that if this location is a high-quality representative keypoint. From the previous objectness prediction module, given $(x, y)$, we can predict both its class objectness scores $\mathbf{c}_{x,y}$and its potential bounding box $\mathbf{b}_{x,y} = (l, t, r, b)$. Instead of using the original feature map $F_i$ at location $(x, y)$ to predict the keypoint score, we propose to use the features from the predicted bounding box (extracted by an ROI encoding method) as the input feature for prediction, namely

$$p_{x,y} = \text{softmax}(\text{ROI-encoding}(F_i, \mathbf{b}_{x,y})) \tag{5.3}$$

where we adopt the RoiAlign [60] for ROI-encoding. Now we discuss how to compute the ground-truth label $p^*_{x,y}$ from training data automatically.

Based on the ground-truth bounding box, we can define if a location $(x, y)$ is qualified as a representative keypoint if its predicted bounding box $\mathbf{b}_{x,y}$ has a sufficient overlap with the ground-truth bounding box at the current location (note that if a location falls into multiple bounding boxes, we simply take the bounding box with minimal area as the target bounding box). More specifically, we measure the overalp score by computing the Intersection over Union (IoU) between the predicted box and the ground-truth box, and use it to define the ground-truth labels for keypoint training. Namely

$$p^*_{x,y} = \mathbb{I}\big(\text{IoU}(\mathbf{b}_{x,y}, \mathbf{b}^*_{x,y}), t_{\text{IoU}}\big) \tag{5.4}$$

where $t_{\text{IoU}}$ is an IoU threshold (which will be changed adaptively during training, to be discussed later), and $\mathbb{I}(a, b)$ is an indicator function that output 1 when $a \geq b$ and 0 otherwise.

During keypoint prediction training, we only consider the set of locations that fall into any ground-truth bounding box in the training data. Specifically, we can then define the training loss for keypoint prediction with respect to an IoU threshold $t_{\text{IoU}}$ as follows:

$$L_{\text{key}}(t_{\text{IoU}}) = \frac{1}{|\mathcal{Q}|} \sum_{(x,y)\in\mathcal{Q}} L_{\text{ce}}(p_{x,y}, p^*_{x,y}) \tag{5.5}$$

where $L_{\text{ce}}$ denotes the cross-entropy loss and $p^*_{x,y}$ is defined in (4) for a given IoU threshold.

### Overall Training Loss and Adaptive Keypoint Training

Combining the above training losses for both objectness prediction and keypoint prediction modules, we can define the overall training loss function as follows:

$$L_{\text{overall}} = L_{\text{key}}(t_{\text{IoU}}) + \lambda_{\text{cls}} L_{\text{cls}} + \lambda_{\text{loc}} L_{\text{loc}} \tag{5.6}$$

where both $\lambda_{\text{cls}}$ and $\lambda_{\text{loc}}$ are simply set to 1 during our training.

During training, $t_{\text{IoU}}$ is a key parameter used to control the quality of points to become representative keypoints and will be changed adaptively during our training process. Specifically, in our current approach, we start with an initial value $t_{\text{IoU}} = 0.3$ and gradually increase this threshold in our training process. When increasing $t_{\text{IoU}}$, we essentially impose a higher quality constraint on the selection of keypoints, which will result in a more compact set of high-quality keypoints.

## 5.3.5 Inference

During the inference stage, the KPNet detector takes an image as an input and passes it through the CNN network followed by the FPN module to obtain the feature maps. Based on the resulting feature maps, we then predict the objectness scores $\mathbf{c}_{x,y}$ and bounding box offsets $\mathbf{b}_{x,y}$ of each point/location $(x, y)$ on the feature map. After that, we select top k points/locations with the highest class objectness scores as the candidate points to the keypoint prediction module. For each candidate keypoint, we compute the region features of the boxes using the RoiAlign enchoding method, and use it to predict the quality scores of being a representative keypoint. Finally, based on the compact set of keypoints, we combine their quality scores with the class objectness scores of the keypoint, and pass them to a soft-version of Non-Maximum Suppression (NMS) to obtain the final detection result.

### 5.3.6 Implementation Details

#### Network Backbones

Following the recent state-of-the-art object detectors such as FCOS [196] and RetinaNet [119], we adopt the ResNet [62] and ResNeXt [215] CNN network as our backbone architecture. ResNet and ResNeXt are two fully convolutional networks, which are composed of a sequence of residual modules and were first used for image classification. Residual module first encodes the input feature by a sequence of convolution and normalization layers, and then aggregates the generated feature map with the original input features. In order to predict objects with large scale variance, we also apply the Feature Pyramid Network (FPN) [118] in our approach, which combines the shallow layer features with deep layer features by the latent connection. To learn a scale-robust detector, each level of FPN is responsible for a certain scale of objects, making it very suitable for object detection. Specifically, we use 5 FPN levels to make prediction, with stride 8, 16, 32, 64 and 128 compared with the original image, and each of the level is responsible for a certain scale of the objects: (0, 64], (64, 128], (128, 256], (256, 512] and (512, INF]. We adopt ResNet-50 [62], ResNet-101 [62] and DCN2-ResNeXt-101-64x4 [216] as our backbone architecture in our experiments.

**Initialization and Optimization**

We train the model from weights pre-trained on ImageNet classification task and other parameters are initialized by the same methods as RetinaNet [119]. The model is trained with SGD optimization methods with 180k iterations with 16 images per mini-batch. The initial learning rate is set to 1e-2 and is reduced 10 times at 120k and 160k iterations. During our training, we adaptively change the IoU threshold parameter $t_{\text{IoU}}$. Specifically, we first train the model for 60k iterations with an initial IoU threshold $t_{\text{IoU}}$ at 0.3, then increase it to 0.5 for training another 60k iterations, and finally increase it to 0.7 for another 10 iterations of training. We re-scale the the input images into 800x1333 pixels before training. We use the same data augmentation strategy presented in [196] when training the model, and for each image, the top-50 predictions are produced.

## 5.4 Experiments

### 5.4.1 Experimental Dataset and Setup

We conducted experiments on MSCOCO dataset, which has 80 categories in three splits: train (115k images), val (5k images), and test-dev (20k images). Following common practice, we used the train set to train our model and the val set for ablation studies, and finally report the results on test-dev set for comparison. In our experiments, only bounding box level annotations are used. We consider four types of backbones: ResNet-50 [62], ResNet-101 [62] and ResNeXt-101-DCNv2[216]. For efficiency, ResNet-50 and ResNet-101 is used in our ablation study.

### 5.4.2 Experimental Results

Table 5.1 shows the results on the COCO val set by comparing our method with other popular anchor-free detectors mostly with ResNet-101 (for CornerNet and CenterNet we add results on Hourglass-104 since they are designed based on Hourglass backbones). For comparison purposes, we also include two anchor-based detectors, including the popular RetinaNet[119] and the state-of-the-art ATSS [235] .

| Object Detectors | Anchor-free | Backbone | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|---|
| RetinaNet[119] | Anchor-based | R-101 | 39.1 | 59.1 | 42.3 | 21.8 | 42.7 | 50.2 |
| ATSS [235] | Anchor-based | R-101 | 43.6 | 62.1 | 47.4 | 26.1 | 47.0 | 53.6 |
| CornerNet[98] | two corners | R-101 | 30.2 | 44.1 | 32.0 | 13.3 | 33.3 | 42.7 |
| CornerNet[98] | two corners | HG-104 | 40.5 | 56.5 | 43.1 | 19.4 | 42.7 | 53.9 |
| CenterNet[243] | one center | R-101 | 34.6 | 53.0 | 36.9 | - | - | - |
| CenterNet[243] | one center | HG-104 | 42.1 | 61.1 | 45.9 | 24.1 | 45.5 | 52.8 |
| FSAF[247] | multiple keypoints | R-101 | 40.9 | 61.5 | 44.0 | 24.0 | 44.2 | 51.3 |
| FoveaBox[93] | multiple keypoints | R-101 | 40.6 | 60.1 | 43.5 | 23.3 | 45.2 | 54.5 |
| FCOS [196] | all pixels/locations | R-101 | 41.5 | 60.7 | 45.0 | 24.4 | 44.8 | 51.6 |
| RPDet [225] | multiple keypoints | R-101 | 41.0 | 62.9 | 44.3 | 23.6 | 44.1 | 51.7 |
| KPNet (ours) | multiple keypoints | R-101 | **44.0** | 62.5 | 46.9 | 26.6 | 47.3 | 53.6 |

Table 5.1: Performance evaluation of popular keypoint based detectors and two anchor-baesd detectors. The models are trained on MSCOCO train with 115k images, and validated on MSCOCO val set with 5k images. “R-101” denotes ResNet-101 backbone and “HG-104” denotes Hourglass-104 backbone.

From the results, we can see that all anchor-free/keypoint-based methods outperform RetinaNet which uses predefined anchors and IoU matching methods. This confirms the advantage of keypoint-based detection methods over heuristic anchor-based designs. However, the existing anchor-free detectors are worse than ATSS, which is a recent state-of-the-art anchor-based method by borrowing and adapting some advanced strategies from anchor-free methods. By examining the results of our KPNet, we found that it outperforms all the existing keypoint-based detectors. This is mainly because all these methods are based on either manual keypoint design and fixed keypoint sampling strategy. By contrast, our method is capable of learning dynamical set of keypoints automatically to significantly boost the performance of keypiont-based detectors. Finally, our method is also better than the state-of-the-art anchor-based ATSS, but eliminates the need of manually designed anchors.

### 5.4.3 Ablation Study

This ablation study aims to examine if our KPNet with automated keypoint learning is able to outperform a variety of keypoint design strategies using predefined layouts. Table 5.2 shows the results of our ablation study and Table 3 illustrates

| #Keypoints | Backbone | AP | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|---|
| FASF | R-50 | 35.9 | 55.0 | 37.9 |
| FCOS | R-50 | 37.8 | 55.6 | 40.7 |
| 1-C | R-50 | 34.1 | 52.8 | 38.8 |
| 4-C | R-50 | 37.9 | 56.0 | 41.0 |
| 9-C | R-50 | 38.6 | 57.4 | 41.4 |
| 17-C | R-50 | 38.5 | 57.3 | 41.4 |
| 8-S | R-50 | 35.3 | 54.1 | 37.8 |
| 16-S | R-50 | 35.0 | 53.8 | 37.6 |
| 1-C+8-S | R-50 | 37.2 | 55.0 | 39.6 |
| 4-C+8-S | R-50 | 38.5 | 56.9 | 41.3 |
| 9-C+16-S | R-50 | 39.3 | 57.5 | 42.1 |
| 9-C+8-S | R-50 | 39.5 | 57.7 | 42.2 |
| KPNet (ours) | R-50 | **40.5** | 58.8 | 42.9 |

Table 5.2: Ablation study on different keypoint sampling strategies. ”S” denotes keypoints from surrounding regions and ”C” denotes keypoints from central region. Models are trained on COCO train2017 and tested on COCO val2017 with ResNet-50.

some example configurations (these baselines follow the similar pipeline of FCOS). From the results in Table 5.2, we can see that our KPNet with the dynamic keypoint auto-learning strategy outperforms all kinds of fixed keypoint designs strategies.

### 5.4.4 Comparison with State-of-the-art Detectors

We now compare our KPNet with other state-of-the-art detectors on COCO test-dev set. Unlike the previous experiments, we train our models on two backbones ResNet-101 and ResNeXt-101-DCNv2. Table 5.3 shows the results on COCO test-dev under the single-model single-scale settings. Our KPNet outperforms all the the one-stage detectors in literature by a substantial margin (except ATSS), also outperform a variety of two-stage/multi-stage detectors, and achieves the best results among all the keypoint based detectors. This promising result validates our hypothesis of automatically learned keypoints is essential to capture more discriminative information of objects for improving the results. Specifically, compared with other center-based methods such as CenterNet or FSFA, our method not only extracts the feature from the central region, but also encodes the structure information

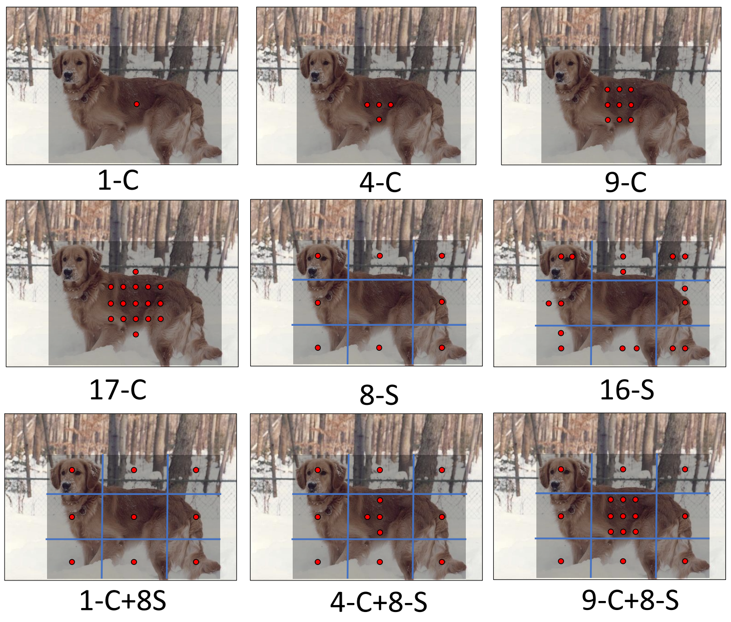


Figure 5.3: Example configurations. Red dots denote keypoints we aim to learn to predict. ”C” denotes keypoints from central region and ”S” denotes keypoints from surrounding regions.

from the entire bounding boxes. Compared with other methods using multiple keypoints, our detector has better results due to the optimization of the dynamic set of keypoints instead of heuristic sampling. Finally, compared with the SOTA anchor-based ATSS, our method eliminates the need of using anchors and thus avoids the heuristic anchor design.

