THE UNIVERSITY of ADELAIDE

Fully Convolutional Instance-level Visual Recognition

Zhi Tian

A thesis submitted for the degree of
DOCTOR OF PHILOSOPHY
The University of Adelaide

May 14, 2021

Contents

List of Figures

List of Tables

University of Adelaide

Abstract

Fully Convolutional Instance-level Visual Recognition

by Zhi Tian

Instance-level recognition such as object detection and instance segmentation are the fundamental problems in computer vision, which underpins many downstream computer vision applications. In this thesis, we propose a series of new methods to solve the problems with the simple and effective fully convolutional networks.

First, we propose a fully convolutional one-stage object detector (FCOS) to solve object detection in a per-pixel prediction fashion. Unlike previous detectors, FCOS completely avoids the complicated computation related to anchor boxes such as calculating the intersection over union (IoU) scores during training. More importantly, we also avoid all hyper-parameters related to anchor boxes. We demonstrate a much simpler and flexible detection framework achieving improved detection accuracy.

Second, we propose the first direct end-to-end multi-person pose estimation framework, termed DirectPose. The proposed framework directly predicts instance-aware keypoints for all the instances from a raw input image, eliminating the heuristic grouping in bottom-up methods or box detection and RoI operations in top-down ones.

Third, we propose a simple yet effective instance segmentation framework, termed CondInst (conditional convolutions for instance segmentation). Top-performing instance segmentation methods such as Mask R-CNN rely on ROI operations (typically ROIPool or ROIAlign) to obtain the final instance masks. In contrast, we propose to solve instance segmentation with dynamic instance-aware networks, conditioned on instances. For the first time, we demonstrate a simpler instance segmentation method that can achieve improved performance in both accuracy and inference speed.

Finally, we present a high-performance method that can achieve mask-level instance segmentation with only box annotations for training. Our core idea is to redesign the loss of learning masks in CondInst, with no modification to the network itself. The new loss functions can supervise the mask training without relying on mask annotations. Our excellent experiment results on COCO and Pascal VOC indicate that our method dramatically narrows the performance gap between weakly and fully supervised instance segmentation.

Codes are publicly available at `https://github.com/aim-uofa/AdelaiDet`.

Declaration of Authorship

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name, in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint-award of this degree.

I acknowledge that copyright of published works contained within this thesis resides with the copyright holder(s) of those works.

I also give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

I acknowledge the support I have received for my research through the provision of an Australian Government Research Training Program Scholarship.

Zhi Tian

May 14, 2021

Statement of Authorship

Title of Paper	BoxInst: High-Performance Instance Segmentation with Box Annotations
Publication Status	☐ Published ☐ Accepted for Publication ☑ Submitted for Publication ☐ Unpublished and Unsubmitted work written in manuscript style
Publication Details	in CVPR 2021 submission

Principal Author

Name of Principal Author (Candidate)	Zhi Tian		
Contribution to the Paper	Proposed the ideas, conducted experiments and wrote the manuscript of the paper.		
Overall percentage (%)	70%		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	11/01/2021

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);
ii. permission is granted for the candidate in include the publication in the thesis; and
iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Xinlong Wang		
Contribution to the Paper	Discussion, writing revision and conducting some experiments.		
Signature		Date	11/01/2021

Name of Co-Author	Hao Chen		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	27/10/2020

Name of Co-Author	Chunhua Shen
Contribution to the Paper	Discussion and writing revision.
Signature	

Date	21/01/2024

Statement of Authorship

Title of Paper	FCOS: A Simple and Strong Anchor-free Object Detector
Publication Status	☑ Published ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and Unsubmitted work written in manuscript style
Publication Details	Published in TPAMI 2020

Principal Author

Name of Principal Author (Candidate)	Zhi Tian
Contribution to the Paper	Proposed the ideas, conducted experiments and wrote the manuscript of the paper.
Overall percentage (%)	70%
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.
Signature	Date 11/01/2021

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);
ii. permission is granted for the candidate in include the publication in the thesis; and
iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Chunhua Shen
Contribution to the Paper	Discussion and writing revision.
Signature	Date 11/01/2021

Name of Co-Author	Hao Chen
Contribution to the Paper	Discussion and writing revision.
Signature	Date 27/10/2020

Name of Co-Author	Tong He		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	2021/1/11

Statement of Authorship

Title of Paper	Conditional Convolutions for Instance Segmentation
Publication Status	☑ Published ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and Unsubmitted work written in manuscript style
Publication Details	Published in ECCV 2020

Principal Author

Name of Principal Author (Candidate)	Zhi Tian		
Contribution to the Paper	Proposed the ideas, conducted experiments and wrote the manuscript of the paper.		
Overall percentage (%)	70%		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	11/01/2021

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);
ii. permission is granted for the candidate in include the publication in the thesis; and
iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Chunhua Shen		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	11/01/2021

Name of Co-Author	Hao Chen		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	27/10/2020

Statement of Authorship

Title of Paper	DirectPose: Direct End-to-End Multi-Person Pose Estimation
Publication Status	☐ Published ☐ Accepted for Publication ☐ Submitted for Publication ☑ Unpublished and Unsubmitted work written in manuscript style
Publication Details	The preprint available on ArXiv

Principal Author

Name of Principal Author (Candidate)	Zhi Tian		
Contribution to the Paper	Proposed the ideas, conducted experiments and wrote the manuscript of the paper.		
Overall percentage (%)	70%		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	11/01/2021

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

i. the candidate's stated contribution to the publication is accurate (as detailed above);
ii. permission is granted for the candidate in include the publication in the thesis; and
iii. the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Chunhua Shen		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	11/01/2021

Name of Co-Author	Hao Chen		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	27/10/2020

Acknowledgements

I have spent wonderful years at the University of Adelaide. First of all, I would appreciate my principal supervisor Prof. Chunhua Shen for his great help in these years. I am very lucky to have had such an awesome supervisor. He helped me improve my comprehensive research skills and also taught me many things other than research. Without his help, I cannot succeed in my PhD career.

I am also grateful to my co-authors and peers. Thank you for your constructive discussions with me. These discussions gave me much inspiration. They are Tong He, Hao Chen, Yuliang Liu, Rufeng Zhang, Bohan Zhuang, Zhipeng Cai, Qichang Hu, Hu Wang, Wei Yin, Yifan Liu, Libo Sun, Yunzhi Zhuge, Jiawang Bian, Xinyu Zhang, Dong Gong, Yang Zhao, Yutong Dai, Peng Chen, Xinlong Wang, Weian Mao, Bowen Zhang, Xinyu Wang and Rafael Felix. I also want to thank Google, LLC for their generous financial support with the Google PhD fellowship.

Moreover, thank the coffee machine in our lab. It has made thousands of cups of coffee for me. The great coffee makes me energetic when I am coding or writing papers. Last but not least, I would like to thank my family for their selfless support.

Publications

This thesis contains the following works that have been published or prepared for publication:

- FCOS: Fully Convolutional One-Stage Object Detection.
  Zhi Tian, Chunhua Shen, Hao Chen, Tong He.
  The International Conference on Computer Vision (ICCV), 2019.
- FCOS: A Simple and Strong Anchor-free Object Detector.
  Zhi Tian, Chunhua Shen, Hao Chen, Tong He.
  IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2020.
- DirectPose: Direct End-to-End Multi-Person Pose Estimation.
  Zhi Tian, Hao Chen, Chunhua Shen.
  ArXiv Preprint, 2020.
- Conditional Convolutions for Instance Segmentation.
  Zhi Tian, Chunhua Shen, Hao Chen.
  The European Conference on Computer Vision (ECCV), 2020.
- BoxInst: High-Performance Instance Segmentation with Box Annotations.
  Zhi Tian, Chunhua Shen, Xinlong Wang, Hao Chen.
  The Conference on Computer Vision and Pattern Recognition (CVPR), 2021.

In addition, I have the following papers not included in this thesis:

- Decoders Matter for Semantic Segmentation: Data-Dependent Decoding Enables Flexible Feature Aggregation.
  Zhi Tian, Tong He, Chunhua Shen, Youliang Yan.
  The Conference on Computer Vision and Pattern Recognition (CVPR), 2019.
- Learning and Memorizing Representative Prototypes for 3D Point Cloud Semantic and Instance Segmentation.
  Tong He, Dong Gong, **Zhi Tian**, Chunhua Shen.
  The European Conference on Computer Vision (ECCV), 2020.
- BlendMask: Top-Down Meets Bottom-Up for Instance Segmentation.
  Hao Chen, Kunyang Sun, **Zhi Tian**, Chunhua Shen, Yongming Huang, Youliang Yan.
  The Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

- NAS-FCOS: Fast Neural Architecture Search for Object Detection.
  Ning Wang, Yang Gao, Hao Chen, Peng Wang, **Zhi Tian**, Chunhua Shen.
  The Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

- Mask Encoding for Single Shot Instance Segmentation.
  Rufeng Zhang, **Zhi Tian**, Mingyu You, Chunhua Shen, Youliang Yan.
  The Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

- Convolutional Character Networks.
  Linjie Xing, **Zhi Tian**, Weilin Huang, Matthew Scott.
  The International Conference on Computer Vision (ICCV), 2019.

- Knowledge Adaptation for Efficient Semantic Segmentation.
  Tong He, Chunhua Shen, **Zhi Tian**, Dong Gong, Changming Sun, Youliang Yan.
  The Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

Chapter 1

Introduction

Instance-level visual recognition, one of the most fundamental tasks in computer vision, requires the computer algorithms to recognize each individual instance (*e.g.*, persons, cars, dogs, and cats) in an image. For each instance, the computer might be required to output a bounding-box tightly covering the entire instance, the keypoints of the instance, and/or the per-pixel mask of the instance. It results in three computer vision tasks — object detection, keypoint detection and instance segmentation, respectively. The targets of the three tasks are shown in Fig. 1.1. The three tasks are of great importance in modern computer vision systems, and also underpin many downstream applications such as tracking Wang et al., 2019, person re-identification Milan et al., 2016, and image retrieval Plummer et al., 2015, to name just a few.

There have been many different methods developed to solve these tasks. For example, for object detection, we have one-stage anchor-based detectors such as SSD Liu et al., 2016b, YOLO Redmon and Farhadi, 2017; Redmon and Farhadi, 2018, and RetinaNet Lin et al., 2017b, as well as two-stage region proposals based methods such as Fast R-CNN Girshick, 2015 and Faster R-CNN Ren et al., 2015. For instance segmentation, we have the region proposals based Mask R-CNN He et al., 2017. These methods have achieved unprecedented performance on these tasks.

Although these methods look totally different, at the core of these methods, they all rely on fully convolutional networks (FCNs) Long, Shelhamer, and Darrell, 2015 to obtain the dense features of an input image. Built on the dense features, different designs are used for these tasks. For example, anchor boxes and ROIPooling are used for object detection; ROIAlign He et al., 2017 is used for instance segmentation; and Part Affinity Fields (PAFs) Cao et al., 2017 are for keypoint detection. These task-specific operations require special coding implementation and optimization, and make these methods deviate from the neat fully convolutional frameworks. Recently, the fully convolutional frameworks have achieved tremendous success in dense prediction tasks such as semantic segmentation Long, Shelhamer, and Darrell, 2015; Tian et al., 2019a; He et al., 2019a, depth estimation Liu et al., 2016a; Yin et al., 2019, keypoint detection Chen et al., 2017b and counting. It is natural to ask a question: *Can we solve all these instance-level recognition tasks in the neat fully convolutional fashion,*

FIGURE 1.1. **Instance-level recognition tasks.** From left to right: object detection; keypoint detection; and instance segmentation.

analogue to FCNs for semantic segmentation Long, Shelhamer, and Darrell, 2015, for example? Thus those fundamental vision tasks can be unified in the fully convolutional framework. We show in this thesis that the answer is affirmative.

1.1 Object Detection

Among the three tasks, object detection is the simplest one and is often used as the first step in other two tasks. The basic idea to solve object detection is to exhaustively examine every possible location on an image with sliding-windows or region proposals. In the context of deep learning, the sliding-windows in traditional detectors have been transformed into anchor boxes, which are some initial boxes on the convolutional feature maps. Anchor boxes can take advantages of the sharing computation in convolutional neural networks (CNNs), which makes the detector tremendously faster and thus becomes very popular in object detection. All current mainstream detectors such as Faster R-CNN Ren et al., 2015, SSD Liu et al., 2016b and YOLOv2, v3 Redmon and Farhadi, 2018 rely on a set of pre-defined anchor boxes and *it has long been believed that the use of anchor boxes is the key to modern detectors' success.* Despite the great success, anchor-based detectors still suffer some drawbacks. For example, the scales and aspect ratios of anchor boxes are kept fixed once trained, and thus detectors encounter difficulties to deal with object candidates with large shape variations. In this thesis, we propose to solve object detection in the fashion of per-pixel prediction, eliminating the anchor boxes. Specifically, we directly predict a 4D vector plus a class category at each spatial location on a level of feature maps. The 4D vector depicts the relative offsets from the four sides of a bounding box to the location. The framework is similar to the FCNs for semantic segmentation, except that each location is required to regress a 4D continuous vector. We term the proposed detector fully convolutional one-stage object detector (FCOS). FCOS can bypass all of the drawbacks mentioned before since it does not use anchor boxes at all. It is the first anchor-free detector achieving improved detection performance than its anchor-based counterparts. Details are in Chapter 3.

1.2 Keypoint Detection

The key challenge in multi-person keypoint detection is how to obtain the *instance-level* keypoints. In other words, the detected keypoints need to be grouped according to the instance they belong to. Currently, the mainstream methods tackle this challenge with bottom-up or top-down methodologies. Top-down methods He et al., 2017; Cheng et al., 2020a employ ROI operations, which can make the FCNs only focus on a single instance in the second stage. The ROIs come with some drawbacks. For example, 1) ROIs are forwarded separately and cannot share the convolutional computation. Thus, the inference time of these methods heavily depends on the number of instances in the image, which impedes these methods from being real-time; 2) the top-down methods cannot be end-to-end trainable since the ROIs are often obtained from a person detector; 3) these ROI-based methods also rely on the localization quality of the ROIs, which may harm the keypoint detection performance. On the other hand, bottom-up methods Cao et al., 2017; Newell, Huang, and Deng, 2017 do not rely on ROIs. They first detect instance-agnostic keypoints and then employ a grouping post-processing to obtain the full-body results. The processing of assembling the keypoints is usually heuristic and involves many hyper-parameters, making these methods more complicated.

As mentioned before, in FCOS, we solve object detection by directly regressing two corners of a target bounding-box (*i.e.*, the four relative offsets). The straightforward and effective solution for object detection gives rise to a question: *can keypoint detection be solved with this simple framework as well?* It is easy to see that the keypoints for an instance can be considered as a special bounding-box with more than two corner points, and thus the task could be solved by attaching more output heads to the object detection networks. Based on this idea, we propose a direct end-to-end keypoint detection framework, termed DirectPose, which is fully convolutional and able to directly map a raw input image to the desired instance-aware keypoints. The keypoint alignment (KPAlign) module is also proposed to align the features and the keypoint predictions, improving the performance by a large margin. DirectPose can avoid the shortcomings of both top-down and bottom-up methods as it needs neither grouping or bounding-box detection. Moreover, it shares the same methodology as the detector FCOS, unifying the two tasks into the same pipeline. We will elaborate it in Chapter 4.

1.3 Instance Segmentation

Similar to keypoint detection, the key challenge of instance segmentation is also how to obtain instance-level results. We have shown before that, in keypoint detection, the challenge can be addressed by extending FCOS. However, it is hard to use the same idea for instance segmentation. Unlike the keypoints of an instance, which are structured (only several 2D coordinates) and can be parameterized as easily as

a box, the mask of an instance is a map, which are not structured and hard to be parameterized. In this thesis, instead of parameterizing the mask itself, we propose to use a set of filters to encode the mask, and the final mask can be decoded by applying the set of filters to the CNN's feature maps. The set of filters are dynamically generated by introducing a new branch to FCOS, which is similar to dynamic filter networks Jia et al., 2016 or conditional convolutions Yang et al., 2019a. We term our method conditional convolutions for instance segmentation (CondInst). CondInst eliminates the RoI operations of previous methods Lee and Park, 2020; He et al., 2017; Liu et al., 2018, thus being fully convolutional. CondInst also enjoys many other advantages over previous methods. Details will be introduced in Chapter 5.

1.4 Instance Segmentation with Box Annotations

CondInst has almost made the previously much more challenging instance segmentation task be as simple and fast as bounding-box object detection. For example, it only introduces less than 10% computation overhead, compared to FCOS. Instance segmentation is able to provide more accurate and fine *mask-level* object location than detection. Thus, given that the extra computation cost is negligible, instance segmentation should be preferred over bounding box detection in many cases. For example, if a robot wants to grasp an object, an accurate mask will be much more helpful than a box. Now the only obstacle that impedes instance segmentation replacing box detection is the significantly heavier pixel-wise mask annotations. Compared to box-level annotations required by object detection, annotating pixel-level masks is notoriously time-consuming, as shown in Bearman et al., 2016; Everingham et al., 2010; Kulharia et al., 2020. Here, we aim to eliminate this obstacle by training instance segmentation using box annotations only. We will show that with some clever designs, CondInst can be trained with only box annotations and yield high-quality instance masks. We term our method BoxInst and will introduce it in Chapter 6.

In summary, in this thesis, we propose three novel methods for object detection, keypoint detection and instance segmentation, respectively, and solve these challenging instance-level recognition tasks in the simple fully convolutional frameworks. Additionally, we also present BoxInst, which can obtain the high-quality instance masks with only box annotations, largely reducing the cost of the mask annotations in instance segmentation.

Chapter 2

Literature Review

Here we review some works that are closest to ours.

2.0.1 Object Detection

Anchor-based Detectors. Anchor-based detectors inherit the ideas from traditional sliding-window and proposal based detectors such as Fast R-CNN Girshick, 2015. In anchor-based detectors, the anchor boxes can be viewed as pre-defined sliding windows or proposals, which are classified as positive or negative patches, with an extra offsets regression to refine the prediction of bounding box locations. Therefore, the anchor boxes in these detectors may be viewed as *training samples*. Unlike previous detectors like R-CNN Girshick et al., 2014, which compute image features for each sliding window/proposal repeatedly, anchor boxes make use of the feature maps of CNNs and avoid repeated feature computation, speeding up detection process dramatically. The design of anchor boxes are popularized by Faster R-CNN in its RPNs Ren et al., 2015, SSD Liu et al., 2016b and YOLOv2 Redmon and Farhadi, 2017, and has become the convention in a modern detector.

However, as described before, anchor boxes result in excessively many hyper-parameters, which typically need to be carefully tuned in order to achieve good performance. Besides the hyper-parameters describing anchor shapes, the anchor-based detectors also need other hyper-parameters to label each anchor box as a positive, ignored or negative sample. Previous works often employ intersection over union (IOU) between anchor boxes and ground-truth boxes to determine the label of an anchor box (*e.g.*, a positive anchor if its IOU is in $[0.5, 1]$). These hyper-parameters have shown a great impact on the final accuracy, and require heuristic tuning. Meanwhile, these hyper-parameters are specific to detection tasks, making detection tasks deviate from a neat fully convolutional network architectures used in other dense prediction tasks such as semantic segmentation.