## 5.5 Discussion

In this chapter, we presented a unified view of the popular anchor-free object detectors from the keypoint based detection perspective. We argue that the existing keypoint based detectors that often use heuristic keypoint designs or fixed keypoint sampling strategies may not fully exploit the potential of keypoint based detectors. To overcome the limitation, we proposed KPNet, a new keypoint based detector which is able to learn a compact set of representative keypoints automatically from data without manual design. Our new detector shows significant improvement over existing anchor-based and anchor-free methods, especially in high-quality settings,

| Method | Backbone | FPS | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|---|
| **Anchor-based** | | | | | | | | |
| **Multi-stage** | | | | | | | | |
| FRCN-FPN [118] | R-101 | 9.9 | 36.2 | 59.1 | 39.0 | 18.2 | 39.0 | 48.2 |
| Cascade R-CNN[10] | R-101 | 8.0 | 42.8 | 62.1 | 46.3 | 23.7 | 45.5 | 55.2 |
| Libra R-CNN [145] | X-101-64x4d | 5.6 | 43.0 | 64.0 | 47.0 | 25.3 | 45.6 | 54.6 |
| TridentNet [110] | R-101-DCN | 1.3 | 46.8 | 67.6 | 51.5 | 28.0 | 51.2 | 60.5 |
| FreeAnchor [238] | X-101-32x8d | 5.4 | 44.8 | 64.3 | 48.4 | 27.0 | 47.9 | 56.0 |
| Fitness-NMS [198] | R-101 | 5.0 | 41.8 | 60.9 | 44.9 | 21.5 | 45.0 | 57.5 |
| **Single-stage** | | | | | | | | |
| RefineDet[236] | R-101 | - | 36.4 | 57.5 | 39.5 | 16.6 | 39.9 | 51.4 |
| RetinaNet [119] | R-101 | 8.0 | 39.1 | 59.1 | 42.3 | 21.8 | 42.7 | 50.2 |
| AB+FSAF [247] | R-101 | 7.1 | 40.9 | 61.5 | 44.0 | 24.0 | 44.2 | 51.3 |
| AB+FSAF [247] | X-101-64x4d | 4.2 | 42.9 | 63.8 | 46.3 | 26.6 | 46.2 | 52.7 |
| M2Det [240] | VGG-16 | 11.8 | 41.0 | 59.7 | 45.0 | 22.1 | 46.5 | 53.8 |
| ATSS [235] | X-101-64x4d-DCNv2 | 7.1 | 47.7 | 66.5 | 51.9 | 29.7 | 50.8 | 59.4 |
| **Anchor-free** | | | | | | | | |
| GA-FRCN [206] | R-50 | 9.4 | 39.8 | 59.2 | 43.5 | 21.8 | 42.6 | 50.7 |
| GA-RetinaNet [206] | R-50 | 10.8 | 37.1 | 56.9 | 40.0 | 20.1 | 40.1 | 48.0 |
| CornerNet [98] | HG-104 | 3.1 | 40.5 | 56.5 | 43.1 | 19.4 | 42.7 | 53.9 |
| ExtremeNet [244] | HG-104 | 2.8 | 40.2 | 55.5 | 43.2 | 20.4 | 43.2 | 53.1 |
| FoveaBox [93] | R-101 | 11.2 | 40.6 | 60.1 | 43.5 | 23.3 | 45.2 | 54.5 |
| FoveaBox [93] | X-101 | - | 42.1 | 61.9 | 45.2 | 24.9 | 46.8 | 55.6 |
| FCOS [196] | R-101 | 9.3 | 41.5 | 60.7 | 45.0 | 24.4 | 44.8 | 51.6 |
| FCOS w/ imprv[196] | X-101-64x4d | 5.4 | 44.7 | 64.1 | 48.4 | 27.6 | 47.5 | 55.6 |
| CenterNet [243] | HG-104 | 7.8 | 42.1 | 61.1 | 45.9 | 24.1 | 45.5 | 52.8 |
| CenterNet [29] | HG-104 | 3.3 | 44.9 | 62.4 | 48.1 | 25.6 | 47.4 | 57.4 |
| RPDet [225] | R-101-DCN | 8.0 | 45.0 | 66.1 | 49.0 | 26.6 | 48.6 | 57.5 |
| SAPD [246] | X-101-64x4d-DCN | 4.5 | 47.4 | 67.4 | 51.1 | 28.1 | 50.3 | 61.5 |
| **KPNet** (ours) | R-101 | 8.3 | **44.0** | 62.5 | 46.9 | 26.6 | 47.3 | 53.6 |
| **KPNet** (ours) | X-101-64x4d-DCNv2 | 5.9 | **48.3** | 67.1 | 52.3 | 30.0 | 51.7 | 59.5 |

Table 5.3: Comparison of KPNet with other state-of-the-art two-stage or one-stage detectors (*single-model and single-scale results*). All models were trained on CO-CO train set and tested on test-dev.

and has obtained the state-of-the-art on the MS COCO test-dev set (under single-model single-scale settings).

# Part III

# Few-shot Detection

# Chapter 6

# Meta Learning for Few-shot Object Detection

## 6.1 Introduction

Following the success of deep learning for image classification [86, 96], recent years have witnessed remarkable progress in object detection with deep learning. A series of detection algorithms based on DCNNs have been proposed which achieve state-of-the-art results on public detection benchmark datasets [45, 48, 162, 118, 119, 123, 158]. However, all these methods are data hungry, and require large amounts of annotated data to learn an immense number of parameters. For object detection, annotating the data is very expensive (far more than image classification), as it requires not only identifying the categorical labels for every object in the image, but also providing accurate localization information through bounding box coordinates. Moreover, in real-world applications, such as medical research, it's often impossible to even collect sufficient data to annotate. This warrants a need for effective detectors that can generalize well from small amounts of annotated data. We refer to this real-world problem of learning detectors from limited labeled data as *few-shot detection*. For example, in *one-shot* detection, only one image is available with objects of interest annotated, and a detector needs to

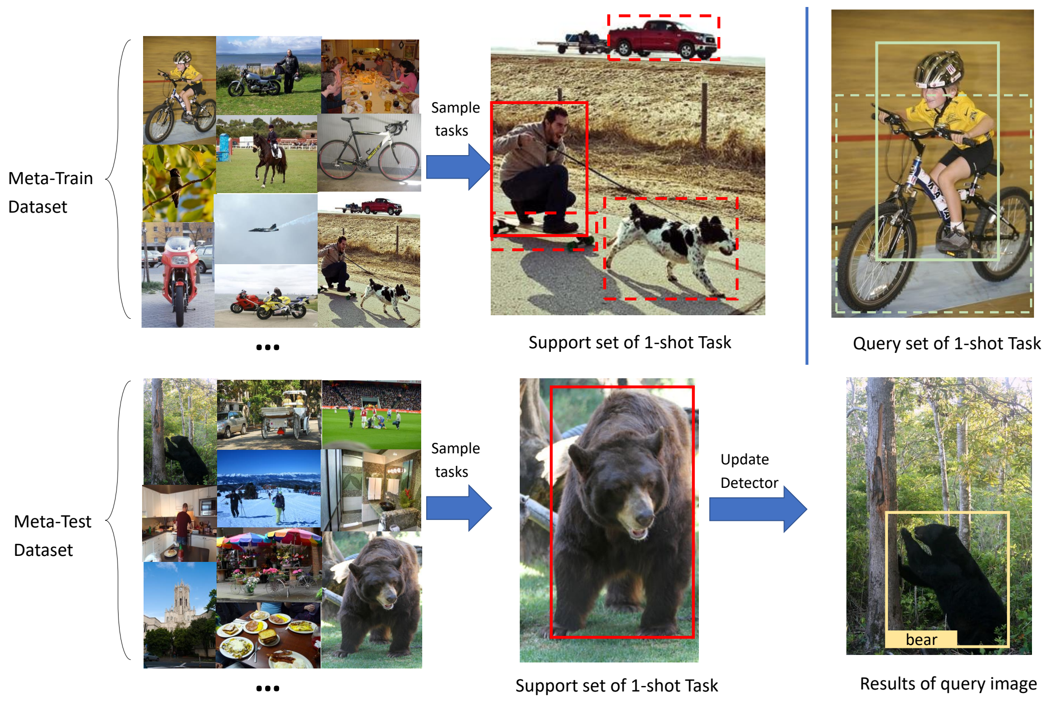


Figure 6.1: Few-shot object detection in the meta-learning setting. From the meta-train dataset, a $K$way-$N$shot support set and a query set are sampled to create a task. The meta detector makes predictions on the query set by using the knowledge from the support set, and updates the detector based on the loss on the query set. In this example, despite many objects (“person”, “dog”, “truck”, etc), the meta-train sample task aims to just detect “person”. At test time, a single annotated image from a novel class (e.g., “bear”) is available for the detector to learn a model that can generalize.

train on just this image and generalize. When presented with such small amounts of annotated data, traditional detectors tend to suffer from overfitting. Inspired by the fact that humans can learn a new concept from limited training data, we aim to develop a new few-shot detection method.

Recent years have seen active efforts for few-shot learning [201, 36, 184]. Many of them follow the principle of meta learning, where a set of tasks in a few-shot setting is simulated from a large corpus of annotated data, and the model is optimized to perform well over these few shot tasks. This trains the model to learn *how* to solve few-shot tasks. However, most existing efforts of meta learning are mainly focused on classification. Adapting few-shot classification algorithms directly for few-shot detection (e.g. by replacing the region classification branch of detector with a meta-learner) is non-trivial because of two major concerns: i). Detection algorithms not only require classifying objects but also need to correctly localize objects in cluttered backgrounds by using a Region Proposal Network (RPN) and bounding box (bbox) regressors. It is thus also desirable that both RPN and bbox regressors should also be capable enough to adapt to few-shot settings. ii). For a given task with one (or few) annotated image(s), the annotated image may contain objects from several classes. But only a few objects of interest are annotated. The goal of the few-shot detector is to detect only these objects of interest. Unfortunately, a naively trained meta-detector's RPN would detect all objects (even objects from classes not of interest) and try to classify them as one of the classes of interest rather than background images (See Figure 6.1 for an example).

We aim to address these challenges in few-shot object detection by proposing a novel method using the meta-learning paradigm. In particular, we propose Meta-RCNN, an end to end trainable meta object detector, which follows the episodic learning paradigm of meta-learning [201], where multiple few-shot tasks are simulated based on a give meta-train dataset. Specifically, for a given task, we first construct a class prototype for each of the annotated object categories in the support set. Using these prototypes, a class-specific feature map of the entire image is constructed, i.e., we obtain a feature map of the entire image for each of the class prototypes. These feature maps are tailored to detect only objects of the class of the prototype, by giving higher attention to appropriate regions in the image containing that object. A weight-shared RPN follows the class-aware feature map to generate proposals. For each generated proposal, we concatenate the feature of its corresponding prototype to further enhance its representation ability. This is followed by a binary classifier and a bbox regressor.

Meta-RCNN learns a few-shot detector where the whole framework can be trained via meta-learning in an end-to-end manner. In contrast to the naive adaptation of meta-learning for classification into an object detection framework, Meta-RCNN learns the few-shot classifier, the RPN, and the bbox regressor in the meta-learning setting, thus making all three components suitable for handling few-shot scenarios. Moreover, Meta-RCNN learns a class-specific feature map for a given class prototype enabling easier distinction between classes of interest and backgrounds (where other objects in the image from classes not of interest are considered as backgrounds). We demonstrate the effectiveness of Meta-RCNN on the popular few-shot detection benchmarks: Pascal VOC, and show that Meta-RCNN significantly improves the detection result in few shot settings.

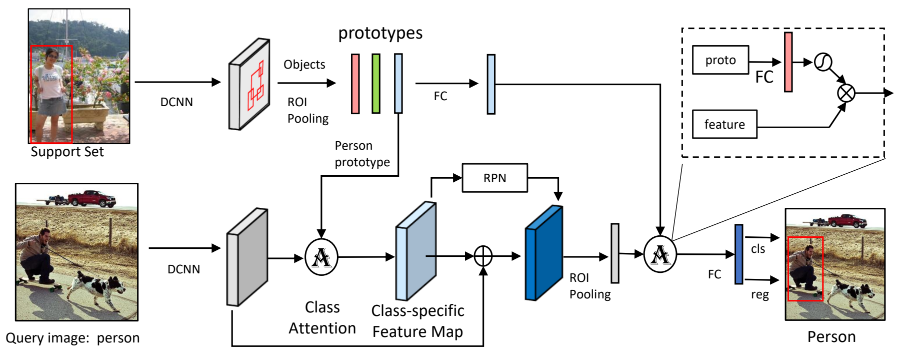


Figure 6.2: The Meta-RCNN workflow. A set of prototypes of different categories are extracted from the support set. For each class, conditioned on these prototypes, a class-specific feature map from query set is generated by applying the class attention module to the feature map of the entire image. The new class-specific feature map is tailored to detecting objects of that specific class (class 'person' in the workflow, etc.). An weight-shared RPN is applied in the class-specific feature map, followed with an binary region classification layer and bounding box regressor. The whole network is optimized via meta learning and can be trained end-to-end.

## 6.2 Preliminaries

### 6.2.1 Problem Setting

We now present the formal problem setting of few-shot detection in this Chapter. Consider two datasets $L$ and $S$, where $L$ is a large-scale annotated dataset with $L_c$ categories and $S$ is a dataset with only a few annotated images with $S_c$ categories. There is no category overlap between two datasets: $L_c \cap S_c = \phi$. Our goal is to learn a robust detector from the annotated data in $L$ and $S$ to detect unlabeled images in $S$.

The proposed Meta-RCNN aims to learn a general detection framework which can be quickly adapted to detection tasks with only a few labeled samples. We follow the standard training scheme of meta learning, which splits the whole learning stage into two parts: meta-training and meta-testing, and the model is optimized over multiple few-shot tasks simulated from the meta-training data. Specifically, during meta-training, few-shot detection tasks are sampled from $L$, and each task contains a support set and a query set. For the $i$-th task, $K$ ways (or categories) and $N$ images per category are randomly selected from $L_c$ to build support set: $\mathrm{T}_i^{\mathrm{L,s}}$. Similarly, $Q$ images per category are randomly selected to build query set $\mathrm{T}_i^{\mathrm{L,q}}$. Support set $\mathrm{T}_i^{\mathrm{L,s}}$ and query set $\mathrm{T}_i^{\mathrm{L,q}}$ construct a complete task extracted from $L$ (See Figure 6.1):

$$\mathrm{T}_i^{L} = \left\{\mathrm{T}_i^{\mathrm{L,s}}, \mathrm{T}_i^{\mathrm{L,q}}\right\} \tag{6.1}$$

where both the support set and query set are used to train the meta-model. The meta-model optimizes the base-model with respect to the support set and makes predictions on query set. Finally the loss suffered on the query set is used to update the model. In the meta-testing stage, similar to meta-training stage, a set of few-shot tasks are sampled from $S$:

$$\mathrm{T}_i^{S} = \left\{\mathrm{T}_i^{\mathrm{S,s}}, \mathrm{T}_i^{\mathrm{S,q}}\right\} \tag{6.2}$$

where $\mathrm{T}_i^{\mathrm{S,s}}$ is support set and $\mathrm{T}_i^{\mathrm{S,q}}$ is query set. The model makes predictions on the query set, and these results are averaged across several few-shot tasks to evaluate the expected performance of the few-shot detector over a variety of novel few-shot detection tasks.