Anchor-free Detectors. The most popular anchor-free detector might be YOLOv1 Redmon et al., 2016. Instead of using anchor boxes, YOLOv1 predicts bounding boxes at points near the center of objects. Only the points near the center are used since they are considered to be able to produce higher-quality detection. However, since

only points near the center are used, YOLOv1 suffers from low recall as mentioned in YOLOv2 Redmon and Farhadi, 2017. As a result, YOLOv2 Redmon and Farhadi, 2017 employs anchor boxes as well. CornerNet Law and Deng, 2018 is a recently proposed one-stage anchor-free detector, which detects a pair of corners of a bounding box and groups them to form the final detected bounding box. CornerNet requires much more complicated post-processing to group the pairs of corners belonging to the same instance. An extra distance metric is learned for the purpose of grouping. Another family of anchor-free detectors such as Yu et al., 2016 are based on DenseBox Huang et al., 2015. The family of detectors have been considered unsuitable for generic object detection due to difficulty in handling overlapping bounding boxes and the recall being relatively low. In FCOS, we show that both problems can be largely alleviated with multi-level prediction (*i.e.*, FPNs). Moreover, we also show that together with our proposed center-ness branch, the much simpler detector can achieve much better detection performance than its anchor-based counterparts. Recently, FSAF Zhu, He, and Savvides, 2019 was proposed to employ an anchor-free detection branch as a complement to an anchor-based detection branch since they consider that a totally anchor-free detector cannot achieve good performance. They also make use of a feature selection module to improve the performance of the anchor-free branch, making the anchor-free detector have a comparable performance to its anchor-based counterpart. However, in this work, we surprisingly show that the totally anchor-free detector can actually obtain better performance than its anchor-based counterpart, without the need for the feature selection module in FSAF. Even more surprisingly, it can outperform the detectors in FSAF that combine the anchor-free and anchor-based designs. As a result, the long-standing anchor-boxes can be completely eliminated, making detection significantly simpler.

Some other anchor-free detectors were developed concurrently to FCOS. For example, CenterNet Zhou, Wang, and Krähenbühl, 2019 predicts the center, width and height of objects with hourglass networks Newell, Yang, and Deng, 2016, demonstrating promising performance. Compared to CenterNet, FCOS enjoys faster training, and has a better accuracy/speed trade-off. RepPoints Yang et al., 2019b represents the boxes by a set of points and uses converting functions to predict the target boxes. In contrast, FCOS is a more concise and straightforward method for box detection.

2.0.2 Keypoint Detection

Top-down Methods. Top-down methods Sun et al., 2019; Fang et al., 2017; Pishchulin et al., 2012; Gkioxari et al., 2014; Papandreou et al., 2017; Chen et al., 2018; Xiao, Wu, and Wei, 2018; Chen et al., 2017b break the multi-person pose estimation task into two sub-tasks – person detection and single-person pose estimation. The person detection predicts a bounding-box for each instance in the input image. Next, the instance is cropped from the original image and a single-person pose estimation is applied to predict the keypoints for the cropped instance. Moreover, some

approaches such as Mask R-CNN He et al., 2017 crop convolutional features rather than raw images, improving the efficiency of these methods. Top-down methods often have better performance but have higher computational complexity as it needs to repeatedly run the single-person pose estimation for each instance. Moreover, it also suffers from early commitment. In other words, it is difficult for these methods to recover an instance if it is missing in detection results.

Bottom-up Methods. In contrast to top-down methods, which first identify individual instances by a detector, bottom-up methods Cao et al., 2017; Pishchulin et al., 2016; Insafutdinov et al., 2016 first detect all possible keypoints in an instance-agnostic fashion. Afterwards, a grouping process is employed to assemble these keypoints into full-body keypoints. Bottom-up methods can take advantage of the sharing convolutional computation, thus being faster than top-down methods. However, the grouping process is heuristic and involves many tricks and hyper-parameters. Recently, a one-stage framework Nie et al., 2019 makes the grouping process simpler. Compared to this work, our end-to-end framework further reduces the design complexity of a human pose estimation framework by directly mapping an input image to the desired keypoints.

2.0.3 Instance Segmentation and Conditional Convolutions

Instance Segmentation. To date, the dominant framework for instance segmentation is still Mask R-CNN. Mask R-CNN first employs an object detector to detect the bounding-boxes of instances (*e.g.*, ROIs). With these bounding-boxes, an ROI operation is used to crop the features of the instance from the feature maps. Finally, a compact FCN head is used to obtain the desired instance masks. Many works Chen et al., 2019a; Liu et al., 2018; Huang et al., 2019 with top performance are built on Mask R-CNN. Moreover, some works have explored to apply FCNs to instance segmentation. InstanceFCN Dai et al., 2016 may be the first instance segmentation method that is fully convolutional. InstanceFCN proposes to predict position-sensitive score maps with vanilla FCNs. Afterwards, these score maps are assembled to obtain the desired instance masks. Note that InstanceFCN does not work well with overlapping instances. Others Neven et al., 2019; Newell, Huang, and Deng, 2017; Fathi et al., 2017 attempt to first perform segmentation and the desired instance masks are formed by assembling the pixels of the same instance. Novotny et al. Novotny et al., 2018 propose semi-convolutional operators to make FCNs applicable to instance segmentation. To our knowledge, thus far none of these methods can outperform Mask R-CNN both in accuracy and speed on the public COCO benchmark dataset.

Recently AdaptIS Sofiiuk, Barinova, and Konushin, 2019 proposes to solve panoptic segmentation with FiLM Perez et al., 2018. The idea shares some similarity with CondInst in that information about an instance is encoded in the coefficients generated by FiLM. Since only the batch normalization coefficients are dynamically generated, AdaptIS needs a large mask head to achieve good performance. In contrast, CondInst

directly encodes them into conv. filters of the mask head, thus having much stronger capacity. As a result, even with a very compact mask head, we believe that CondInst can achieve instance segmentation accuracy that would not be possible for AdaptIS to attain.

Conditional Convolutions. Unlike traditional convolutional layers, which have fixed filters once trained, the filters of conditional convolutions are conditioned on the input and are dynamically generated by another network (*i.e.*, a controller). This idea has been explored previously in dynamic filter networks Jia et al., 2016 and CondConv Yang et al., 2019a mainly for the purpose of increasing the capacity of a classification network. In CondInst, we extend this idea to solve the significantly more challenging task of instance segmentation.

2.0.4 Box-supervised Segmentation

Box-supervised Semantic Segmentation. A few works attempted to obtain semantic masks using box annotations. For example, BoxSup Dai, He, and Sun, 2015 uses the region proposals from MCG as the pseudo labels to train an FCN, and an iterative training algorithm is used to refine the estimated masks. The recent Box2Seg Kulharia et al., 2020 method employs the masks generated by GrabCut to supervise training of the mask prediction model. In addition, a per-class attention map is also predicted by the model to make the per-pixel cross entropy loss focus on foreground pixels and refine the segmentation boundaries. This method shows excellent performance on Pascal VOC Everingham et al., 2010. Song et al., 2019 propose to use the unsupervised CRF Krähenbühl and Koltun, 2011 to generate the segment proposals. Additionally, a class-wise filling rate loss to supervise the models for training, resulting in improved segmentation performance. One of the crucial steps in these methods is to employ the pseudo labels generated by unsupervised segmentation methods such as MCG Pont-Tuset et al., 2016 or GrabCut Rother, Kolmogorov, and Blake, 2004. This is because these method all rely on pixel-wise mask loss functions, thus not being able to work without mask annotations. In this work, we remove the dependency on pixel-wise mask losses, as a result, eliminating the region proposals. Our new loss functions ensure that mask prediction can still be *imperfectly* supervised without using any mask annotations.

Box-supervised Instance Segmentation. In the context of deep learning, instance segmentation with box annotations has not explored too much yet. SDI Khoreva et al., 2017 might be the first instance segmentation framework with box annotations. Similar to the methods for semantic segmentation, SDI also relies on the region proposals generated by MCG. Then they make use of an iterative training procedure to further refine the segmentation results. Recently, BBTP Hsu et al., 2019 formulates the box-supervised instance segmentation into a multiple instance learning (MIL) problem. BBTP is built on Mask R-CNN and samples the positive and negative bags according to the ROIs on CNN feature maps. In contrast, our method is built on the ROI-free

CondInst and employs our proposed projection loss term to supervise the mask learning, eliminating the need for sampling. BBTP also makes use of the pairwise term. However, their pairwise term is defined on the set containing all neighboring pixel pairs with the oversimplified assumption of spatially neighboring pixel pairs being encouraged to have the same label, inevitably introducing heavily noisy supervision. Our experiments show that, the heavily noisy supervision can have a negative impact on accuracy. In BoxInst, we take advantage of the crucial prior derived from proximal pixels' colors. As a result, we significantly outperform the mask AP of BBTP on COCO by an absolute 10%.

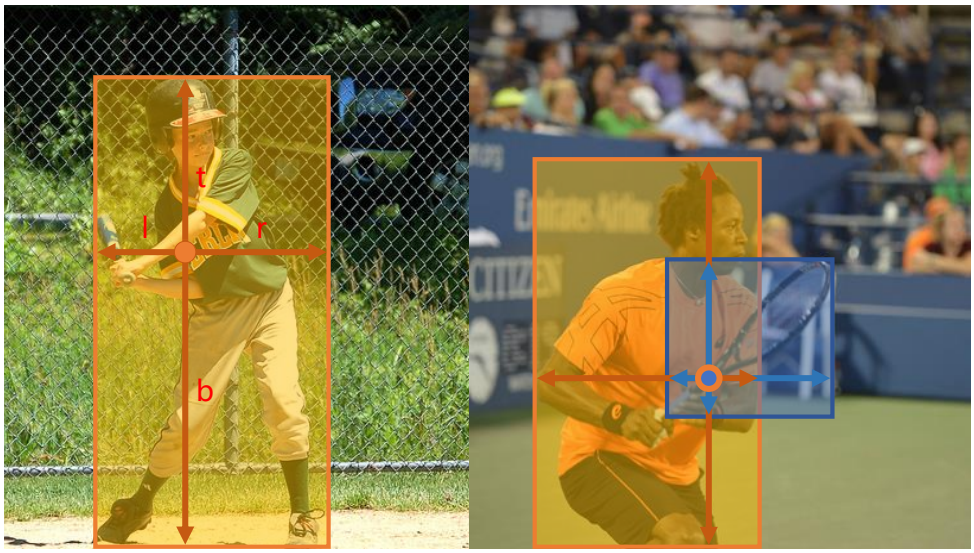

FIGURE 3.1. **Overall concept of FCOS**. As shown in the left image, FCOS works by predicting a 4D vector (l, t, r, b) encoding the location of a bounding box at each foreground pixel (supervised by ground-truth bounding box information during training). The right plot shows that when a location residing in multiple bounding boxes, it can be ambiguous in terms of which bounding box this location should regress.