### 6.2.2 Overview of Faster RCNN

Meta-RCNN is based on two-stage region based object detection algorithms. In this work, we use the popular Faster RCNN algorithm [162] as our base model. Faster RCNN consists of two components, an RPN (Region Proposal Network) for proposal generation and Fast RCNN for region classification. RPN generates a sparse set of proposals which are classified into different categories by the region classifiers. Specifically, RPN extracts a feature vector from each region by scanning the whole image using sliding windows. This is followed by a binary classifier (objects vs backgrounds) and a bounding box regressor, where easy negatives are filtered. For each proposal, a fixed-length feature vector is extracted by using ROI Pooling layers. This vector is then fed into a sequence of dense connected layers branching into two outputs. One output is responsible for representing softmax probability over $K + 1$ classes($K$ target classes and one background class), and the other one encodes four real-values for refining bounding box position. We denoted $u$ and $v$ as the category and bounding box label respectively, $p$ as the predicted probability distribution over C classes, and $t_u$ as the predicted bounding box prediction of class $u$, and $\lambda$ as the trade-off parameter. $L_{\text{cls}}$ represents softmax loss and $L_{\text{loc}}$ represents SmoothL1 loss function. The entire network can be optimized end-to-end by minimizing loss $L(p, u, t^u, v)$:

$$L(p, u, t^u, v) = L_{\text{cls}}(p, u) + \lambda[u \geq 1]L_{\text{loc}}(t^u, v), \tag{6.3}$$

However, two-stage detectors require a lot of training samples to obtain a good performance. In the next section, we present the proposed Meta-RCNN which builds over Faster RCNN and is specifically designed to address few-shot detection.

## 6.3 Meta-RCNN

### 6.3.1 Overview

We now present our proposed method Meta-RCNN for few-shot detection (See Figure 6.2 for an overview). Meta-RCNN is trained with multiple few-shot tasks simulated from the meta-train dataset. For each episode, a few object categories of interest are assumed to be annotated (Support set). During meta-training, a prototype is computed for each object category. For each of these category prototypes, a class-specific feature map is generated by using a class-attention module which combines the prototype information with the feature map of the entire image. This feature map only highlights the signals of the class of interest, and suppresses information from other classes. An weight-shared RPN is applied into these feature maps followed with a binary region classifier and bbox regressor to make prediction. Based on the loss on the query set, the model is updated.

Meta-RCNN is general paradigm to train few-shot detector via by meta-learning. For each task, irrelevant categories and background can be filtered by attention module, and the final generated feature map learns a general representation for the given few-shot detection task. Compared with [170] and [218], Meta-RCNN is more general and the whole framework can be optimized including RPN and bbox regressors, making all the components few-shot capable. Next, we present the details of the model.

### 6.3.2 Meta-Training

During Meta-Training, multiple $K$way-$N$shot tasks are simulated from the annotated dataset $L$. To fit memory size, in Meta-Training stage we train the model using 5way-1shot tasks, and only 5 query images (1 query image per class), which results in only a total of 10 images for each task. For each task $T_i^L$, images of support set $T_i^{L,s}$ are fed into Faster RCNN to generate region features. For each of the object categories of interest (those assumed to be annotated in the support image), a prototype $P_c$ is generated based on the corresponding region features:

$$P_c = \frac{1}{N_c} \sum_{i}^{N_c} r_c^i \tag{6.4}$$

where $P_c$ denotes prototype of class $c$, and $r_c^i$ denotes $i$-th region features of all annotated objects from class $c$. Based on these generated prototypes, images of query set $T_i^{L,q}$ are fed into the same Faster RCNN model and we obtain the image feature map before RPN and ROI Pooling. For each category, a class-specific feature map is learned based on the input query image and its corresponding prototype. We use a learnable class attention module here to highlight the signals of target class and suppress signals of other categories. The class attention module is based on basic channel-wise multiplication. The prototype $P_c$ is encoded by a FC layer $\phi$, which is later combined with feature map $f$ by element-wise multiplication: $F_c = f \odot \phi(P_c)$

For each category $c$, one new feature map $F_c$ is generated which aims to highlight the objects of class $c$. Based on the new feature map $F_c$, a weight-shared RPN is followed to produce region proposals. In order to recover the information lost in attention module, we finally combine the new generated feature map with original feature map by element-wise summation, and crop region features based on the new generated map. To further enhance the representation of region proposals, we attach the prototype with the region feature $r$: $R_c = r \odot \phi(P_c)$

Finally, a binary region classifier and a bbox regressors are optimized w.r.t the label info from query set $T_i^{L,q}$:

$$L(T_i^{L,q}; T_i^{L,s}, \theta) = L_{\text{loc}} + L_{\text{cls}} + L_{\text{RPN}} \tag{6.5}$$

where $\theta$ represents the parameters of Meta-RCNN.

| DATASET | Train | #Img | #cls | Test | #Img | #cls |
|---|---|---|---|---|---|---|
| **FSOD-VOC** | VOC2007trainval | $\sim$ 4.9k | 10 | VOC2007test | $\sim$ 2.2k | 10 |
| **FSOD-IMAGENET** | ImageNet-LOC | $\sim$ 53k | 100 | ImageNet-LOC | $\sim$ 117k | 214 |

Table 6.1: Two Few-Shot Object Detection (FSOD) benchmark testbeds for performance evaluation

| **Method** | Backbone | 5way-1shot | 5way-3shot | 5way-5shot |
|---|---|---|---|---|
| vanilla FRCN [162] | VGG16 | 14.78% $\pm$ 1.02% | 20.34% $\pm$ 1.26% | 26.89% $\pm$ 1.23% |
| LSTD [13] | VGG16 | 17.66% $\pm$ 1.65% | 22.37% $\pm$ 0.81% | 29.00% $\pm$ 1.28% |
| FRCN-PN | VGG16 | 12.71% $\pm$ 0.70% | 13.91% $\pm$ 0.70% | 14.33% $\pm$ 0.61% |
| FRCN-PN (Finetuned.) | VGG16 | 16.48% $\pm$ 1.04% | 21.51% $\pm$ 0.98% | 26.01% $\pm$ 1.03% |
| Meta-RCNN (ours) | VGG16 | **19.22%** $\pm$ **1.01%** | **24.45%** $\pm$ **1.20%** | **31.11%** $\pm$ **0.88%** |

Table 6.2: mAP Performance Evaluation on the FSOD-VOC BENCHMARK

### 6.3.3 Meta-Testing

During meta-testing, we sample few-shot detection tasks from $S$. The annotations of support set are available and we make predictions on the query set to evaluate the performance of Meta-RCNN. For each task $T_i^S$, prototypes are generated from support set $T_i^{S,q}$, which are later used to generate new class-specific feature maps of images from query set $T_i^{S,q}$. In this stage, we need to finetune the model based on the labeled images of support set. The finetuning operation addresses the learning limitation of non-parametric method when more labeled images are provided. Finally, we evaluate the output from the query set as traditional detection problem:

$$p, u = \text{MetaRCNN}(T_i^{S,q}; T_i^{S,s}, \theta) \tag{6.6}$$

where $p$ is class probability vector and $u$ is location set of bounding boxes.

## 6.4 Experiments

### 6.4.1 Datasets and Implementation Details

**Datasets:** We construct two benchmark testbeds to facilitate the performance evaluation of few-shot object detection (FSOD) in meta-learning settings: (i) FSOD-VOC based on Pascal VOC2007 and (ii) FSOD-ImageNet based on the animal subset of ImageNet-LOC dataset. Table 6.1 gives a summary of the datasets. Pascal VOC2007 has 20 categories with 5k images in trainval set and 5k images in test set. A subset of 10 categories are randomly selected from VOC2007 trainval set for Meta-Training and the remaining 10-category subset of VOC2007 test set is used for Meta-Testing. Images without target object categories are excluded in Meta-Testing. For FSOD-ImageNet benchmark, we use the subset of first 100 animal classes of ImageNet in Meta-Training stage and the subset of remaining 214 animal species in ImageNet-LOC in Meta-Testing stage. The model used in FSOD-VOC benchmark is pre-trained on ImageNet, while in FSOD-ImageNet benchmark, the model is pre-trained on MSCOCO dataset with 115k images in 80 categories.

**Task Generation:** For each benchmark, Meta-RCNN is evaluated on multiple tasks with different $K$way-$N$shot few-shot settings ($N$ annotated images per category). For FSOD-VOC benchmark, we have 3 few-shot settings to evaluate Meta-RCNN: 5way-1shot, 5way-3shot and 5way-5shot. In detection, a single image has more than one object, and proposal generation will automatically increase the number of training samples, so the real number of training samples is about 5 times larger than $N$. On FSOD-ImageNet benchmark, we mainly follow [13] and [170] with two settings: 50way-1shot and 50way-5shot.

**Meta-model Parameter Setting:** In Meta-Training stage, we totally finetune the model for 20 epochs and 10 epochs in FSOD-VOC benchmark and FSOD-ImageNet benchmark respectively. There are 5 images per class in query set to update the model weights. The initial learning rate is set to 1e-3 and is reduced to 1e-4 every 5 epochs. We set the batch size as 5 during query update.

**Basic Detection Parameter Settings:** The parameter settings for Meta-RCNN are identical to vanilla Faster RCNN. Proposals overlap with objects higher than 0.5 are considered positive and less than 0.3 are negative. During Meta-Training the top 128 most confident proposals are selected for training, and 300 proposals with highest confidence score are selected during evaluation. We build our Meta-RCNN based on Faster RCNN with VGG16 [181] and ResNet101 [62] model which is pretrained on ImageNet.

**Model Evaluation:** We evaluate Meta-RCNN based on multiple tasks of few-shot settings, which follows the evaluation metric of standard meta learning settings. Specifically, in the evaluation stage, 200 $K$shot-$N$shot tasks are sampled from dataset $S$ and images in the query set will be evaluated. The mean Average Precision (mAP) over the selected $K$ categories is used as the performance evaluation score.

### 6.4.2 Results on FSOD-VOC Benchmark

We evaluate our Meta-RCNN on FSOD-VOC benchmark where a subset of 10 Pascal VOC categories are selected for Meta-Training and another 10 categories are used for Meta-Testing. For a fair comparison, these two subsets are split as similar as possible. For example, we keep animal categories on both sides since they share similar semantics information (see appendix for details). We implement three baselines on FSOD-VOC to compare with the proposed Meta-RCNN.

- **vanilla FRCN** [162]: the vanilla Faster RCNN which is the most popular object detection algorithm with competitive performance on many benchmarks. The vanilla FRCN is not designed for few-shot detection problem, but we try to include this baseline by fine-tuning the detector on the few-shot training data.
- **LSTD** [13] is a few-shot detection algorithm based on Faster RCNN. LSTD uses categorical regularization to transfer knowledge from $L$ to $S$.
- **FRCN-PN** is a simple baseline for few-shot object detection using meta learning, which combines Faster RCNN and Prototype Networks [184]. Specifi-

cally, it replaces the final FC layer of Faster RCNN by the non-parametric prototypical networks.

All the above baselines including the proposed Meta-RCNN are based on VGG16 backbone [181]. For regular FRCN and LSTD, we first train a global Faster RCNN during Meta-Training, and then the pretrained detectors are adapted to different tasks during Meta-Testing. During Meta-Testing, Meta-RCNN and vanilla FRCN are finetuned over 4 epochs while LSTD requires longer finetuning period (10 epochs). For FRCN-PN, prototypes of different categories are extracted as Meta-RCNN, and metric distances are learned to assign correct labels to each proposal. For fair comparison, we also add one baseline of finetuning FRCN-PN in meta-testing stage, where the images of support set are also used as query images. Table 6.2 shows the results on three settings.

From Table 6.2, the performances of all four methods improve with training shot increasing. Notably, FRCN-PN obtains less improvement when shot increases because the non-parametric property of PN layer limits its learning capacity from increased training samples. Benefit from the finetuning operation as well as FC layer in final classification and regression, Meta-RCNN can still maintain consistent improvement when trained with more samples. Furthermore, it's interesting that vanilla FRCN outperforms FRCN-PN even in very few-shot cases (5way-1shot) if FRCN-PN is not finetuned, where non-parametric property does not help PN obtain better performance. We argue this is because few-shot detection is generally more challenging than few-shot classification, as we discussed in introduction section. FRCN-PN cannot learn a representative prototype of background classes and the whole framework cannot be optimized by meta learning style (e.g., RPN and bbox regressors). The failure of FRCN-PN indicates naively attach components from few-shot classification framework cannot solve few-shot detection problem. Finally, our Meta-RCNN achieves better results than all the baselines.

**Ablation study of RPN:** Here, we analyze the performance of RPN to validate our concerns of negative impact of irrelevant categories. We use vanilla FRCN and FRCN-PN as our baselines. The models are optimized in the same manner as before, but during Meta-Testing phase, we evaluate the **recall** performance on each task instead of mAP performance.

As observed from Table 6.3, the vanilla FRCN outperforms FRCN-PN significantly. This is because objects of irrelevant categories in the same image hurt the training process of RPN. And our Meta-RCNN outperforms these two baselines significantly. Meta-RCNN learns a general feature map for all $K$way-$N$shot detection tasks and optimize RPN by meta learning, which proves more effective in few-shot settings. Notably, the results are surprising since the recall of RPN in few-shot scenario is significantly lower ($> 90\%$ with enough training data on VOC dataset).

| **Model** | Backbone | 5w-1s | 5w-3s | 5w-5s |
|---|---|---|---|---|
| vanilla FRCN | VGG16 | 24.9% | 26.5% | 28.4% |
| FRCN-PN | VGG16 | 24.7% | 24.9% | 26.1% |
| Meta-RCNN (ours) | VGG16 | **26.1**% | **27.9**% | **33.7**% |

Table 6.3: Recall evaluation of Meta-RCNN on FSOD-VOC BENCHMARK test set. For brevity, “5way-1shot” is abbreviated as ”5w-1s”.

## 6.4.3 Results on FSOD-ImageNet Benchmark

On FSOD-ImageNet benchmark, we adapt weights of detector pretrained on M-SCOCO trainval set, and then optimize Meta-RCNN based on this starting point. The Meta-RCNN is evaluated on animal subset of ImageNet-LOC. Animal subset of ImageNet-LOC only contains single animal category per image, so there are no irrelevant classes during training and it’s simpler than the situation we discussed. In addition to FRCN and LSTD, we also include another latest baseline **RepMet** [170], which replaces FC classification layers in FRCN with more careful design of PN layers (learning multiple prototypes per class etc.), as well as much stronger backbone (DCN [25] and FPN [118]). Table 6.4.3 shows the results, in which our Meta-RCNN outperforms the other methods significantly.

| **Model** | Backbone | 50w-1s | 50w-5s |
|---|---|---|---|
| vanilla FRCN [162] | VGG16 | 16.5% | 34.3% |
| LSTD [13] | VGG16 | 19.2% | 37.4% |
| RepMet [170] | DCN+FPN | 24.1% | 39.6% |
| Meta-RCNN (ours) | ResNet101 | **25.3%** | **40.6%** |

Table 6.4: mAP performance evaluation on FSOD-IMAGENET BENCHMARK. Here "50way-1shot" is abbreviated as "50w-1s".

### 6.4.4 Results on Traditional VOC Bencmark

Here we compare our model on VOC dataset in the same manner (1 task) as several baselines in literature, instead of multiple episodic tasks. We follow the experiment settings as [84]. We first train on a large annotated dataset, and then finetune on a single few-shot dataset. We use VOC2007 trainval and VOC2012 trainval for training, and VOC2007 test set for testing. During training, we use 15 categories for large annotated dataset and 5 categories for few-shot dataset. Note that number of shots here denotes number of objects instead of number of images, for fair comparison. We report the results in Table 6.4.4, in which our Meta-RCNN surpasses all the competitors on the same benchmark.

| **Model** | 1s | 2s | 3s | 5s | 10s |
|---|---|---|---|---|---|
| YOLO-joint | 0.0 | 0.0 | 1.8 | 1.8 | 1.8 |
| YOLO-ft | 3.2 | 6.5 | 6.4 | 7.5 | 12.3 |
| YOLO-ft-full | 6.6 | 10.7 | 12.5 | 24.8 | 38.6 |
| LSTD [13] | 8.2 | 11.0 | 12.4 | 29.1 | 38.5 |
| MetaYolo [84] | 14.8 | 15.5 | 26.7 | 33.9 | 47.2 |
| MetaDet-YOLO [210] | 17.1 | 19.1 | 28.9 | 35.0 | 48.8 |
| MetaDet-FRCN [210] | 18.9 | 20.6 | 30.2 | 36.8 | 49.6 |
| Meta-RCNN (ours) | **19.1** | **23.6** | **32.5** | **39.9** | **50.5** |

Table 6.5: mAP performance on PASCAL VOC BENCHMARK. All the models are evaluated with 5 ways on VOC2007 test set. For brevity, "1-shot" is abbreviated as "1s".

## 6.5 Discussion

We investigated the problem of few-shot object detection and proposed a meta-learning based few-shot object detection named Meta-RCNN. The proposed training strategies make Meta-RCNN robust and more suitable in few-shot detection s-cenarios. Specifically it adapts the Faster RCNN method and enables meta-learning of the object classifier, the RPN and the bounding box regressor. The RPN is meta-trained through a novel class-specific attention module. We conduct extensive experiments and obtain promising results. In future work, we plan to extend our framework by exploring more recent meta-learning techniques and evaluating diverse detectors.

# Chapter 7

# MCD: Meta Contrastive Learning for Few-shot Object Detection

## 7.1 Introduction

In Chapter 6 we have introduced Meta-RCNN, a few-shot detector which is optimized following the principle of meta-learning. The weight-shared binary classifier in Meta-RCNN makes it robust to few-shot detection and it has achieved achieved promising results with only a few training data annotated. However, the binary classifier potentially makes it weak in feature representation learning, and the ratio of positive and negative samples is extremely imbalanced. In this Chapter, we further explore the limitations of Meta-RCNN and propose a new framework, Meta Constrastive Detector (MCD). The new proposed MCD overcomes the listed limitations, and shows significant improvement.

Inspired by the success of meta-learning on image classification, in addition to Meta-RCNN, other meta-learning based detection frameworks have also been proposed [218, 84, 170] for few-shot detection. The meta-learning or learning to learn principle guides the detector on how to quickly adapt to do well on a new few-shot detection task. Despite promising performances, existing approaches for few-shot detection primarily aim to use a pre-trained model and train under a meta-learning framework. As a result, their primary goal is to learn how to adapt quickly to a new task, and they do not focus on learning an effective representation. In order to learn both the detector and an effective representation, we propose a novel framework called Meta-Contrastive Detector (MCD), where a detector is trained using a contrastive loss in a meta-learning framework. This enables learning both an effective few-shot detector and a discriminative representation in a synergic manner. We also develop a strategy for sampling hard negative examples during training, which further increases the effectiveness of the meta-contrastive learning framework.

MCD is an end-to-end trainable meta detector which is optimized under the episodic learning paradigm. During training, multiple few-shot tasks are generated from the annotated meta-training dataset. Each task comprises a support set (simulating few-shot annotated training data) and a query set (simulating the test data). For a given task, prototypes for all the categories in support set are generated by the meta-model. The prototype of each category is combined with a query image through an attention mechanism to generate a class-specific feature map for that particular category. This feature map aims to highlight the regions in the query image where the objects of the specific category maybe found. A weight-shared RPN [162] is attached to each class-specific feature map to generate proposals, where each proposal is followed by an ROI pooling layer and FC layers. This is followed by three parallel branches: classification branch, localization branch and contrastive loss branch. The classification branch is a binary classifier which predicts whether a given proposal from a class-specific feature map is an object of that particular class or not. If the classifier makes the correct prediction, it is called a positive prediction, and if it makes an error, it is called a negative prediction. The localization branch predicts the bounding box coordinates of the detected object. Classification and regression losses are used to learn these branches. The third branch uses a contrastive loss principle to aid representation learning. In particular, the contrastive loss helps to learn an embedding that minimizes the distance between positive examples, and maximizes the distance of positive examples from negative examples. This leads to

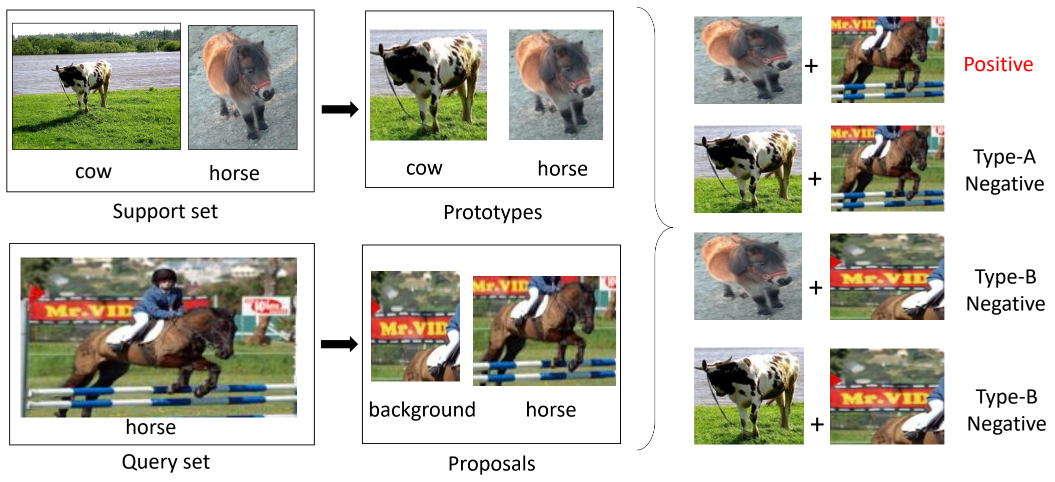


Figure 7.1: Examples of Positive, Type-A Negative and Type-B Negative samples in a meta-contrastive learning setting. Given support for class ”horse”, and a query image, several region proposals are generated. If the region proposal is a ”Horse”, it is considered as Positive; if it is an object of another class (e.g. ”cow”), it is considered as a Type-A Negative; if it is a background region, it is considered as a Type-B Negative. The number of Type-B negatives is much larger than Type-A negatives. Type-A negatives are often harder and offer stronger learning signals.

more discriminative representation, which gives a more robust few-shot detector.

During the training of the Meta-Contrastive Detector (and in general Meta-Learning for few-shot learning), category specific binary classifier gives the classification predictions. Existing approaches treat all negatives with equal importance. However, this should not be the case, as different types of negatives provide different levels of learning signals to train the model. For a given class specific feature map for class $C_1$, we identify two types of negatives that can occur: i) Type A: where an object belonging to another class $C_2$, where $C_1 \neq C_2$ is misclassified as $C_1$ ; and ii) Type B: where a background region in the image is classified as belonging to class $C_1$. See Figure 7.1 for Positive, Type-A Negative and Type-B Negative examples. Type-A negative is much harder than Type-B, as often two classes may share some similarities (e.g., horse and cow may have many common visual features), and thus Type-A negatives offer a much stronger learning signal than Type-B negatives. However, the number of Type-B negatives is extremely large during training. If both types of negatives are treated equally, the large number of Type-B negatives will dominate the training process, thus may lead to undesired poor performance. To address such imbalance issue, we propose to uniformly sample Type-A and Type-B negatives for training. In particular, this improves the utility of the contrastive loss function.

Combining the Meta-Contrastive framework with an effective sampling strategy, MCD achieves state-of-the-art results on Pascal VOC [31] and MSCOCO [120] in few-shot settings. The Contrastive learning branch is only used during training and thus there is no additional cost during inference. Further, the new proposed sampling strategy samples hard negatives without any extra computation.

## 7.2 Meta-Contrastive Detector for Few-shot Detection

### 7.2.1 Problem Setting

To train MCD, we have two training stages: meta-training and meta-testing. Assume we have a total of $C$ categories which are split into two sets: $C_{train}$ and $C_{test}$, ($C_{train} \bigcap C_{test} = \varnothing$). $C_{train}$ is used for meta-training with large annotated data, while $C_{test}$ is used for meta-testing with only a few objects annotated. In each episode, the model is trained with a support image $s_c$ from support set $T_s$ and a query image $q_c$ from query set $T_q$ which contains objects belong to category $c$. The detector is required to detect all objects in query images belong to the support category. If the support image contains K categories with N object samples per class, we denote this task as $K$way-$N$shots task ($N$ denotes the number of the objects). The goal of the meta-learning based detector is to get the highest object detection accuracy on a novel few-shot test tasks (tasks drawn from the meta-test dataset).

### 7.2.2 Overview of Existing Methods and Their Limitations

Our model is based on Faster RCNN [162], a state-of-the-art two-stage detection algorithm. Here we briefly review the structure of Faster RCNN. Faster RCNN consists of two components: region proposal network (RPN) to generate proposals and Fast RCNN for region classification. RPN aims to generate a sparse set of proposals to filter easy negatives. RPN extracts fixed-length features on each position of the feature map by scanning the whole image with sliding windows. The extracted features are then fed into two sibling branches: binary classification branch for objectness prediction and regression branch to refine the bounding box coordinates. After proposal generation, a fixed-length region feature is extracted from each proposal by region feature encoding method (ROI Pooling, etc.), which is followed by a $C+1$ region classifiers ($C+1$ denotes the $C$ category plus $1$ background class) and a bounding box regressors. The whole framework can be optimized in an end-to-end manner.

Although Faster RCNN and its variants have obtained state-of-the-art results, it still requires a lot of training samples. Yan et al. [218] proposed Meta-RCNN which is based on Faster RCNN and follows the principle of meta-learning. In their model, a set of prototypes of support categories are extracted from support set by the meta-model. Then a set of class-specific region features is generated by combining query features with prototypes. These class-specific region features are then fed into a binary classifier for classification. While learning to quickly adapt, this approach does not aim to learn a discriminative representation. Moreover, it does not distinguish between the different types of negatives during training, which results in hard negatives not contributing sufficiently to the training.

### 7.2.3 Meta-Contrastive Detector

We now present our proposed framework MCD for few-shot detection. The pipeline is shown in Figure 7.2. In meta-training stage, the whole model is trained with mul-

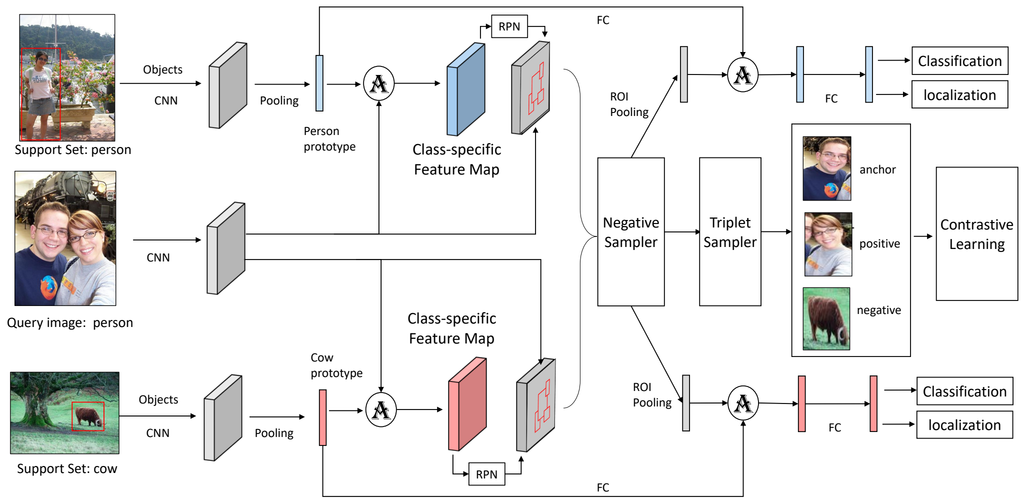


Figure 7.2: The pipeline of Meta-Contrastive Detector. Prototypes of all categories are extracted and a set of class-specific feature maps are produced by combining the features of query image and prototypes via an attention mechanism (shown as “A”-operation). This is followed by a weight-shared RPN to generate proposals. The negative sampler is applied to sample training samples, which are extracted from the original query feature map and are encoded with the prototypes. The regions are fed into the binary classifier and loc regressor. Contrastive loss is used in the contrastive learning branch to improve the representation, and triplet sampler is applied here to sample hard triplets.

tiple few-shot tasks generated from meta-training dataset. For each $K$way-$N$shot task, $N$ objects per class are annotated and a single query image is fed into the network. The prototype of each category is extracted by the meta-model from the support set. The query feature map and the category prototype are merged via an attention mechanism to generate a class-specific feature map. An RPN is applied into the new generated feature maps to generate proposals. Proposals are generated for positive samples, Type-A negatives and Type-B negatives. We apply our negative sampling strategy to randomly choose a Type-A negative (a proposal containing an object of another class) and a Type-B negative, to maintain uniform distribution over the number of Type-A negative and Type-B negative samples. Then we encode the region features of the selected proposals by ROI Pooling from original query image. These region features are then encoded with their corresponding prototype and are fed into three sibling branches: binary classification branch, localization branch, and contrastive loss branch. We use triplet margin loss in our contrastive learning module with the proposed hard samples mining.