Chapter 3

FCOS: Fully Convolutional One-Stage Object Detection

3.1 Introduction

Object detection requires an algorithm to predict a bounding box location and a category label for each instance of interest in an image. Prior to deep learning, the sliding-window approach was the main method Viola and Jones, 2001; Shen et al., 2013; Dollár et al., 2014, which exhaustively classifies every possible location, thus requiring feature extraction and classification evaluation to be very fast. With deep learning, detection has been largely shifted to the use of fully convolutional networks (FCNs) since the invention of Faster R-CNN Ren et al., 2015. All current mainstream detectors such as Faster R-CNN Ren et al., 2015, SSD Liu et al., 2016b and YOLOv2,

v3 Redmon and Farhadi, 2018 rely on a set of pre-defined anchor boxes and *it has long been believed that the use of anchor boxes is the key to modern detectors' success.* Despite their great success, it is important to note that anchor-based detectors suffer some drawbacks:

- As shown in Faster R-CNN and RetinaNet Lin et al., 2017b, detection performance is sensitive to the sizes, aspect ratios and number of anchor boxes. For example, in RetinaNet, varying these hyper-parameters affects the performance up to 4% in AP on the COCO benchmark Lin et al., 2014. As a result, these hyper-parameters need to be carefully tuned in anchor-based detectors.
- Even with careful design, because the scales and aspect ratios of anchor boxes are kept fixed, detectors encounter difficulties to deal with object candidates with large shape variations, particularly for small objects. The pre-defined anchor boxes also hamper the generalization ability of detectors, as they need to be re-designed on new detection tasks with different object sizes or aspect ratios.
- In order to achieve a high recall rate, an anchor-based detector is required to densely place anchor boxes on the input image (*e.g.*, more than 180K anchor boxes in feature pyramid networks (FPN) Lin et al., 2017a for an image with its shorter side being 800). Most of these anchor boxes are labeled as negative samples during training. The excessive number of negative samples aggravates the imbalance between positive and negative samples in training.
- Anchor boxes also involve complicated computation such as calculating the intersection-over-union (IoU) scores with ground-truth bounding boxes.

Recently, per-pixel prediction FCNs Long, Shelhamer, and Darrell, 2015 have achieved tremendous success in dense prediction tasks such as semantic segmentation Long, Shelhamer, and Darrell, 2015; Tian et al., 2019a; He et al., 2019a, depth estimation Liu et al., 2016a; Yin et al., 2019, keypoint detection Chen et al., 2017b and counting. As one of high-level vision tasks, object detection might be the only one deviating from the neat fully convolutional per-pixel prediction framework mainly due to the use of anchor boxes.

It is natural to ask a question: *Can we solve object detection in the neat per-pixel prediction fashion, analogue to FCN for semantic segmentation, for example?* Thus those fundamental vision tasks can be unified in (almost) one single framework. We show in this work that the answer is affirmative. Moreover, we demonstrate that, the much simpler FCN-based detector can surprisingly achieve even better performance than its anchor-based counterparts.

In the literature, some works attempted to leverage the per-pixel prediction FCNs for object detection such as DenseBox Huang et al., 2015. Specifically, these FCN-based frameworks directly predict a 4D vector plus a class category at each spatial location on a level of feature maps. As shown in Fig. 3.1 (left), the 4D vector depicts the relative

offsets from the four sides of a bounding box to the location. These frameworks are similar to the FCNs for semantic segmentation, except that each location is required to regress a 4D continuous vector. However, to deal with bounding boxes with different sizes, DenseBox Huang et al., 2015 crops and resizes training images to a fixed scale. Thus, DenseBox has to perform detection on image pyramids, which is against FCN's philosophy of computing all convolutions once.

Besides, more significantly, these methods are mainly used in special domain objection detection such as scene text detection Zhou et al., 2017; He et al., 2018 or face detection Yu et al., 2016; Huang et al., 2015, since it is believed that these methods do not work well when applied to generic object detection with highly overlapped bounding boxes. As shown in Fig. 3.1 (right), the highly overlapped bounding boxes result in an intractable ambiguity: it is not clear w.r.t. which bounding box to regress for the pixels in the overlapped regions.

In the sequel, we take a closer look at the issue and show that with FPN this ambiguity can be largely eliminated. As a result, our method can already obtain similar or even better detection accuracy with those traditional anchor based detectors. Furthermore, we observe that our method may produce a number of low-quality predicted bounding boxes at the locations that are far from the center of an target object. It is easy to see that the locations near the center of its target bounding box can make more reliable predictions. As a result, we introduce a novel "center-ness" score to depict the deviation of a location to the center, as defined in Eq. (3.3), which is used to down-weigh low-quality detected bounding boxes and thus helps to suppress these low-quality detections in NMS. The center-ness score is predicted by a branch (only one layer) in parallel with the bounding box regression branch, as shown in Fig. 3.2. The simple yet effective center-ness branch remarkably improves the detection performance with a negligible increase in computational time.

This new detection framework enjoys the following advantages.

- Detection is now unified with many other FCN-solvable tasks such as semantic segmentation, making it easier to re-use ideas from those tasks. An example is shown in Liu et al., 2020b, where a structured knowledge distillation method was developed for dense prediction tasks. Thanks to the standard FCN framework of FCOS, the developed technique can be immediately applied to FCOS based object detection.

- Detection becomes proposal free and anchor free, which significantly reduces the number of design parameters. The design parameters typically need heuristic tuning and many tricks are involved in order to achieve good performance. Therefore, our new detection framework makes the detector, particularly its training, *considerably* simpler.

- By eliminating the anchor boxes, our new detector completely avoids the complicated computation related to anchor boxes such as the IOU computation and

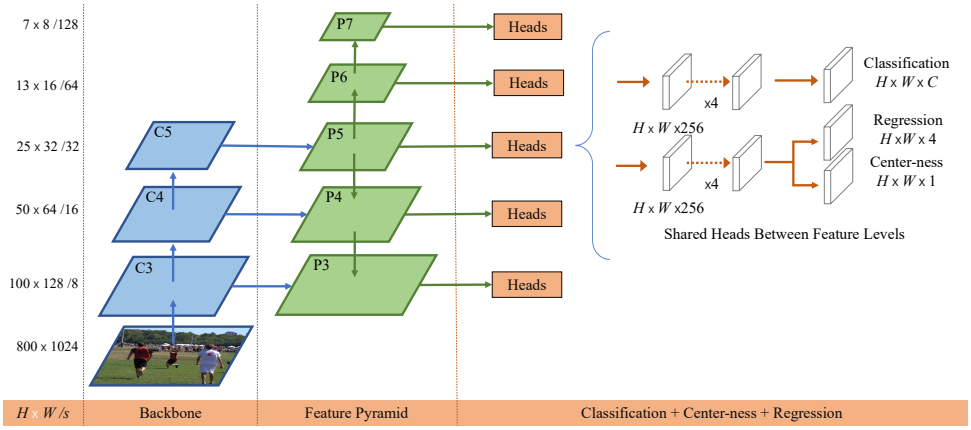

FIGURE 3.2. **The network architecture of FCOS**, where C3, C4, and C5 denote the feature maps of the backbone network and P3 to P7 are the feature levels used for the final prediction. $H \times W$ is the height and width of feature maps. '$/s$' ($s = 8, 16, ..., 128$) is the down-sampling ratio of the feature maps at the level to the input image. As an example, all the numbers are computed with an 800×1024 input.

matching between the anchor boxes and ground-truth boxes during training, resulting in faster training and testing than its anchor-based counterpart.

- Without bells and whistles, we achieve state-of-the-art results among one-stage detectors. Given its improved accuracy of the much simpler anchor-free detector, *we encourage the community to rethink the necessity of anchor boxes in object detection*, which are currently considered as the *de facto* standard for designing detection methods.

- With considerably reduced design complexity, our proposed detector outperforms previous strong baseline detectors such as Faster R-CNN Ren et al., 2015, RetinaNet Lin et al., 2017b, YOLOv3 Redmon and Farhadi, 2018 and SSD Liu et al., 2016b. More importantly, due to its simple design, FCOS can be easily extended to solve other instance-level recognition tasks with minimal modification, as already evidenced by instance segmentation Xie et al., 2020; Zhang et al., 2020; Lee and Park, 2020; Chen et al., 2020, keypoint detection Tian, Chen, and Shen, 2019, text spotting Liu et al., 2020c, and tracking Wang et al., 2020a; Guo et al., 2020. We expect to see more instance recognition methods built upon FCOS.

3.2 Our Approach

In this section, we first reformulate object detection in a per-pixel prediction fashion. Next, we show that how we make use of multi-level prediction to improve the recall and resolve the ambiguity resulted from overlapped bounding boxes. Finally, we present our proposed "center-ness" branch, which helps suppress the low-quality detected bounding boxes and improves the overall performance by a large margin.

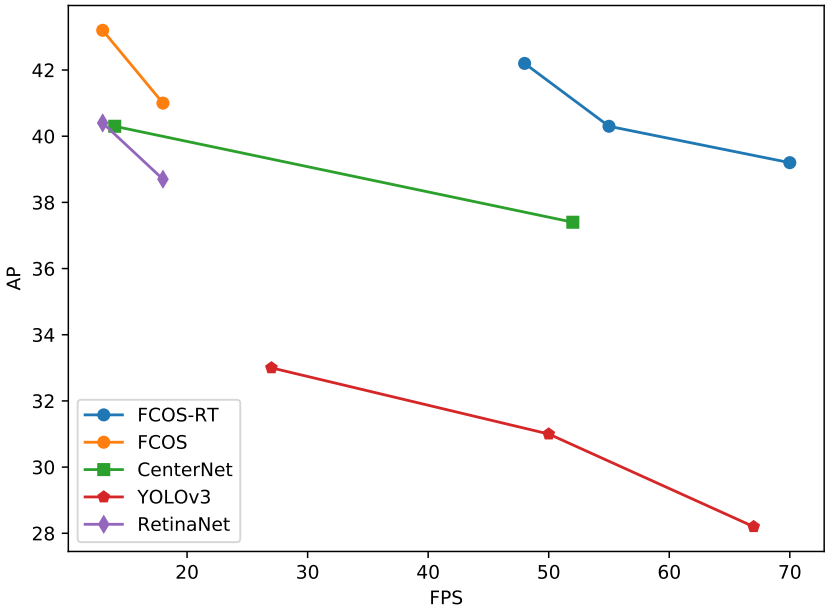

FIGURE 3.3. **Speed/accuracy trade-off between FCOS and several recent methods**: CenterNet Zhou, Wang, and Krähenbühl, 2019, YOLOv3 Redmon and Farhadi, 2018 and RetinaNet Lin et al., 2017b. Speed is measured on a NVIDIA 1080Ti GPU.

3.2.1 Fully Convolutional One-Stage Object Detector

Let $F_i \in \mathbb{R}^{H \times W \times C}$ be the feature maps at layer i of a backbone CNN and s be the total stride until the layer. The ground-truth bounding boxes for an input image are defined as $\{B_i\}$, where $B_i = (x_0^{(i)}, y_0^{(i)}, x_1^{(i)} y_1^{(i)}, c^{(i)}) \in \mathbb{R}^4 \times \{1, 2 \ldots C\}$. Here $(x_0^{(i)}, y_0^{(i)})$ and $(x_1^{(i)} y_1^{(i)})$ denote the coordinates of the left-top and right-bottom corners of the bounding box. $c^{(i)}$ is the class that the object in the bounding box belongs to. C is the number of classes, which is 80 for the MS-COCO dataset.

For each location (x, y) on the feature map F_i, we can map it back onto the input image as $(\lfloor \frac{s}{2} \rfloor + xs, \lfloor \frac{s}{2} \rfloor + ys)$, which is near the center of the receptive field of the location (x, y). Different from anchor-based detectors, which consider the location on the input image as the center of (multiple) anchor boxes and regress the target bounding box with these anchor boxes as references, we directly regress the target bounding box at the location. In other words, our detector directly views locations as *training samples* instead of anchor boxes in anchor-based detectors, which is the same as FCNs for semantic segmentation Long, Shelhamer, and Darrell, 2015.

Specifically, location (x, y) is considered as a positive sample if it falls into the center area of any ground-truth box, by following 2019. The center area of a box centered at (c_x, c_y) is defined as the sub-box $(c_x - rs, c_y - rs, c_x + rs, c_y + rs)$, where s is the total stride until the current feature maps and r is a hyper-parameter being 1.5 on COCO. The sub-box is clipped so that it is not beyond the original box. Note that this is different from our original conference version Tian et al., 2019b, where we consider the

locations positive as long as they are in a ground-truth box. The class label c^* of the location is the class label of the ground-truth box. Otherwise it is a negative sample and $c^* = 0$ (background class). Besides the label for classification, we also have a 4D real vector $\boldsymbol{t}^* = (l^*, t^*, r^*, b^*)$ being the regression targets for the location. Here l^*, t^*, r^* and b^* are the distances from the location to the four sides of the bounding box, as shown in Fig. 3.1 (left). If a location falls into the center area of multiple bounding boxes, it is considered as an *ambiguous sample*. We simply choose the bounding box with minimal area as its regression target. In the next section, we will show that with multi-level prediction, the number of ambiguous samples can be reduced significantly and thus they hardly affect the detection performance. Formally, if location (x, y) is associated to a bounding box B_i, the training regression targets for the location can be formulated as,

$$\begin{aligned} l^* &= (x - x_0^{(i)})/s, \quad t^* = (y - y_0^{(i)})/s, \\ r^* &= (x_1^{(i)} - x)/s, \quad b^* = (y_1^{(i)} - y)/s, \end{aligned} \tag{3.1}$$

where s is the total stride until the feature maps F_i, which is used to scale down regression targets and prevents the gradients from exploding during training. Together with these designs, FCOS can detect objects in an anchor-free fashion, and everything is learned by the networks without the need for any pre-defined anchor-boxes. *It is worth noting that this is not identical to an anchor-based detector with one anchor-box per location*, and the crucial difference is the way we define positive and negative samples. The single-anchor detector still uses pre-defined anchor-boxes as a prior and uses IoUs between the anchor-boxes and ground-truth boxes to determine the labels for these anchor-boxes. In FCOS, we remove the need for the prior and the locations are labeled by their inclusion in ground-truth boxes. In the experiments, we show that using a single anchor can only achieve inferior performance.

Network Outputs. Corresponding to the training targets, the final layer of our networks predicts an 80D vector $\boldsymbol{p}$ for classification and a 4D vector $\boldsymbol{t} = (l, t, r, b)$ encoding bounding-box coordinates. Following Lin et al., 2017b, instead of training a multi-class classifier, we train C binary classifiers. Similar to Lin et al., 2017b, we add two branches, respectively with four convolutional layers (exclude the final prediction layers) after the feature maps produced by FPNs for classification and regression tasks, respectively. Moreover, since the regression targets are always positive, we employ $\text{ReLU}(x)$ to map any real number to $(0, \infty)$ on the top of the regression branch. *It is worth noting that FCOS has* $9\times$ *fewer network output variables than the popular anchor-based detectors Lin et al., 2017b; Ren et al., 2015 with 9 anchor boxes per location,* which is of great importance when FCOS is applied to keypoint detection Tian, Chen, and Shen, 2019 or instance segmentation Tian, Shen, and Chen, 2020.

Loss Function. We define our training loss function as follows:

$$L(\{\boldsymbol{p}_{x,y}\}, \{\boldsymbol{t}_{x,y}\}) = \frac{1}{N_{\text{pos}}} \sum_{x,y} L_{\text{cls}}(\boldsymbol{p}_{x,y}, c^*_{x,y}) + \frac{\lambda}{N_{\text{pos}}} \sum_{x,y} \mathbb{1}_{\{c^*_{x,y}>0\}} L_{\text{reg}}(\boldsymbol{t}_{x,y}, \boldsymbol{t}^*_{x,y}), \tag{3.2}$$

where L_{cls} is focal loss as in Lin et al., 2017b and L_{reg} is the GIoU loss Rezatofighi et al., 2019. As shown in experiments, the GIoU loss achieves better performance than the IoU loss in UnitBox Yu et al., 2016, which is used in our preliminary version Tian et al., 2019b. N_{pos} denotes the number of positive samples and λ being 1 in this paper is the balance weight for L_{reg}. The summation is calculated over all locations on the feature maps F_i. $\mathbb{1}_{\{c_i^*>0\}}$ is the indicator function, being 1 if $c_i^* > 0$ and 0 otherwise.

Inference. Performing inference for FCOS is straightforward. Given an input images, we forward it through the network and obtain the classification scores $\boldsymbol{p}_{x,y}$ and the regression prediction $\boldsymbol{t}_{x,y}$ for each location on the feature maps F_i. Following Lin et al., 2017b, we choose the location with $p_{x,y} > 0.05$ as positive samples and invert Eq. (3.1) to obtain the predicted bounding boxes.

3.2.2 Multi-level Prediction with FPN for FCOS

Here we show that how two possible issues of the proposed FCOS can be resolved with multi-level prediction with FPN Lin et al., 2017a.

First, the large stride (*e.g.*, 16×) of the final feature maps in a CNN can result in a relatively low *best possible recall (BPR)*[1]. For anchor based detectors, low recall rates due to the large stride can be compensated to some extent by lowering the IOU score requirements for positive anchor boxes. For FCOS, at the first glance one may think that the BPR can be much lower than anchor-based detectors because it is impossible to recall an object which no location on the final feature maps encodes due to a large stride. Here, we empirically show that even with a large stride, FCOS is still able to produce a good BPR, and it can even better than the BPR of the anchor-based detector RetinaNet Lin et al., 2017b in the official implementation Detectron Girshick et al., 2018 (refer to Table 3.1). Therefore, the BPR is actually not a problem of FCOS. Moreover, with multi-level FPN prediction Lin et al., 2017a, the BPR can be improved further to match the best BPR the anchor-based RetinaNet can achieve.

Second, as shown in Fig. 3.1 (right), overlaps in ground-truth boxes can cause intractable ambiguity, *i.e.*, which bounding box should a location in the overlap regress? This ambiguity results in degraded performance. In this work, we show that the ambiguity can be greatly resolved with multi-level prediction (and center sampling), and FCOS can obtain *on par*, sometimes even better, performance compared with anchor-based ones.

Specifically, following FPN Lin et al., 2017a, we detect different size objects on different feature map levels. we make use of five levels of feature maps defined as $\{P_3, P_4, P_5, P_6, P_7\}$. As shown in Fig. 3.2, P_3, P_4 and P_5 are produced by the backbone CNNs' feature maps C_3, C_4 and C_5 with the top-down connections as in Lin et al., 2017a. P_6 and P_7 are produced by applying one 3×3 convolutional layer with the

[1]Upper bound of the recall rate that a detector can achieve.

stride being 2 on P_5 and P_6, respectively. Note that this is different from the original RetinaNet, which obtain P_6 and P_7 from the backbone feature maps C_5. We find both schemes achieve similar performance but the one we use has fewer parameters. Moreover, the feature levels P_3, P_4, P_5, P_6 and P_7 have strides 8, 16, 32, 64 and 128, respectively.