In our proposed MCD, we adopt the similar detection structure as original Meta-RCNN but with several critical modifications: (1) We learn a contrastive learning branch for representation learning, aided by hard sample mining; (2) We apply a negative-type aware sampling strategy to select training samples; (3) We apply RPN after learning class-specific feature maps to meta-optimize the whole model. Our contrastive learning branch and negative-type aware sampling strategy are only used during training and thus there is no computation cost in inference stage. Next, we will introduce the details of our method.

### 7.2.4 Meta-training

During meta-training, $K$way-$N$shot tasks are generated from the subset of category $C_{train}$ to form support set $T_s$. Follow the similar principle of Yan et al. [218], we crop objects of support categories from meta-training dataset with 16 pixels contexts, and then resize the selected images into 320x320 with an additional binary mask denoting the location of the target objects. For each category, a prototype $P_t$ is generated based on the corresponding region features:

$$P_t = \frac{1}{N}\sum_{i}^{N} r_t^i \tag{7.1}$$

where $N$ is the number of examples (shots) and $r_t^i$ is the global pooling feature of $i^{th}$ image in the support set belong to class $t$. Then we randomly sample one query image $I_q$ with category $q$ from query set $T_q$, and feed $I_q$ into the network to obtain feature map $F_q$ before RPN. For each prototype $P_t$, a class-specific feature map $FC_t$ is generated by an attention module from $F_q$. The attention module (as shown as "A"-operation in Fig. 2) we use here is an element-wise multiplication between prototype and query feature:

$$FC_t = F_q \odot P_t \tag{7.2}$$

This is followed by an RPN to generate a proposal set $R_t$. Specifically, we select the top 128 proposals from each class-specific feature map, which results in a total of $128 * K$ proposals for all $K$ categories. We then sample top 128 proposals from this set.

**Negative Sampling Strategy.**

To balance the ratio of training samples, we adopt a negative-type aware sampling strategy to select proposals. Let $C1$ denote the class of class-specific feature map. For each region $r \in R$, the region $r$ is: (1). Positive, if $r$ is an object of class $C1$; (2). Type-A Negative, if $r$ is an object of another class $C2$ where $C2 \neq C1$; (3). Type-B Negative, $r$ is a background patch. Here we sample all positive samples, and for the negative samples, we randomly select from the two negative types. More specifically, if the number of positives is $N_p$, then we select $128 - N_p$ negatives which are uniformly sampled from Type-A and Type-B negatives. Type-A negatives tends to be harder than Type-B, and thus offer a stronger learning signal. Random sampling would have been dominated by Type-B negatives as they significantly outnumber them Type-A negatives, but in our approach, Type-A negatives are able to significantly contribute to the learning. In addition, compared with online hard example mining, we do not need an extra feed forward the whole batch samples into the network, and are thus computationally more efficient.

**Contrastive Learning.**

The features of selected training samples are extracted by ROI Pooling from the original query map, which are then encoded by the prototypes via an element-wise

| Name | Contrastive loss function |
|---|---|
| Blank | $L_{\text{CL}} = 0$ |
| Margin loss | $L_{\text{CL}} = y * \lvert p-n \rvert + (1-y) * \max(0, m - \lvert p-n \rvert)$ |
| Repulsion loss [208] | $L_{\text{CL}} = -ln(1 - \lvert p-n \rvert)$ |
| Triplet Margin loss | $L_{\text{CL}} = \max(0, \lvert p-a \rvert - \lvert n-a \rvert + m)$ |

Table 7.1: List of contrastive losses: (1) Blank; (2) Margin loss; (3) Repulsion; (4) Triplet Margin loss, where $m$ is the margin of contrastive loss, $p$, $a$, $n$ denote relevance scores of positive, anchor and negative respectively, and $y$ denotes label of training samples.

multiplication attention module:

$$RC_t = r_t \odot f(P_t; \phi) \tag{7.3}$$

where $r_t \in R_t$, denotes the region feature which is generated by RPN on $FC_t$, $f(P_t; \phi)$ denotes nonlinear transformation of prototype $P_t$, and $RC_t$ is the encoded region features of $r_t$. The encoded region features are fed into three sibling branches: a binary classification branch, a localization branch and a contrastive learning branch. The target of contrastive loss is to learn a distance-based embedding by contrasting positive and negative examples, especially hard negatives. Here, we list the 4 selections of contrastive learning loss in Table 7.1. From Table 7.3: (1). Blank: we don't consider any contrastive loss in this settings; (2) Margin loss: we change the triplet loss into basic margin loss; (3) Repulsion loss: we use repulsion loss [208] for our model, replacing the IoU value with relevance score, which has been proved effective in pedestrian detection tasks; (4) Triplet margin loss: the loss function used in our detector finally.

In our experiments, we show that the triplet margin contrastive loss performs best. In particular, a triplet marign contrastive loss has three inputs: $(a, p, n)$, where $a$ denotes the embedding of the anchor, $p$ denotes the emebedding of positive sample of the same category as $a$ and $n$ is the embedding of the negative sample. The parameter $m$ is the margin and we set it to 0.3. The probability of region in vanilla detection branch is used as relevance score to compute this loss. During training, triplet selection also has an important impact on final accuracy. In MCD, we sample the hard samples as triplet. More specifically, each positive sample $a$ is selected as anchor, and we compute the distances of all other proposals to it. The region with maximum distance and the same category with $a$ are selected as $p$, and the region with minimum distance with different category with $a$ is selected as $n$.

Contrastive learning branch models the region similarity with positives with most difficult negative samples, and thus leads to a a better representation learning giving us a robust detector. In our contrastive branch, we add region features computed by different query and support category (Type-B negative) and thus increase the diversity of negatives. Triplet loss is more reliable and stable than other metric loss (such as margin loss).

**Joint Multi-task Training.**

Finally the whole model is optimized in an end-to-end manner by minimizing the following multi-task objective function:

$$L(T_q, T_s, \theta) = L_{\text{cls}} + L_{\text{loc}} + L_{\text{RPN}} + L_{\text{CL}} \tag{7.4}$$

where $\theta$ is the parameters of our model, $L_{\text{cls}}$ is the binary classification loss, $L_{\text{loc}}$ is the bounding box localization loss, $L_{\text{RPN}}$ is the loss for learning the RPN, and $L_{\text{CL}}$ is the contrastive loss.

### 7.2.5 Meta-testing

During meta-testing stage, we sample few-shot tasks from meta-testing dataset ($C_{test}$). The annotated support set is used to finetune the model, and the query set is used to evaluate the final performance. The sampling strategy and contrastive learning branch are only used in finetuning, so there is no extra computation during inference. To further accelerate the model inference, we first extract the prototypes of target categories from support set by the finetuned model, and during inference stage, we input the pre-computed prototype as fixed inputs.

## 7.3 Experiments

### 7.3.1 Datasets and Implementation Details

*Datasets:* We evaluate our models on two benchmarks in few-shot settings: (i) Pascal VOC 2007/2012 and (2) MSCOCO. Pascal VOC 2007 and 2012 are two widely used datasets for object detection with 20 categories. Pascal VOC 2007 has 5k images in trainval set and 5k images in test set, while Pascal VOC 2012 has 16k images in trainval set. In our experiments, we merge the trainval set of VOC2012 and VOC2007 as our training set, and randomly select 15 categories as $C_{train}$ for meta-training, and the other 5 categories as $C_{test}$ for meta-testing. We use test set of VOC2007 as our test set. MSCOCO has 80 categories with 115k images in train set and 5k images in val set. We keep the same 20 categories of VOC dataset as $C_{test}$ and the rest 60 categories as $C_{train}$ for meta-training.

*Meta-model setting:* In our experiments, we randomly generate multiple $K$way-$N$shot tasks to train the model. We set $K$=2, and thus in each training iteration, an additional category of support set is selected that is different from the category of query image. The images belong to this set is used to generate more hard examples (Type-A negative) to train the model. We evaluate different values of $N$ ($N = 1, 3, \cdots, 10.$), and thus each training batch contains $2 * N + 1$ images.

*Basic detection settings:* We follow the basic detection settings as Faster RCNN. We use ResNet-101 and ResNet-50 as our backbone architectures trained with ImageNet classification dataset. Proposals overlaps with objects larger than 0.5 are considered as positive samples, and less than 0.3 are negative samples. For each image, top 128 proposals with highest confidence score are selected for training, and top 300 proposals with highest confidence score are used for region classification. For Pascal VOC dataset, we finetune the model for 5 epochs in meta-training stage, and 5 epochs in meta-testing stage. For MSCOCO, we finetune the model for 3 epochs in meta-training stage and 5 epochs in meta-testing stage. The initial learning rate is set to 0.001 and will decays 10 times by every 2 epochs. We crop the object from the original image with extra 16 pixels contexts to form support set. And the images are resized into 320x320 before fed into meta-model. For query image, the shorter size of the image is resized into 600 pixels, with maximum size as 1000 pixels in training. No additional data augmentation strategy is used except horizontal flipping.

### 7.3.2 Results on Pascal VOC Benchmark

In this section, we report the performance of our model on Pascal VOC benchmarks and compare it with the state-of-the-art methods. The model is trained on ResNet-101 and tested on the Pascal VOC 2007 test set. We set the shot number $N$ from 1 to 10, and keep the way number $K$ as 2. In Table 7.2, we report the results of 5 novel classes in few-shot settings, and from the table we can see, our method outperforms all other few-shot detection algorithms with large margin. Specifically, our results are significantly better than Meta-RCNN in all settings, which proves the effectiveness of the components of our model.

| **Model** | 1-shots | 2-shots | 3-shots | 5-shots | 10-shots |
|---|---|---|---|---|---|
| YOLO-joint [218] | 0.0 | 0.0 | 1.8 | 1.8 | 1.8 |
| YOLO-ft [218] | 3.2 | 6.5 | 6.4 | 7.5 | 12.3 |
| YOLO-ft-full [218] | 6.6 | 10.7 | 12.5 | 24.8 | 38.6 |
| FRCN+joint [218] | 2.7 | 3.1 | 4.3 | 11.8 | 29.0 |
| LSTD [13] | 8.2 | 11.0 | 12.4 | 29.1 | 38.5 |
| FRCN+ft [218] | 11.9 | 16.4 | 29.0 | 36.9 | 36.9 |
| FRCN+ft-full [218] | 13.8 | 19.6 | 32.8 | 41.5 | 45.6 |
| Meta-Yolo [84] | 14.8 | 15.5 | 26.7 | 33.9 | 47.2 |
| MetaDet-YOLO [210] | 17.1 | 19.1 | 28.9 | 35.0 | 48.8 |
| MetaDet-FRCN [210] | 18.9 | 20.6 | 30.2 | 36.8 | 49.6 |
| Meta-RCNN [218] | 19.9 | 25.5 | 35.0 | 45.7 | 51.5 |
| MCD (Ours) | **23.0** | **29.4** | **39.1** | **50.9** | **55.1** |

Table 7.2: mAP performance on PASCAL VOC BENCHMARK. All the models are evaluated with 5 ways on VOC2007 test set.

**Contrastive learning:** In this part, we aim to explore the effectiveness of contrastive learning and the factors which impact the final performances. Here, we set

7 selections of contrastive loss: 4 selections shown in Table 7.1 and 1 additional settings of triplet contrastive loss: triplet loss with randomly select triplet samples $(a, p, n)$. All models are trained with Negative Sampler shown in Figure 7.2. The results are shown in Table 7.3.

| **Model** | 1-shots | 2-shots | 3-shots | 5-shots | 10-shots |
|---|---|---|---|---|---|
| Blank | 21.2 | 27.2 | 37.9 | 47.0 | 53.5 |
| Margin loss | 23.0 | 29.4 | 39.1 | 50.9 | 55.1 |
| Repulsion loss | 22.5 | 29.1 | 38.5 | 49.2 | 54.8 |
| Triple loss +Rand loss | 22.3 | 27.8 | 38.4 | 47.3 | 53.9 |
| Triple loss +Triplet sampler (Ours) | **24.1** | **30.2** | **40.6** | **51.1** | **55.8** |

Table 7.3: Ablation study on contrastive learning. mAP performance on PASCAL VOC BENCHMARK. All the models are evaluated with 5 ways on VOC2007 test set.

All baselines with contrastive learning outperform the baseline "Blank" without contrastive loss. We note that "Blank" baseline still achieves better results than vanilla Meta R-CNN using the same structure mainly due to the effectiveness of the proposed negative sampler. Compared with Row 2, Row 3 and Row 7, our triplet loss with triplet hard sampling (Triplet Sampler in Figure 7.2) outperforms pairwise rank loss design, but triplet loss with random triplet sampling cannot defeat these two baselines. These results indicate that the triplet loss is more powerful than the pairwise rank loss, but the triplet sampler has a big impact on the final performance.

**Proposal Sampling.** In this section, we explore the effectiveness of our proposal sampling strategy of RPN output (Negative Sampler in Figure 7.2). Here we set 4 different sampling strategies and report the results with contrastive learning module: (1). All: we keep all the training samples generated by meta-model to train the model. In our 2way-$N$shot setting, this strategy increases 2 times proposals. (2). Positive: We only keep the class-specific feature maps generated by query features and support features which share the same categories. This strategy will not increase additional proposals, but lose the diversity of negatives. (3). We randomly sample negative training samples from all negative samples to build up the mini-batch, which avoids increasing negatives. (4). We only consider top 128 training sample with highest loss value of confidence score for training, which also avoids increasing negatives.

| **Model** | 1-shots | 2-shots | 3-shots | 5-shots | 10-shots |
|---|---|---|---|---|---|
| All | 8.4 | 14.2 | 18.1 | 24.6 | 27.1 |
| Random | 16.1 | 22.5 | 35.1 | 43.8 | 47.9 |
| Positive | 21.0 | 27.1 | 36.9 | 46.5 | 52.4 |
| OHEM | 19.9 | 28.2 | 38.7 | 49.9 | 54.7 |
| Ours | **23.0** | **29.4** | **39.1** | **50.9** | **55.1** |

Table 7.4: Ablation study on sampling strategy. mAP performance on PASCAL VOC BENCHMARK. All the models are evaluated with 5 ways on VOC2007 test set.

From Table 7.4, training detectors with all training samples performs extremely worse, and it's even worse than vanilla detectors. This demonstrates that sampling strategy can further boost the performances of contrastive learning model, and is necessary when applying meta-learning methods into detection task to address few-shot detection. Training detectors with random sampling performs unsatisfactory too, this is because most negative samples are Type-b negative, which is easy to identify, and the hard samples cannot be mined due to the number of it is relatively small. Meta-RCNN adopts positive training strategy, and it performs well. However, it loses the diversity of hard negatives and thus our training strategy and OHEM can defeat it. OHEM performs best except our strategy, but in very low shot cases it even performs worse than Positive training. I argue this is because OHEM samples training samples by their loss values, and in extremely low-shot settings, the loss value may not be reliable due to overfitting. Our training strategy samples negatives from different types and thus guarantees both types of negatives can be exploited for training, especially for the hard negatives.