Anchor-based detectors assign different scale anchor boxes to different feature levels. Since anchor boxes and ground-truth boxes are associated by their IoU scores, this naturally enables different FPN feature levels to detect objects of different scales. However, *this couples the sizes of anchor boxes and the target object sizes of each FPN level*, which is problematic. The anchor box sizes should be data-specific, which might be changed from one dataset to another. The target object sizes of each FPN level should depend on the receptive field of the FPN level, which depends on the network architecture. FCOS removes the coupling as we only need focus on the target object sizes of each FPN level and need not design the anchor box sizes. Unlike anchor-based detectors, in FCOS, we directly limit the range of bounding box regression for each level. More specifically, we first compute the regression targets l^*, t^*, r^* and b^* for each location on all feature levels. Next, if a location at feature level i satisfies $\max(l^*, t^*, r^*, b^*) \leq m_{i-1}$ or $\max(l^*, t^*, r^*, b^*) \geq m_i$, it is set as a negative sample and thus not required to regress a bounding box anymore. Here m_i is the maximum distance that feature level i needs to regress. In this work, m_2, m_3, m_4, m_5, m_6 and m_7 are set as 0, 64, 128, 256, 512 and ∞, respectively. We argue that bounding the maximum distance is a better approach to determine the range of target objects for each feature level because this makes sure that the complete objects are always in the receptive field of each feature level. Moreover, since objects of different sizes are assigned to different feature levels and overlapping mostly happens between objects with considerably different sizes, the aforementioned ambiguity can be largely alleviated. If a location, even with multi-level prediction used, is still assigned to more than one ground-truth boxes, we simply choose the ground-truth box with minimal area as its target. As shown in our experiments, with the multi-level prediction, both anchor-free and anchor-based detectors can achieve the same level performance.

Finally, following Lin et al., 2017a; Lin et al., 2017b, we share the heads between different feature levels, not only making the detector parameter-efficient but also improving the detection performance. However, we observe that different feature levels are required to regress different size range (*e.g.*, the size range is $[0, 64]$ for P_3 and $[64, 128]$ for P_4), and therefore it may not be the optimal design to make use of identical heads for different feature levels. In our preliminary version Tian et al., 2019b, this issue is addressed by multiplying a learnable scalar to the convolutional layer's outputs. In this version, since the regression targets are scaled down by the stride of FPN feature levels, as shown in Eq. (3.1), the scalars become less important. However, we still keep them for compatibility.

3.2.3 Center-ness for FCOS

With the multi-level prediction, FCOS can already achieve better performance than its anchor-based counterpart RetinaNet. Furthermore, we have observed that there are many low-quality detections produced by the locations that are far away from the center of an object.

Here we propose a simple yet effective strategy to suppress these low-quality detections. Specifically, we add *a single-layer branch*, in parallel with the regression branch (as shown in Fig. 3.2) to predict the "center-ness" of a location. The center-ness depicts the normalized distance from the location to the center of the object that the location is responsible for, as shown in Fig. 3.4. Given the regression targets l^*, t^*, r^* and b^* for a location, the center-ness target is defined as,

$$\text{centerness}^* = \sqrt{\frac{\min(l^*, r^*)}{\max(l^*, r^*)} \times \frac{\min(t^*, b^*)}{\max(t^*, b^*)}}. \tag{3.3}$$

We employ $\sqrt{\cdot}$ here to slow down the decay of the center-ness. The center-ness ranges from 0 to 1 and can thus be trained with binary cross entropy (BCE) loss. This loss is added to the loss function in Eq. (3.2). During testing, the final score $\boldsymbol{s}_{x,y}$ (it is used for ranking the detections in NMS) is the square root of the product of the predicted center-ness $o_{x,y}$ and the corresponding classification score $\boldsymbol{p}_{x,y}$. Formally,

$$\boldsymbol{s}_{x,y} = \sqrt{\boldsymbol{p}_{x,y} \cdot o_{x,y}}, \tag{3.4}$$

where $\sqrt{\cdot}$ is used to calibrate the order of magnitude of the final score and has no effect on average precision (AP).

Consequently, center-ness can down-weight the scores of bounding boxes far from the center of an object. As a result, with high probability, these low-quality bounding boxes might be filtered out by the final non-maximum suppression (NMS) process, improving the detection performance *remarkably*.

3.3 Experiments

Our experiments are conducted on the large-scale detection benchmark COCO Lin et al., 2014. Following the common practice Lin et al., 2017b; Lin et al., 2017a; Ren et al., 2015, we use the COCO `train2017` split (115K images) for training and `val2017` split (5K images) as validation for our ablation study. We report our main results on the `test`-dev split (20K images) by uploading our detection results to the evaluation server.

Training Details. Unless otherwise specified, we use the following implementation details. ResNet-50 He et al., 2016 is used as our backbone networks and the same hyper-parameters with RetinaNet Lin et al., 2017b are used. Specifically, our network

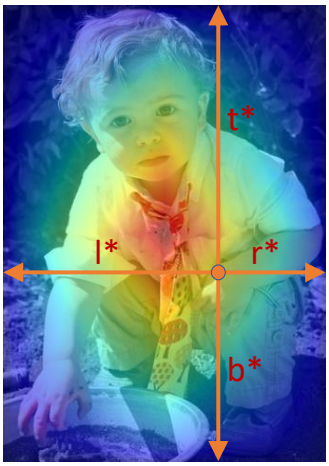

FIGURE 3.4. **Center-ness.** Red, blue, and other colors denote 1, 0 and the values between them, respectively. Center-ness is computed using Eq. (3.3) and decays from 1 to 0 as the location deviates from the center of the object.

method	w/ FPN	Low-quality matches	BPR (%)
RetinaNet	✓	Not used	88.16
RetinaNet	✓	≥ 0.4	91.94
RetinaNet	✓	All	**99.32**
FCOS		-	96.34
FCOS	✓	-	98.95

TABLE 3.1. The best possible recall (BPR) of anchor-based RetinaNet under a variety of matching rules and the BPR of FCOS on the COCO `val2017` split.

is trained with stochastic gradient descent (SGD) for 90k iterations with the initial learning rate being 0.01 and a mini-batch of 16 images. The learning rate is reduced by a factor of 10 at iteration 60k and 80k, respectively. Weight decay and momentum are set as 0.0001 and 0.9, respectively. We initialize our backbone networks with the weights pre-trained on ImageNet Deng et al., 2009. For the newly added layers, we initialize them as in Lin et al., 2017b. Unless specified, the input images are resized to have their shorter side being 800 and their longer side less or equal to 1333.

Inference Details. We firstly forward the input image through the network and obtain the predicted bounding boxes with the predicted class scores. The next post-processing of FCOS exactly follows that of RetinaNet Lin et al., 2017b. The post-processing hyper-parameters are also the same except that we use NMS threshold 0.6 instead of 0.5 in RetinaNet. We carry out experiments to show the impact of the NMS threshold in the sequel. Moreover, we use the same sizes of input images as in training.

w/ ctr. sampling	w/ FPN	1	2	≥ 3
		76.60%	20.05%	3.35%
	✓	92.58%	6.97%	0.45%
✓		96.52%	3.34%	0.14%
✓	✓	97.34%	2.59%	0.07%

TABLE 3.2. The ratios of the ambiguous samples to all the positive samples in FCOS. $1, 2$ and ≥ 3 denote the number of ground-truth boxes a location should be associated to. If the number is greater than 1, the location is defined as an "ambiguous sample" in this work.

method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AR_1	AR_{10}	AR_{100}
RetinaNet (#A=9)	35.9	55.8	38.4	20.6	39.8	46.6	31.0	49.8	53.0
RetinaNet (#A=1) w/ imprv.	35.2	55.6	37.0	19.9	39.2	45.2	30.4	49.9	53.5
RetinaNet (#A=9) w/ imprv.	37.6	56.6	40.6	21.5	42.1	48.0	32.1	52.2	56.4
FCOS w/o ctr.-ness	38.0	57.2	40.9	21.5	42.4	49.1	32.1	52.4	56.2
FCOS w/ ctr.-ness	**38.9**	**57.5**	**42.2**	**23.1**	**42.7**	**50.2**	**32.4**	**53.8**	**57.5**

TABLE 3.3. **FCOS vs. RetinaNet** on `val2017` split with ResNet-50-FPN as the backbone. All experiments use the same training settings. The proposed anchor-free FCOS achieves even better performance than anchor-based RetinaNet. #A: the number of anchors per location. RetinaNet (#A=9): the original RetinaNet from `Detectron2` Wu et al., 2019. RetinaNet w/ imprv. RetinaNet with the universal improvements in FCOS including Group Normalization (GN) Wu and He, 2018, GIoU loss 2019 and scalars in regression, using P_5 instead of C_5 and NMS threshold 0.6 instead of 0.5. We have tried our best to make all the details consistent. As shown the table, even without the center-ness branch, the much simpler FCOS already outperforms "RetinaNet (#A=9) w/ imprv" by 0.4% in AP. With the center-ness branch, the performance is further improved to 38.9% in AP.

3.3.1 Analysis of FCOS

Best Possible Recall (BPR) of FCOS

We first address the concern that is FCOS might not provide a good best possible recall (BPR) (*i.e.*, upper bound of the recall rate). In the section, we show that the concern is not necessary by comparing BPR of FCOS and that of its anchor-based counterpart RetinaNet on the COCO `val2017` split. The following analyses are based on the FCOS implementation in `git.io/AdelaiDet`.

Formally, BPR is defined as the ratio of the number of ground-truth boxes that a detector can recall at the most to the number of all ground-truth boxes. A ground-truth box is considered recalled if the box is assigned to at least one training sample (*i.e.*, a location in FCOS or an anchor box in anchor-based detectors), and a training sampling can be associated to *at most* one ground-truth box. As shown in Table 3.1, both with FPN, FCOS and RetinaNet obtain similar BPR (98.95 vs 99.32). Due to the fact that the best recall of current detectors are much lower than 90%, the small BPR gap (less than 0.5%) between FCOS and the anchor-based RetinaNet will not actually affect the performance of a detector. It is also confirmed in Table 3.3, where FCOS achieves better or similar AR than RetinaNet under the same training

and testing settings. Even more surprisingly, only with feature level P_4 with stride being 16 (*i.e.*, no FPN), FCOS can obtain a decent BPR of 96.34%. The BPR is much higher than the BPR of 91.94% of the RetinaNet in the official implementation `Detectron` Girshick et al., 2018, where only the low-quality matches with IOU ≥ 0.4 are used. Therefore, the concern about low BPR may not be necessary.

Ambiguous Samples in FCOS

The second concern about the FCN-based detector may be that it may have a large number of *ambiguous samples* due to the overlap in ground-truth boxes, as shown in Fig. 3.1 (right). In Table 3.2, we show the ratios of the ambiguous samples to all positive samples on the `val2017` split. If a location should be associated to multiple ground-truth boxes without using the rule of choosing the box with the minimum area, the location is defined as an "ambiguous sample". As shown in the table, there are indeed a large amount of ambiguous samples (23.40%) if the FPN is not used (*i.e.*, only P_4 used). However, with FPN, the ratio can be significantly reduced to only 7.42% since most of overlapped objects are assigned to different feature levels. Furthermore, if the center sampling is employed, the number of ambiguous samples can be further reduced. As shown in Table 3.2, even without FPN, the ratio is only 3.48%. By further applying FPN, the ratio is reduced to 2.66%. *Note that it does not imply that there are* 2.66% *locations where FCOS makes mistakes.* As mentioned earlier, these locations are associated with the smallest one among the ground-truth boxes associated to the same location. Therefore, these locations only take the risk of missing some larger objects. In other words, it may negatively impact the recall of FCOS. However, as shown in Table 3.1, the recall gap between FCOS and RetinaNet is negligible, which suggests that the ratio of the missing objects is extremely low.

The Effect of Center-ness

We have proposed "center-ness" to suppress the low-quality detected bounding boxes produced by the locations that are far from the center of an object. As shown in Table 3.4, the center-ness branch can boost AP from 38.0% to 38.9%. Compared to our conference version Tian et al., 2019b, the gap is relatively smaller since we make

	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
w/o ctr.-ness	38.0	57.2	40.9	21.5	42.4	49.1
w/ ctr.-ness†	37.5	56.5	40.2	21.6	41.5	48.5
w/ ctr.-ness (L1)	**38.9**	**57.6**	42.0	23.0	42.3	**51.0**
w/ ctr.-ness	**38.9**	57.5	**42.2**	**23.1**	**42.7**	**50.2**

TABLE 3.4. **Ablation study for the proposed center-ness branch** on the `val2017` split. ctr.-ness†: using the center-ness computed from the predicted regression vector when testing (*i.e.*, replacing the ground-truth values with the predicted ones in Eq. (3.3)). w/ ctr.-ness (L1): using L1 instead of BCE as the loss to optimize the center-ness.

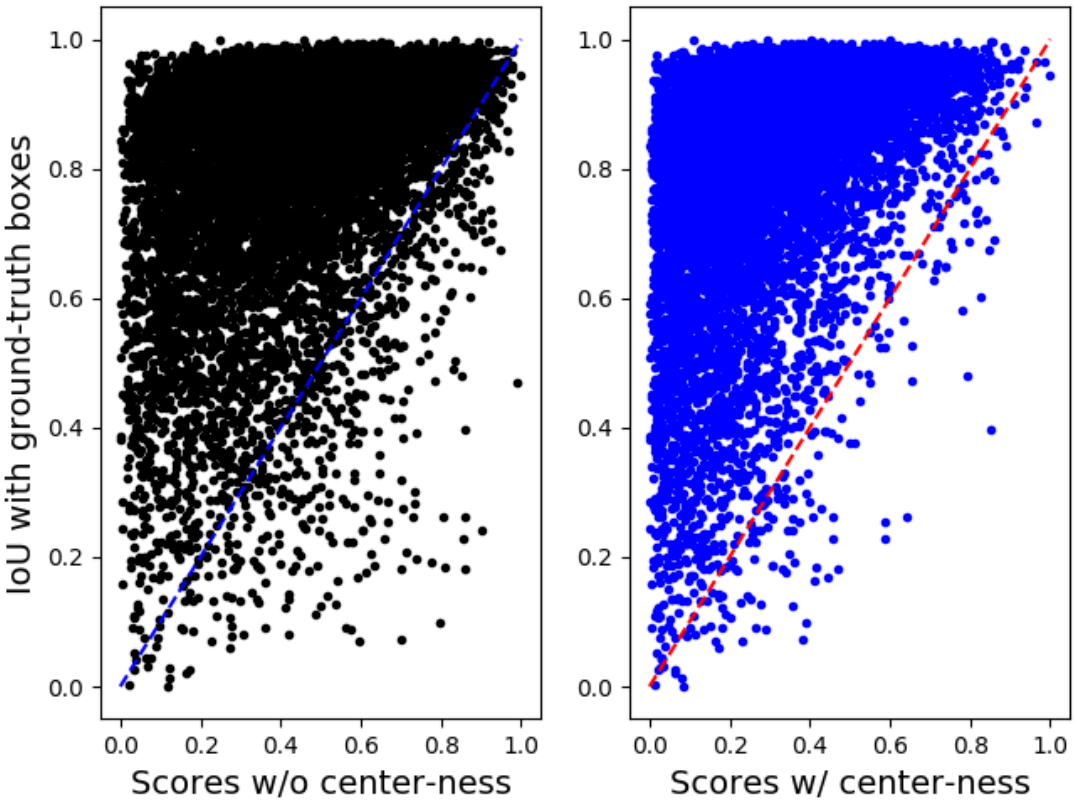

FIGURE 3.5. **Qualitative results of applying the center-ness scores to classification scores**. A point in the figure denotes a bounding box. The dashed line is the line $y = x$. As shown in the right figure, after applying the center-ness scores, the boxes with low IoU scores but high confidence scores (*i.e.*, under the line $y = x$) are reduced substantially.

use of the center sampling by default and it already eliminates a large number of false positives. However, the improvement is still impressive as the center-ness branch only adds negligible computational time. Moreover, we later show that the center-ness can considerably boost the performance for crowded scenarios. One may note that center-ness can also be computed with the predicted regression vector without introducing the extra center-ness branch. However, as shown in Table 3.4, the center-ness computed from the regression vector cannot improve the performance and thus here the separate center-ness is necessary.

We visualize the effect of applying the center-ness in Fig. 3.5. As shown in the figure, after applying the center-ness scores to the classification scores, the boxes with low IoU scores but high confidence scores are largely eliminated (*i.e.*, the points under the line $y = x$ in the Fig. 3.4), which are potential false positives.

Other Design Choices

Other design choices are also investigated. As shown Table 3.5, removing group normalization (GN) Wu and He, 2018 in both the classification and regression heads drops the performance by 1% AP. By replacing GIoU Rezatofighi et al., 2019 with the

	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
baseline	**38.9**	**57.5**	**42.2**	**23.1**	**42.7**	**50.2**
w/o GN	37.9	56.4	40.9	22.1	41.8	48.8
w/ IoU	38.6	57.2	41.9	22.4	42.1	49.8
w/ C_5	38.5	57.4	41.7	22.8	42.1	49.3

TABLE 3.5. Ablation study for design choices in FCOS. w/o GN: without using Group Normalization (GN) for the convolutional layers in heads. w/ IoU: using IoU loss in Yu et al., 2016 instead of GIoU. w/ C_5: using C_5 instead of P_5.

r	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
1.0	38.5	57.2	41.5	22.6	42.3	49.7
1.5	**38.9**	57.5	**42.2**	**23.1**	**42.7**	**50.2**
2.0	38.8	**57.7**	41.7	22.7	42.6	49.9

TABLE 3.6. Ablation study for the radius r of positive sample regions (defined in Section 3.2.1).

origin IoU loss in Yu et al., 2016, the performance drops by 0.3% AP. Using C_5 instead of P_5 also degrades the performance. Moreover, using P_5 can reduce the number of the network parameters. We also conduct experiments for the radius r of positive sample regions. As shown in Table 3.6, $r = 1.5$ has the best performance on COCO `val` split. We also attempted to change the sampling area from square sub-boxes to the rectangle sub-boxes with the same aspect ratio as the ground-truth boxes, which results in similar performance. It suggests that the shape of the sampling area may be not sensitive to the final performance.

We also conduct experiments with different strategies of assigning objects to FPN levels. First, we experiment with the assigning strategy that FPN Lin et al., 2017a assigns the object proposals (*i.e.*, ROIs) to FPN levels. It assigns the objects according to the formulation $k = \lfloor k_0 + \log_2(\sqrt{wh}/224) \rfloor$, where $k \in \{3, 4, 5, 6, 7\}$ is the target FPN level, w and h are the ground-truth box's width and height, respectively, and k_0 is the target level which an object with scale 224 should be mapped into. We use $k_0 = 5$. As shown in Table 3.7, this strategy results in degraded performance (37.7% AP). We conjecture that it may be because the strategy cannot make sure the complete

strategy	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
FPN	37.7	56.6	40.6	22.2	40.9	49.7
$\sqrt{(h^* \times w^*)}/2$	37.6	56.5	40.6	22.4	41.6	47.3
$\max(h^*, w^*)/2$	38.1	57.0	41.3	22.5	41.8	48.7
$\max(l^*, t^*, r^*, b^*)$	**38.9**	**57.5**	**42.2**	**23.1**	**42.7**	**50.2**

TABLE 3.7. Ablation study for different strategies of assigning objects to FPN levels. FPN: the strategy of assigning object proposals (*i.e.*, ROIs) to FPN levels in the original FPN, described in the text. h^* and w^* are the height and width of a ground-truth box, respectively. l^*, t^*, r^* and b^* are the distances from a location to the four boundaries of a ground-truth box. "$\max(l^*, t^*, r^*, b^*)$" (used by FCOS) has the best performance.

object is within the receptive field of the target FPN level. Similarly, $\sqrt{(h^* \times w^*)}/2$ and $\max(h^*, w^*)/2$ also deteriorate the performance. Eventually, $\max(l^*, t^*, r^*, b^*)$ achieves the best performance as the strategy makes sure that the complete target objects are always in the effective receptive field of the FPN level. Moreover, this implies that the range hyper-parameters of each FPN level (*i.e.*, m_i) is mainly related to the network architecture (which determines the receptive fields). This is desirable since it eliminates the hyper-parameter tuning when FCOS is applied to different datasets.