**Way of training:** We examine the impact of the number of ways on final accuracy (See Table. 7.5). In our experiments, we set $K$ as 1, 2, 3, 5, with $K-1$ categories of negative support set. For 1-way, no negative category of support is involved. In our experiment, when $K > 1$ the performance of the detector can significantly outperforms $K = 1$ detectors, which means the effectiveness of negative training samples increasing the diversity. However, the performance will become saturated if $K > 2$. How to explore a more effective sampling strategy to mine more knowledge as $K$ increases remains an open question.

| **Model** | 1-shots | 2-shots | 3-shots | 5-shots | 10-shots |
|---|---|---|---|---|---|
| 1-way | 19.9 | 25.5 | 35.0 | 45.7 | 51.5 |
| 2-way | **23.0** | **29.4** | **39.1** | **50.9** | **55.1** |
| 3-way | 22.7 | 27.9 | 38.9 | 50.0 | 53.5 |
| 5-way | 22.4 | 28.1 | 39.0 | 49.9 | 52.9 |

Table 7.5: Ablation study on the number of ways during training. For each experiments we set number of shot as 5. mAP performance on PASCAL VOC BENCHMARK. All the models are evaluated with 5 ways on VOC2007 test set.

**Margin value:** We examine the impact of margin value $m$ on final results. Since we use the probability of object classifier as relevant score, so we set the value of m from 0 to 1, and we report our results in Table 7.6. From the results table, margin is necessary to improve the results which pulls negative samples away from positives. The margin value is not sensitive between 0.3 to 0.5.

| **Margin** | 1-shots | 2-shots | 3-shots | 5-shots | 10-shots |
|---|---|---|---|---|---|
| 0.0 | 20.6 | 27.7 | 37.0 | 46.8 | 49.3 |
| 0.1 | 22.8 | 28.7 | 39.0 | 49.9 | 54.3 |
| 0.3 | **23.0** | **29.4** | **39.1** | **50.9** | **55.1** |
| 0.5 | 22.6 | 28.5 | 38.9 | 48.8 | 54.1 |
| 0.7 | 18.1 | 25.4 | 36.9 | 46.6 | 49.9 |

Table 7.6: Ablation study on margin values. For each experiments we set number of shot as 5. mAP performance on PASCAL VOC BENCHMARK. All the models are evaluated with 5 ways on VOC2007 test set.

### 7.3.3 Results on MSCOCO Benchmark

In this section, we report the few-shot detector benchmark evaluation results on MSCOCO datasets. We follow the same experimental setup as the previous studies of applying meta learning for few-shot detection [218]. The model is trained with ResNet-50, and we resize the shorter size of the image into 800 pixels, with longer size no more than 1333 pixels. Due to the size of MSCOCO object is very small, we use the whole image as support set appended by a binary mask. The support image is resized into 320x320. Table 7.7 shows the evaluation results. As observed from Table 7.7, the proposed MCD detector outperforms all the state-of-the-art methods with substantial margins.

| Shot | Baselines | Backbone | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|---|
| 10 | Meta-Yolo [84] | DarkNet-19 | 5.6 | 12.3 | 4.6 | 0.9 | 3.5 | 10.5 |
| | FRCN+ft [218] | ResNet-50 | 1.3 | 4.2 | 0.4 | 0.4 | 0.9 | 2.1 |
| | FRCN+ft-full [218] | ResNet-50 | 6.5 | 13.4 | 5.9 | 1.8 | 5.3 | 11.3 |
| | MetaDet [210] | VGG16 | 7.1 | 14.6 | 6.1 | 1.0 | 4.1 | 12.2 |
| | Meta R-CNN [218] | ResNet-50 | 8.7 | 19.1 | 6.6 | 2.3 | 7.7 | 14.0 |
| | MCD (Ours) | ResNet-50 | **9.9** | **21.6** | **7.9** | **2.5** | **9.4** | **14.8** |
| 30 | Meta-Yolo [84] | DarkNet-19 | 9.1 | 19.0 | 7.6 | 0.8 | 4.9 | 16.8 |
| | FRCN+ft [218] | ResNet-50 | 1.5 | 4.8 | 0.5 | 0.3 | 1.8 | 2.0 |
| | FRCN+ft-full [218] | ResNet-50 | 11.1 | 21.6 | 10.3 | 2.9 | 8.8 | 18.9 |
| | MetaDet [210] | VGG16 | 11.3 | 21.7 | 8.1 | 1.1 | 6.2 | 17.3 |
| | Meta R-CNN [218] | ResNet-50 | 12.4 | 25.3 | 10.8 | 2.8 | 11.6 | 19.0 |
| | MCD (Ours) | ResNet-50 | **13.1** | **28.8** | **12.1** | **3.1** | **13.5** | **19.2** |

Table 7.7: Low-shot detection performance on COCO val set for 20-way novel classes. We evaluate the performance for different shot examples of novel classes under FRCN pipeline with ResNet-50.

## 7.4 Discussion

We investigate the problem of existing meta-learning based detection frameworks and proposed a new few-shot detector which follows the principle of meta learning but address the existing issues. The proposed contrastive learning module effectively identify the hard examples, while the new proposed sampling strategy selects training samples by mining hard examples from different type of negatives, without computation cost. We conduct extensive experiments on Pascal VOC and MSCOCO and obtain promising results. In the future, we hope to extend our work by applying more recent proposed meta-learning methods into detection problem.

# Chapter 8

# Future Directions

Object detection has been actively investigated and new state-of-the-art results have been reported almost every few months. However, there are still many open challenges. In this chapter we discuss several open challenges and future directions.

*(i) Scalable proposal generation strategy.* As claimed in Sec. 2.3.4, currently most detectors are anchor-based methods, and there are some critical shortcomings which limit the detection accuracy. Current anchor priors are mainly manually designed which is difficult to match multi-scale objects and the matching strategy based on IoU is also heuristic. Although some methods have been proposed to transform anchor-based methods into anchor-free methods (e.g. methods based on keypoints), there are still some limitations (high computation cost etc.) with large space to improve. From Figure 2.2, developing anchor-free methods becomes a very hot topic in object detection [98, 247, 244, 196, 29], and thus designing an efficient and effective proposal generation strategy is potentially a very important research direction in the future.

*(ii) Effective encoding of contextual information.* Contexts can contribute or impede visual object detection results, as objects in the visual world have strong relationships, and contexts are critical to better understand the visual worlds. However, little effort has been focused on how to correctly use contextual information. How to incorporate contexts for object detection effectively can be a promising future

direction.

*(iii) Detection based on Auto Machine Learning (AutoML).* To design an optimal backbone architecture for a certain task can significantly improve the results but also requires huge engineering effort. Thus to learn backbone architecture directly on the datasets is a very interesting and important research direction. From Figure 2.2, inspired by the pioneering AutoML work on image classification [258, 194], more relevant work has been proposed to address detection problems via AutoM-L [17, 43], such as learning FPN structure [43] and learning data augmentation policies [257], which show significant improvement over the baselines. However, the required computation resource for AutoML is unaffordable to most researcher-s (more than 100 GPU cards to train a single model). Thus, developing a low-computation framework shall have a large impact for object detection. Further, new structure policies (such as proposal generation and region encoding) of detection task can be explored in the future.

*iv) Emerging benchmarks for object detection.* Currently MSCOCO is the most commonly used detection benchmark testbed. However, MSCOCO has only 80 categories, which is still too small to understand more complicated scenes in real world. Recently, a new benchmark dataset LVIS [54] has been proposed in order to collect richer categorical information. LVIS contains 164,000 images with 1000+ categories, and there are total of 2.2 million high-quality instance segmentation masks. Further, LVIS simulates the real-world low-shot scenario where a large number of categories are present but per-category data is sometimes scarce. LVIS will open a new benchmark for more challenging detection, segmentation and low-shot learning tasks in near future.

*(vi) Backbone architecture for detection task.* It has become a common practice to adopt weights of classification models pretrained on a large scale dataset for detection. However, there still exists conflicts between classification and detection tasks [113], and thus directly adopting a pretrained network may not result in the optimal solution. Most state-of-the-art detection algorithms are based on classifica-

tion backbones, and only a few of them try different selections (such as CornerNet based on Hourglass Net). Thus, developing a detection-aware backbone architecture is also an important research direction for the future.

*(vii) Other research issues.* In addition, there are some other open research issues, such as large batch learning [147] and incremental learning [177]. Batch size is a key factor in DCNN training but has not been well studied for detection. For incremental learning, detection algorithms still suffer from catastrophic forgetting if adapted to a new task without initial training data. These open and fundamental research issues also deserve more attention for future work.

# Chapter 9

# Dissertation Conclusion

In this dissertation we present the challenges of applying generic detection algorithms into real-world problems, and propose frameworks to address these challenges. Specifically, we explore three real-world problems: scale-invariant detection, high-quality detection and few-shot detection, and propose multiple frameworks which show significant improvement and achieve state-of-the-art results on these tasks .

For scale-invariant detection, we use face detection as our evaluation benchmark and propose a novel framework of "Feature Agglomeration Networks" (FAN) to build a new single stage face detector. A novel feature agglomeration block is proposed to aggregate higher-level semantic feature maps of different scales as contextual cues to augment lower-level feature maps, which is optimized in a hierarchical manner. The proposed FAN detector is evaluated on several public face detection benchmarks and achieved state-of-the-art results with real time inference speed.

For high-quality detection, we propose two frameworks: "Bidirectional Pyramid Networks" (BPN) and "KPNet". BPN consists of two novel components: (i) a Bidirectional Feature Pyramid structure for more effective and robust feature representations; and (ii) a Cascade Anchor Refinement to gradually refine the quality of pre-designed anchors for more effective training. KPNet is an anchor-free detection algorithms which automatically learns to optimize a dynamic set of high-quality keypoints without heuristic anchor design. In real-world detection problems, domain knowledge is required to design anchor shapes. Our proposed BPN is able to refine the ill-defined anchors, and KPNet selects keypoints making a lot of components automatic. Both BPN and KPNet show significant improvement over existing detection methods on MSCOCO dataset, especially in high quality detection settings.

For few-shot detection, we propose two novel meta-learning based few-shot detectors: "Meta-RCNN" and "Meta Constrastive Detector" (MCD). Meta-RCNN learns a binary object detector in an episodic learning paradigm on the training data with a class-aware attention module. Meta-RCNN can be end-to-end meta-optimized and shows significantly improvements. Based on Meta-RCNN, MCD follows the principle of contrastive learning to enhance the representation for few-shot detection, and a new hard negative sampling strategy is proposed to address imbalance issue of training samples. We demonstrate the effectiveness of Meta-RCNN and MCD in few-shot detection on Pascal VOC dataset and obtain promising results.

In summary, we have explored novel techniques to address three research challenges to make object detection algorithms practical for real-world applications. In particular, we first explore scale-invariant detection and propose FAN to achieve state-of-the-art results on face detection benchmarks (Chapter 3). We second explore high-quality detection and propose an anchor-based method BPN (Chapter 4) and an anchor-free method KPNet (Chapter 5), both of which show significant improvement in high quality settings. Finally, we study the last problem few-shot detection and propose two meta-learning based detectors: Meta-RCNN (Chapter 6) and MCD (Chapter 7). We hope this dissertation can spur future research on developing object detection algorithms for real-world applications.

# List of Publications

## Conference Papers

Doyen Sahoo, Hao Wang, Ke Shu, **Xiongwei Wu**, Hung Le, Palakorn Achananuparp, Eepeng Lim, Steven C. H. Hoi. FoodAI: Food Image Recognition via Deep Learning for Smart Food Logging. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, USA, 2019.

**Xiongwei Wu**, Doyen Sahoo, Steven C. H. Hoi. MCD: Meta Contrastive Learning for Few-shot Object Detection. In *The 16th European Conference on Computer Vision (Under Submission)*, Glasgow, 2020.

**Xiongwei Wu**, Doyen Sahoo, Steven C. H. Hoi. Meta-RCNN: Meta Learning for Few-Shot Object Detection. In *ACM Multimedia Conference 2020 (Under Submission)*, USA, 2020.

**Xiongwei Wu**, Doyen Sahoo, Steven C. H. Hoi. KPNet: Learning to Optimize Keypoints for Keypoint Based Object Detection. In *The 34th Conference on Neural Information Processing Systems (Under Submission)*, Canada, 2020.

**Xiongwei Wu**, Steven C. H. Hoi, David Lo, Dehai Zhao, Zhenchang Xing. Flow Attention Network for Workflow Action Recognition from Programming Screencasts. In *The ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2020) (Under Submission)*, USA, 2020.

## Journal Papers

**Xiongwei Wu**, Doyen Sahoo, Steven C. H. Hoi. Recent Advances in Deep Learning for Object Detection. *Neurocomputing*, In press, 2020, doi:10.1016/j.neucom.2020.01.085.

**Xiongwei Wu**, Doyen Sahoo, Daoxin Zhang, Jianke Zhu, Steven C. H. Hoi. Single-shot Bidirectional Pyramid Networks for High-quality Object Detection. *Neurocomputing*, In press, 2020, doi:10.1016/j.neucom.2020.02.116.

Jialiang Zhang*, **Xiongwei Wu***, Steven C. H. Hoi, Jianke Zhu. Feature Agglomeration Networks for Single Stage Face Detection. *Neurocomputing*, Volume 380, 2020, pp. 180-189, doi:10.1016/j.neucom.2019.10.087. *($*$ denotes equal contribution)*

# Bibliography

[1] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. In *TPAMI*, 2012.

[2] A. Angelova, A. Krizhevsky, V. Vanhoucke, A. S. Ogale, and D. Ferguson. Real-time pedestrian detection with deep network cascades. In *BMVC*, 2015.

[3] A. Antoniou, H. Edwards, and A. Storkey. How to train your maml. *arXiv preprint arXiv:1810.09502*, 2018.

[4] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *ECCV*, 2006.

[5] 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.

[6] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis. Soft-nms – improving object detection with one line of code. In *ICCV*, 2017.

[7] A. Brock, J. Donahue, and K. Simonyan. Large scale gan training for high fidelity natural image synthesis. In *arXiv preprint arXiv:1809.11096*, 2018.

[8] Y. L. Buyu Li and X. Wang. Gradient harmonized single-stage detector. In *AAAI*, 2019.

[9] 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.

[10] Z. Cai and N. Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, 2018.

[11] J. Carreira and C. Sminchisescu. Cpmc: Automatic object segmentation using constrained parametric min-cuts. In *TPAMI*, 2011.

[12] D. Chen, G. Hua, F. Wen, and J. Sun. Supervised transformer network for efficient face detection. In *ECCV*, 2016.

[13] H. Chen, Y. Wang, G. Wang, and Y. Qiao. Lstd: A low-shot transfer detector for object detection. In *AAAI*, 2018.

[14] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *arXiv preprint arXiv:1412.7062*, 2014.

[15] X. Chen and A. Gupta. Spatial memory for context reasoning in object detection. In *ICCV*, 2017.

[16] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng. Dual path networks. In *NeurIPS*, pages 4467–4475, 2017.

[17] Y. Chen, T. Yang, X. Zhang, G. Meng, C. Pan, and J. Sun. Detnas: Neural architecture search on object detection. In *arXiv preprint arXiv:1903.10979*, 2019.

[18] Z. Chen, S. Huang, and D. Tao. Context refinement for object detection. In *ECCV*, 2018.

[19] B. Cheng, Y. Wei, H. Shi, R. Feris, J. Xiong, and T. Huang. Revisiting rcnn: On awakening the classification power of faster rcnn. In *ECCV*, 2018.

[20] C. Chi, S. Zhang, J. Xing, Z. Lei, S. Z. Li, and X. Zou. Selective refinement network for high performance face detection. In *arXiv preprint arXiv:1809.02693*, 2018.

[21] W. Chu and D. Cai. Deep feature based contextual model for object detection. In *Neurocomputing*, 2018.

[22] L. Cui. Mdssd: Multi-scale deconvolutional single shot detector for small objects. In *arXiv preprint arXiv:1805.07009*, 2018.

[23] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. In *CVPR*, 2016.

[24] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In *NeurIPS*, 2016.

[25] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *ICCV*, 2017.

[26] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[28] X. Dong, L. Zheng, F. Ma, Y. Yang, and D. Meng. Few-example object detection with model communication. In *TPAMI*, 2018.

[29] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian. Centernet: Keypoint triplets for object detection. In *arXiv preprint arXiv:1904.08189*, 2019.

[30] I. Endres and D. Hoiem. Category-independent object proposals with diverse ranking. In *TPAMI*, 2014.

[31] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. In *IJCV*, 2010.

[32] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Discriminatively trained mixtures of deformable part models. In *PASCAL VOC Challenge*, 2008.

[33] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. In *TPAMI*, 2010.

[34] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. In *IJCV*, 2004.

[35] S. Fidler, R. Mottaghi, A. Yuille, and R. Urtasun. Bottom-up segmentation for top-down detection. In *CVPR*, 2013.

[36] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning, ICML*, 2017.

[37] Y. Freund, R. E. Schapire, et al. Experiments with a new boosting algorithm. In *ICML*, 1996.

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

[39] K. Fukushima and S. Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*. 1982.

[40] C. Galleguillos and S. Belongie. Context based object categorization: A critical survey. In *Computer vision and image understanding*, 2010.

[41] Y. Gao, F. Wang, H. Luan, and T.-S. Chua. Brand data gathering from live social media streams. In *Proceedings of International Conference on Multimedia Retrieval*, 2014.

[42] T. Gebru, J. Krause, Y. Wang, D. Chen, J. Deng, and L. Fei-Fei. Fine-grained car detection for visual census estimation. In *AAAI*, 2017.

[43] G. Ghiasi, T.-Y. Lin, and Q. V. Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *CVPR*, 2019.

[44] A. Ghodrati, A. Diba, M. Pedersoli, T. Tuytelaars, and L. Van Gool. Deepproposal: Hunting objects by cascading deep convolutional layers. In *ICCV*, 2015.

[45] S. Gidaris and N. Komodakis. Object detection via a multi-region and semantic segmentation-aware cnn model. In *ICCV*, 2015.

[46] S. Gidaris and N. Komodakis. Locnet: Improving localization accuracy for object detection. In *CVPR*, 2016.

[47] R. Girshick. Fast r-cnn. In *ICCV*, 2015.

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

[49] R. Girshick, F. Iandola, T. Darrell, and J. Malik. Deformable part models are convolutional neural networks. In *CVPR*, 2015.

[50] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010.

[51] Y. Gong, L. Liu, M. Yang, and L. Bourdev. Compressing deep convolutional networks using vector quantization. In *Computer Science*, 2014.

[52] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NeurIPS*, 2014.

[53] J. Gu, H. Hu, L. Wang, Y. Wei, and J. Dai. Learning region features for object detection. In *ECCV*, 2018.

[54] A. Gupta, P. Dollar, and R. Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *CVPR*, 2019.

[55] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *Fiber*, 2015.

[56] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *NeurIPS*, 2015.

[57] W. Han, P. Khorrami, T. L. Paine, P. Ramachandran, M. Babaeizadeh, H. Shi, J. Li, S. Yan, and T. S. Huang. Seq-nms for video object detection. In *arXiv preprint arXiv:1602.08465*, 2016.

[58] Z. Hao, Y. Liu, H. Qin, J. Yan, X. Li, and X. Hu. Scale-aware face detection. In *CVPR*, 2017.

[59] H. Harzallah, F. Jurie, and C. Schmid. Combining efficient object localization and image classification. In *ICCV*, 2009.

[60] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *ICCV*, 2017.

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

[62] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[63] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*. Springer, 2016.

[64] Y. He, X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks. In *ICCV*, 2017.

[65] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. Support vector machines. In *IEEE Intelligent Systems and their applications*, 1998.

[66] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. In *arXiv preprint arXiv:1503.02531*, 2015.

[67] S. C. Hoi, X. Wu, H. Liu, Y. Wu, H. Wang, H. Xue, and Q. Wu. Logo-net: Large-scale deep logo detection and brand recognition with deep region-based convolutional networks. In *arXiv preprint arXiv:1511.02462*, 2015.

[68] J. Hosang, R. Benenson, and B. Schiele. Learning non-maximum suppression. In *CVPR*, 2017.

[69] J. Hosang, M. Omran, R. Benenson, and B. Schiele. Taking a deeper look at pedestrians. In *CVPR*, 2015.

[70] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. In *arXiv preprint arXiv:1704.04861*, 2017.

[71] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei. Relation networks for object detection. In *CVPR*, 2018.

[72] P. Hu and D. Ramanan. Finding tiny faces. In *CVPR*, 2017.

[73] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[74] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *CVPR*, 2017.

[75] Z. Huang, L. Huang, Y. Gong, C. Huang, and X. Wang. Mask scoring r-cnn. In *CVPR*, 2019.

[76] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[77] V. Jain and E. Learned-Miller. Fddb: A benchmark for face detection in unconstrained settings. Technical Report UM-CS-2010-009, University of Massachusetts, Amherst, 2010.

[78] J. Jeong, H. Park, and N. Kwak. Enhancement of ssd by concatenating feature maps for object detection. In *arXiv preprint arXiv:1705.09587*, 2017.

[79] 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 *Proceedings of 22nd ACM intl. conference on Multimedia*, 2014.

[80] B. Jiang, R. Luo, J. Mao, T. Xiao, and Y. Jiang. Acquisition of localization confidence for accurate object detection. In *ECCV*, 2018.

[81] R. G. Kaiming He and P. Dollro. Rethinking imagenet pre-training. In *arXiv preprint arXiv:1811.08883*, 2018.

[82] Z. Kalal, J. Matas, and K. Mikolajczyk. Weighted sampling for large-scale boosting. In *BMVC*, 2008.

[83] Y. Kalantidis, L. Pueyo, M. Trevisiol, R. van Zwol, and Y. Avrithis. Scalable triangulation-based logo recognition. In *Proceedings of ACM International Conference on Multimedia Retrieval*, 2011.

[84] B. Kang, Z. Liu, X. Wang, F. Yu, J. Feng, and T. Darrell. Few-shot object detection via feature reweighting. In *ICCV*, 2019.

[85] K. Kang, W. Ouyang, H. Li, and X. Wang. Object detection from video tubelets with convolutional neural networks. In *CVPR*, 2016.

[86] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[87] W. Ke, J. Chen, J. Jiao, G. Zhao, and Q. Ye. Srn: side-output residual network for object symmetry detection in the wild. In *CVPR*, 2017.

[88] K.-H. Kim, S. Hong, B. Roh, Y. Cheon, and M. Park. Pvanet: deep but lightweight neural networks for real-time object detection. In *arXiv preprint arXiv:1608.08021*, 2016.

[89] Y. D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. In *Computer Science*, 2015.

[90] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *arXiv preprint arXiv:1412.6980*, 2014.

[91] J. Kleban, X. Xie, and W.-Y. Ma. Spatial pyramid mining for logo detection in natural scenes. In *Multimedia and Expo, 2008 IEEE International Conference on*, 2008.

[92] T. Kong, F. Sun, W. Huang, and H. Liu. Deep feature pyramid reconfiguration for object detection. In *ECCV*, 2018.

[93] T. Kong, F. Sun, H. Liu, Y. Jiang, and J. Shi. Foveabox: Beyond anchor-based object detector. *arXiv preprint arXiv:1904.03797*, 2019.

[94] T. Kong, F. Sun, A. Yao, H. Liu, M. Lu, and Y. Chen. Ron: Reverse connection with objectness prior networks for object detection. In *CVPR*, 2017.

[95] T. Kong, A. Yao, Y. Chen, and F. Sun. Hypernet: Towards accurate region proposal generation and joint object detection. In *CVPR*, 2016.

[96] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

[97] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, T. Duerig, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. In *arXiv preprint arXiv:1811.00982*, 2018.

[98] H. Law and J. Deng. Cornernet: Detecting objects as paired keypoints. In *ECCV*, 2018.

[99] H. Law, Y. Teng, O. Russakovsky, and J. Deng. Cornernet-lite: Efficient keypoint based object detection. *arXiv preprint arXiv:1904.08900*, 2019.

[100] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, 1998.

[101] K. Lee, J. Choi, J. Jeong, and N. Kwak. Residual features and unified prediction network for single stage detection. In *arXiv preprint arXiv:1707.05031*, 2017.

[102] L. J. Z. X. Lele Xie, Yuliang Liu. Derpn: Taking a further step toward more general object detection. In *AAAI*, 2019.

[103] B. Li, T. Wu, L. Zhang, and R. Chu. Auto-context r-cnn. In *arXiv preprint arXiv:1807.02842*, 2018.

[104] H. Li, Y. Liu, W. Ouyang, and X. Wang. Zoom out-and-in network with recursive training for object proposal. In *arXiv preprint arXiv:1702.05711*, 2017.

[105] J. Li, X. Liang, S. Shen, T. Xu, J. Feng, and S. Yan. Scale-aware fast r-cnn for pedestrian detection. In *IEEE Transactions on Multimedia*, 2018.

[106] J. Li, X. Liang, Y. Wei, T. Xu, J. Feng, and S. Yan. Perceptual generative adversarial networks for small object detection. In *CVPR*, 2017.

[107] J. Li, Y. Wang, C. Wang, Y. Tai, J. Qian, J. Yang, C. Wang, J. Li, and F. Huang. Dsfd: Dual shot face detector. In *CVPR*, 2019.

[108] J. Li, Y. Wei, X. Liang, J. Dong, T. Xu, J. Feng, and S. Yan. Attentive contexts for object detection. In *IEEE Transactions on Multimedia*, 2017.

[109] Q. Li, S. Jin, and J. Yan. Mimicking very efficient network for object detection. In *CVPR*, 2017.

[110] Y. Li, Y. Chen, N. Wang, and Z. Zhang. Scale-aware trident networks for object detection. In *arXiv preprint arXiv:1901.01892*, 2019.

[111] Y. Li, J. Li, W. Lin, and J. Li. Tiny-dsod: Lightweight object detection for resource-restricted usages. In *arXiv preprint arXiv:1807.11013*, 2018.

[112] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun. Light-head r-cnn: In defense of two-stage object detector. In *arXiv preprint arXiv:1711.07264*, 2017.

[113] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun. Detnet: A backbone network for object detection. In *ECCV*, 2018.

[114] Z. Li and F. Zhou. Fssd: Feature fusion single shot multibox detector. In *arXiv preprint arXiv:1712.00960*, 2017.

[115] Z. Li, F. Zhou, F. Chen, and H. Li. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

[116] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *International Conference on Image Processing*, 2002.

[117] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPR workshops*, 2017.

[118] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[119] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *ICCV*, 2017.

[120] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[121] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *arXiv preprint arXiv:1712.01887*, 2017.

[122] S. Liu, D. Huang, and Y. Wang. Receptive field block net for accurate and fast object detection. In *ECCV*, 2018.

[123] 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.

[124] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song. Sphereface: Deep hypersphere embedding for face recognition. In *CVPR*, 2017.

[125] Y. Liu and M. D. Levine. Multi-path region-based convolutional neural network for accurate detection of unconstrained" hard faces". In *Computer and Robot Vision (CRV), 2017 14th Conference on*, 2017.

[126] Y. Liu, H. Li, J. Yan, F. Wei, X. Wang, and X. Tang. Recurrent scale approximation for object detection in cnn. In *ICCV*, 2017.

[127] Y. Liu, R. Wang, S. Shan, and X. Chen. Structure inference net: Object detection using scene-level context and instance-level relationships. In *CVPR*, 2018.

[128] D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999.

[129] D. G. Lowe. Distinctive image features from scale-invariant keypoints. In *IJCV*, 2004.

[130] X. Lu, B. Li, Y. Yue, Q. Li, and J. Yan. Grid r-cnn. In *CVPR*, 2019.

[131] Y. Lu, T. Javidi, and S. Lazebnik. Adaptive object detection using adjacency and zoom prediction. In *CVPR*, 2016.

[132] W. Luo, Y. Li, R. Urtasun, and R. Zemel. Understanding the effective receptive field in deep convolutional neural networks. In *NeurIPS*, 2016.

[133] S. Majid Azimi. Shuffledet: Real-time vehicle detection network in on-board embedded uav imagery. In *ECCV*, 2018.

[134] S. Manen, M. Guillaumin, and L. Van Gool. Prime object proposals with randomized prim's algorithm. In *CVPR*, 2013.

[135] M. Mathias, R. Benenson, M. Pedersoli, and L. Van Gool. Face detection without bells and whistles. In *ECCV*, 2014.

[136] H. Mobahi, R. Collobert, and J. Weston. Deep learning from temporal coherence in video. In *Annual International Conference on Machine Learning*, 2009.

[137] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[138] M. Najibi, P. Samangouei, R. Chellappa, and L. Davis. Ssh: Single stage headless face detector. In *ICCV*, 2017.

[139] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016.

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

[141] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. In *TPAMI*, 2002.

[142] D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. In *Journal of artificial intelligence research*, 1999.

[143] E. Osuna, R. Freund, and F. Girosit. Training support vector machines: an application to face detection. In *CVPR*, 1997.

[144] 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.

[145] J. Pang, K. Chen, J. Shi, H. Feng, W. Ouyang, and D. Lin. Libra r-cnn: Towards balanced learning for object detection. In *CVPR*, 2019.

[146] G. Pavlakos, X. Zhou, K. G. Derpanis, and K. Daniilidis. Coarse-to-fine volumetric prediction for single-image 3d human pose. In *CVPR*, 2017.

[147] C. Peng, T. Xiao, Z. Li, Y. Jiang, X. Zhang, K. Jia, G. Yu, and J. Sun. Megdet: A large mini-batch object detector. In *CVPR*, 2018.

[148] P. O. Pinheiro, R. Collobert, and P. Dollár. Learning to segment object candidates. In *NeurIPS*, 2015.

[149] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. In *ECCV*, 2016.

[150] A. Pon, O. Adrienko, A. Harakeh, and S. L. Waslander. A hierarchical deep architecture and mini-batch selection method for joint traffic sign and light detection. In *Conference on Computer and Robot Vision (CRV)*, 2018.

[151] A. P. Psyllos, C.-N. E. Anagnostopoulos, and E. Kayafas. Vehicle logo recognition using a sift-based enhanced matching scheme. In *Intelligent Transportation Systems, IEEE Transactions on*, 2010.

[152] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *arXiv preprint arXiv:1511.06434*, 2015.

[153] E. Rahtu, J. Kannala, and M. Blaschko. Learning a category independent object detection cascade. In *ICCV*, 2011.

[154] M. Rätsch, S. Romdhani, and T. Vetter. Efficient face detection by a cascaded support vector machine using haar-like features. In *Joint Pattern Recognition Symposium*, 2004.

[155] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. *ICLR*, 2016.

[156] M. Rayat Imtiaz Hossain and J. Little. Exploiting temporal information for 3d human pose estimation. In *ECCV*, 2018.

[157] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.

[158] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *arXiv preprint arXiv:1612.08242*, 2016.

[159] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. In *CVPR*, 2017.

[160] J. Ren, X. Chen, J. Liu, W. Sun, J. Pang, Q. Yan, Y.-W. Tai, and L. Xu. Accurate single stage detector using recurrent rolling convolution. In *CVPR*, 2017.

[161] M. Ren, E. Triantafillou, S. Ravi, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel. Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676*, 2018.

[162] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.

[163] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *CVPR*, 2019.

[164] H. Robbins and S. Monro. A stochastic approximation method. In *The annals of mathematical statistics*, 1951.

[165] S. Romberg, L. G. Pueyo, R. Lienhart, and R. van Zwol. Scalable logo recognition in real-world images. In *Proceedings of ACM International Conference on Multimedia Retrieval*, 2011.

[166] S. Romdhani, P. Torr, B. Scholkopf, and A. Blake. Computationally efficient face detection. In *ICCV*, 2001.

[167] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 2015.

[168] P. Samangouei, M. Najibi, L. Davis, and R. Chellappa. Face-magnet: Magnifying feature maps to detect small faces. In *arXiv preprint arXiv:1803.05258*, 2018.

[169] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. In *arXiv preprint arXiv:1801.04381*, 2018.

[170] E. Schwartz, L. Karlinsky, J. Shtok, S. Harary, M. Marder, S. Pankanti, R. Feris, A. Kumar, R. Giries, and A. M. Bronstein. Repmet: Representative-based metric learning for classification and one-shot object detection. In *CVPR*, 2019.

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

[172] K. S. D. A. Shang, Wenling and H. Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *ICML*, 2016.

[173] W. Shen, K. Zhao, Y. Jiang, Y. Wang, Z. Zhang, and X. Bai. Object skeleton extraction in natural images by fusing scale-associated deep side outputs. In *CVPR*, 2016.

[174] Z. Shen, Z. Liu, J. Li, Y.-G. Jiang, Y. Chen, and X. Xue. Dsod: Learning deeply supervised object detectors from scratch. In *ICCV*, 2017.

[175] Z. Shen, H. Shi, R. Feris, L. Cao, S. Yan, D. Liu, X. Wang, X. Xue, and T. S. Huang. Learning object detectors from scratch with gated recurrent feature pyramids. In *arXiv preprint arXiv:1712.00886*, 2017.

[176] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, 2016.

[177] K. Shmelkov, C. Schmid, and K. Alahari. Incremental learning of object detectors without catastrophic forgetting. In *ICCV*, 2017.

[178] A. Shrivastava and A. Gupta. Contextual priming and feedback for faster r-cnn. In *ECCV*, 2016.

[179] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *CVPR*, 2016.

[180] A. Shrivastava, R. Sukthankar, J. Malik, and A. Gupta. Beyond skip connection-s: Top-down modulation for object detection. In *arXiv preprint arXiv:1612.06851*, 2016.

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

[182] B. Singh and L. S. Davis. An analysis of scale invariance in object detection–snip. In *CVPR*, 2018.

[183] B. Singh, M. Najibi, and L. S. Davis. Sniper: Efficient multi-scale training. In *NeurIPS*, 2018.

[184] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, 2017.

[185] H. Su, S. Gong, and X. Zhu. Scalable deep learning logo detection. In *arXiv preprint arXiv:1803.11417*, 2018.

[186] H. Su, X. Zhu, and S. Gong. Deep learning logo detection with data expansion by synthesising context. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017.

[187] X. Sun, P. Wu, and S. C. Hoi. Face detection using deep learning: An improved faster rcnn approach. In *Neurocomputing*, 2018.

[188] Y. Sun, Y. Chen, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. In *NeurIPS*, 2014.

[189] Y. Sun, D. Liang, X. Wang, and X. Tang. Deepid3: Face recognition with very deep neural networks. In *arXiv preprint arXiv:1502.00873*, 2015.

[190] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[191] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, 2017.

[192] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[193] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

[194] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *arXiv preprint arXiv:1905.11946*, 2019.

[195] X. Tang, D. K. Du, Z. He, and J. Liu. Pyramidbox: A context-assisted single shot face detector. In *ECCV*, 2018.

[196] Z. Tian, C. Shen, H. Chen, and T. He. Fcos: Fully convolutional one-stage object detection. In *arXiv preprint arXiv:1904.01355*, 2019.

[197] L. Tychsen-Smith and L. Petersson. Denet: Scalable real-time object detection with directed sparse sampling. In *ICCV*, 2017.

[198] L. Tychsen-Smith and L. Petersson. Improving object localization with fitness nms and bounded iou loss. In *arXiv preprint arXiv:1711.00164*, 2017.

[199] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. In *IJCV*, 2013.

[200] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *ICCV*, 2009.

[201] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, 2016.

[202] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001.

[203] P. Viola and M. J. Jones. Robust real-time face detection. In *IJCV*, 2004.

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

[205] H. Wang, Q. Wang, M. Gao, P. Li, and W. Zuo. Multi-scale location-aware kernel representation for object detection. In *CVPR*, 2018.

[206] J. Wang, K. Chen, S. Yang, C. C. Loy, and D. Lin. Region proposal by guided anchoring. In *CVPR*, 2019.

[207] X. Wang, A. Shrivastava, and A. Gupta. A-fast-rcnn: Hard positive generation via adversary for object detection. In *CVPR*, 2017.

[208] X. Wang, T. Xiao, Y. Jiang, S. Shao, J. Sun, and C. Shen. Repulsion loss: Detecting pedestrians in a crowd. In *CVPR*, 2018.

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

[210] Y.-X. Wang, D. Ramanan, and M. Hebert. Meta-learning to detect rare objects. In *ICCV*, 2019.

[211] A. Wong, M. J. Shafiee, F. Li, and B. Chwyl. Tiny ssd: A tiny single-shot detection deep convolutional neural network for real-time embedded object detection. In *arXiv preprint arXiv:1802.06488*, 2018.

[212] S. Woo, S. Hwang, and I. S. Kweon. Stairnet: Top-down semantic aggregation for accurate one shot detection. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018.

[213] J. Wu, L. Cong, Y. Wang, Q. Hu, and J. Cheng. Quantized convolutional neural networks for mobile devices. In *CVPR*, 2016.

[214] X. Wu, D. Zhang, J. Zhu, and S. C. Hoi. Single-shot bidirectional pyramid networks for high-quality object detection. In *arXiv preprint arXiv:1803.08208*, 2018.

[215] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

[216] S. L. Xizhou Zhu, Han Hu and J. Dai. Deformable convnets v2: More deformable, better results. In *CVPR*, 2019.

[217] H. Xu, X. Lv, X. Wang, Z. Ren, and R. Chellappa. Deep regionlets for object detection. In *ECCV*, 2018.

[218] X. Yan, Z. Chen, A. Xu, X. Wang, X. Liang, and L. Lin. Meta r-cnn : Towards general solver for instance-level low-shot learning. In *ICCV*, 2019.

[219] B. Yang, J. Yan, Z. Lei, and S. Z. Li. Craft objects from images. In *CVPR*, 2016.

[220] F. Yang, W. Choi, and Y. Lin. Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In *CVPR*, 2016.

[221] S. Yang, P. Luo, C.-C. Loy, and X. Tang. From facial parts responses to face detection: A deep learning approach. In *ICCV*, 2015.

[222] S. Yang, P. Luo, C.-C. Loy, and X. Tang. Wwider face: A face detection benchmark. In *CVPR*, 2016.

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

[224] T. Yang, X. Zhang, Z. Li, W. Zhang, and J. Sun. Metaanchor: Learning to detect objects with customized anchors. In *NeurIPS*. 2018.

[225] Z. Yang, S. Liu, H. Hu, L. Wang, and S. Lin. Reppoints: Point set representation for object detection. In *ICCV*, 2019.

[226] B. Yu and D. Tao. Anchor cascade for efficient face detection. In *arXiv preprint arXiv:1805.03363*, 2018.

[227] Y. Yu, J. Zhang, Y. Huang, S. Zheng, W. Ren, C. Wang, K. Huang, and T. Tan. Object detection by context and boosted hog-lbp. In *PASCAL VOC Challenge*, 2010.

[228] S. Zagoruyko, A. Lerer, T.-Y. Lin, P. O. Pinheiro, S. Gross, S. Chintala, and P. Dollár. A multipath network for object detection. In *BMVC*, 2016.

[229] X. Zeng, W. Ouyang, B. Yang, J. Yan, and X. Wang. Gated bi-directional cnn for object detection. In *ECCV*, 2016.

[230] Y. Zhai, J. Fu, Y. Lu, and H. Li. Feature selective networks for object detection. In *CVPR*, 2018.

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

[232] J. Zhang, X. Wu, J. Zhu, and S. C. Hoi. Feature agglomeration networks for single stage face detection. In *arXiv preprint arXiv:1712.00721*, 2017.

[233] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. Jjoint face detection and alignment using multi-task cascaded convolutional networks. In *IEEE Signal Processing Letters*, 2016.

[234] K. Zhang, Z. Zhang, H. Wang, Z. Li, Y. Qiao, and W. Liu. Detecting faces using inside cascaded contextual cnn. In *ICCV*, 2017.

[235] S. Zhang, C. Chi, Y. Yao, Z. Lei, and S. Z. Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *CVPR*, 2020.

[236] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li. Single-shot refinement neural network for object detection. In *CVPR*, 2018.

[237] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li. S3fd: Single shot scale-invariant face detector. In *ICCV*, 2017.

[238] X. Zhang, F. Wan, C. Liu, R. Ji, and Q. Ye. FreeAnchor: Learning to match anchors for visual object detection. In *NeurIPs*, 2019.

[239] Z. Zhang, S. Qiao, C. Xie, W. Shen, B. Wang, and A. L. Yuille. Single-shot object detection with enriched semantics. In *CVPR*, 2018.

[240] Q. Zhao, T. Sheng, Y. Wang, Z. Tang, Y. Chen, L. Cai, and H. Ling. M2det: A single-shot object detector based on multi-level feature pyramid network. In *AAAI*, 2019.

[241] X. Zhao, S. Liang, and Y. Wei. Pseudo mask augmented object detection. In *CVPR*, 2018.

[242] P. Zhou, B. Ni, C. Geng, J. Hu, and Y. Xu. Scale-transferrable object detection. In *CVPR*, 2018.

[243] X. Zhou, D. Wang, and P. Krähenbühl. Objects as points. In *arXiv preprint arXiv:1904.07850*, 2019.

[244] X. Zhou, J. Zhuo, and P. Krahenbuhl. Bottom-up object detection by grouping extreme and center points. In *CVPR*, 2019.

[245] Y. Zhou, L. Liu, L. Shao, and M. Mellor. Dave: A unified framework for fast vehicle detection and annotation. In *ECCV*, 2016.

[246] C. Zhu, F. Chen, Z. Shen, and M. Savvides. Soft anchor-point object detection. *arXiv preprint arXiv:1911.12448*, 2019.

[247] C. Zhu, Y. He, and M. Savvides. Feature selective anchor-free module for single-shot object detection. In *CVPR*, 2019.

[248] C. Zhu, R. Tao, K. Luu, and M. Savvides. Seeing small faces from robust anchors perspective. In *CVPR*, 2018.

[249] C. Zhu, Y. Zheng, K. Luu, and M. Savvides. Cms-rcnn: Contextual multi-scale region-based cnn for unconstrained face detection. In *Deep Learning for Biometrics*. 2017.

[250] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networkss. In *ICCV*, 2017.

[251] R. Zhu, S. Zhang, X. Wang, L. Wen, H. Shi, L. Bo, and T. Mei. Scratchdet: Exploring to train single-shot object detectors from scratch. In *CVPR*, 2019.

[252] X. Zhu and D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *CVPR*, 2012.

[253] Y. Zhu, R. Urtasun, R. Salakhutdinov, and S. Fidler. segdeepm: Exploiting segmentation and context in deep neural networks for object detection. In *CVPR*, 2015.

[254] Y. Zhu, C. Zhao, J. Wang, X. Zhao, Y. Wu, and H. Lu. Couplenet: Coupling global structure with local parts for object detection. In *ICCV*, 2017.

[255] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu. Traffic-sign detection and classification in the wild. In *CVPR*, 2016.

[256] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*. 2014.

[257] B. Zoph, E. D. Cubuk, G. Ghiasi, T.-Y. Lin, J. Shlens, and Q. V. Le. Learning data augmentation strategies for object detection. In *arXiv preprint arXiv:1906.11172*, 2019.

[258] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018.