3.3.2 FCOS vs. Anchor-based Counterparts

Here, we compare FCOS with its anchor-based counterpart RetinaNet on the challenging benchmark COCO, demonstrating that the much simpler anchor-free FCOS is superior.

In order to make a fair comparison, we add the universal improvements in FCOS to RetinaNet. The improved RetinaNet is denoted as "*RetinaNet w/ imprv.*" in Table 3.3. As shown the table, even without the center-ness branch, FCOS achieves 0.4% better AP than "RetinaNet (#A=9) w/ imprv." (38.0% vs 37.6% in AP). The performance of FCOS can be further boosted to 38.9% with the help of the proposed center-ness branch. Moreover, it is worth noting that FCOS achieves much better performance than the RetinaNet with a single anchor per location "RetinaNet (#A=1) w/ imprv." (38.0% vs 35.2%), which suggests that FCOS is not equivalent to the single-anchor RetinaNet. The major difference is FCOS does not employ IoU scores between anchor boxes and ground-truth boxes to determine the training labels.

Given the superior performance and merits of the anchor-free detector (*e.g.*, much simpler and fewer hyper-parameters), we encourage the community to rethink the necessity of anchor boxes in object detection.

3.3.3 Comparison with State-of-the-art Detectors on COCO

We compare FCOS with other state-of-the-art object detectors on `test-dev` split of MS-COCO benchmark. For these experiments, following previous works Lin et al., 2017b; Liu et al., 2016b, we make use of multi-scale training. To be specific, during training, the shorter side of the input image is sampled from $[640, 800]$ with a step of 32. Moreover, we double the number of iterations to 180K (with the learning rate change points scaled proportionally). Other settings are exactly the same as the model with AP 38.9% on `val2017` in Table 3.3.

As shown in Table 3.8, with ResNet-101-FPN, FCOS outperforms the original RetinaNet with the same backbone by 4.1% AP (43.2% vs. 39.1%). Compared to other one-stage detectors such as SSD Liu et al., 2016b and DSSD Fu et al., 2017, we also achieve much better performance. Moreover, FCOS also surpasses the classical two-stage anchor-based detector Faster R-CNN by a large margin (43.2% vs. 36.2%). To

method	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods:							
Faster R-CNN+++	ResNet-101	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w/ FPN	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI	Inception-ResNet-v2	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w/ TDM	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	52.1
One-stage methods:							
YOLOv2	DarkNet-19	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 Liu et al., 2016b	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
YOLOv3 608×608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9
DSSD513 Fu et al., 2017	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet Lin et al., 2017b	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
CornerNet	Hourglass-104	40.5	56.5	43.1	19.4	42.7	53.9
FSAF	ResNeXt-64x4d-101-FPN	42.9	63.8	46.3	26.6	46.2	52.7
CenterNet511	Hourglass-104	44.9	62.4	48.1	25.6	47.4	57.4
HoughNet	Hourglass-104	46.4	65.1	50.7	29.1	48.5	58.1
CPN Duan et al., 2020	Hourglass-104	49.2	67.4	53.7	31.0	51.9	62.4
FCOS	ResNet-101-FPN	43.2	62.4	46.8	26.1	46.2	52.8
FCOS	ResNeXt-32x8d-101-FPN	44.1	63.7	47.9	27.4	46.8	53.7
FCOS	ResNeXt-64x4d-101-FPN	44.8	64.4	48.5	27.7	47.4	55.0
FCOS w/ deform. conv. v2	ResNeXt-32x8d-101-FPN	46.6	65.9	50.8	28.6	49.1	58.6
FCOS	ResNet-101-BiFPN	45.0	63.6	48.7	27.0	47.9	55.9
FCOS	ResNeXt-32x8d-101-BiFPN	46.2	65.2	50.0	28.7	49.1	56.5
FCOS w/ deform. conv. v2	ResNeXt-32x8d-101-BiFPN	47.9	66.9	51.9	30.2	50.3	59.9
w/ test-time augmentation:							
FCOS	ResNet-101-FPN	45.9	64.5	50.4	29.4	48.3	56.1
FCOS	ResNeXt-32x8d-101-FPN	47.0	66.0	51.6	30.7	49.4	57.1
FCOS	ResNeXt-64x4d-101-FPN	47.5	66.4	51.9	31.4	49.7	58.2
FCOS w/ deform. conv. v2	ResNeXt-32x8d-101-FPN	49.1	68.0	53.9	31.7	51.6	61.0
FCOS	ResNet-101-BiFPN	47.9	65.9	52.5	31.0	50.7	59.7
FCOS	ResNeXt-32x8d-101-BiFPN	49.0	67.4	53.6	32.0	51.7	60.5
FCOS w/ deform. conv. v2	ResNeXt-32x8d-101-BiFPN	**50.4**	**68.9**	**55.0**	**33.2**	**53.0**	**62.7**

TABLE 3.8. FCOS vs. other state-of-the-art two-stage or one-stage detectors (*single-model results*). FCOS outperforms a few recent anchor-based and anchor-free detectors by a considerable margin.

our knowledge, *this is the first time that an anchor-free detector, without any bells and whistles, outperforms anchor-based detectors by a large margin.* Moreover, FCOS also outperforms the previous anchor-free detector CornerNet Law and Deng, 2018 and CenterNet Duan et al., 2019 while being much simpler since they requires to group corners with embedding vectors, which needs special design for the detector. Thus, we argue that FCOS is more likely to serve as a strong and simple alternative to current mainstream anchor-based detectors. Some qualitative results are shown in Fig. 3.6. It appears that FCOS works well with a variety of challenging cases.

We also introduce some complementary techniques to FCOS. First, deformable convolutions are used in stages 3 and 4 of the backbone, and also replace the last convolutional layers in the classification and regression towers (*i.e.*, the $4\times$ convolutions shown in Fig. 3.2). As shown in Table 3.8, by applying deformable convolutions Dai et al., 2017; Zhu et al., 2019 to ResNeXt-32x8d-101-FPN based FCOS, the performance is improved from 44.1% to 46.6% AP, as shown in Table 3.8. In addition, we

FIGURE 3.6. Qualitative results. FCOS works well with a wide range of objects including crowded, occluded, extremely small and very large objects. Best viewed on screen.

method	FPS	AP	AP `test-dev`
YOLOv3 (Darknet-53) Redmon and Farhadi, 2018	26	–	33.0
CenterNet (DLA-34) Zhou, Wang, and Krähenbühl, 2019	**52**	37.4	37.3
FCOS-RT (R-50)	38	40.2	40.2
FCOS-RT (DLA-34-BiFPN)	43	**42.1**	**42.2**
FCOS-RT (DLA-34)	46	40.3	40.3
FCOS-RT w/ shtw. (DLA-34)	**52**	39.1	39.2

TABLE 3.9. **Real-time FCOS (FCOS-RT) models.** AP (%) is on COCO `val` split. "shtw.": sharing towers (*i.e.*, 4× conv. layers shown in Fig. 3.2) between the classification and regression branches. The inference time is measured with a single 1080Ti or Titan XP GPU (these two GPUs' speeds are close).

also attempt to replace FPN in FCOS with BiFPN Tan, Pang, and Le, 2020. We make use of BiFPN in the D3 model in Tan, Pang, and Le, 2020. To be specific, the single cell of BiFPN is repeated 6 times and the number of its output channels is set to 160. Note that unlike the original BiFPN, we do not employ depth-wise separable convolutions here. As a result, BiFPN generally improves all FCOS models by $\sim 2\%$ AP and pushes the performance of the best model to 47.9%.

We also report the results of using test-time data augmentation. Specifically, in inference, the input image is respectively resized to $[400, 1200]$ pixels with step 100. At each scale, the original image and its horizontal flip are evaluated. The results from these augmented images are merged by NMS. As shown in Table 3.8, the test-time augmentation improves the best performance to 50.4% AP.

3.3.4 Real-time FCOS

We also design a real-time version FCOS-RT. In the real-time settings, we decrease the shorter side of input images from 800 to 512 and the maximum longer size from 1333 to 736, which decreases the inference time per image by $\sim$ 50%. With the smaller input size, the higher feature levels P_6 and P_7 become less important. Thus, following BlendMask-RT Chen et al., 2020, we remove P_6 and P_7, further reducing the inference time. Moreover, in order to boost the performance of the real-time version, we employ a more aggressive training strategy. Specifically, during training, multi-scale data augmentation is used and the shorter size of input image is sampled from

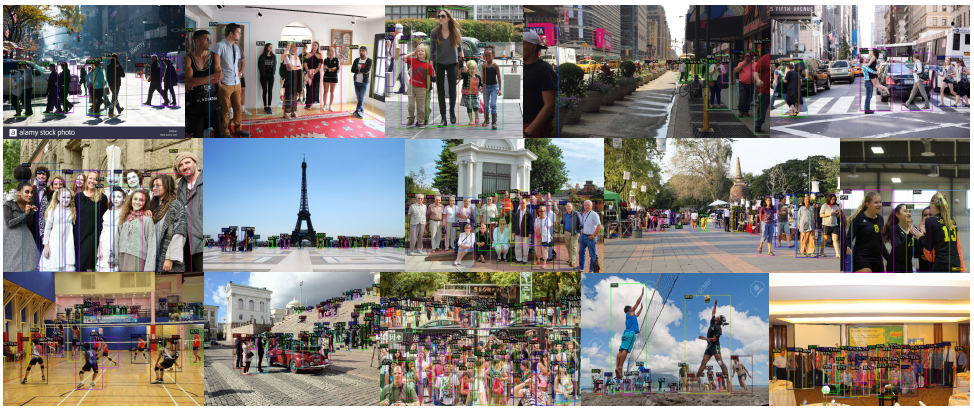

FIGURE 3.7. Qualitative results on the CrowdHuman `val` set with the ResNet-50-FPN backbone. Best viewed on screen.

256 to 608 with interval 32. Synchronized batch normalization (SyncBN) is used. We also increase the training iterations to $360K$ (*i.e.*, 4×). The learning rate is decreased by a factor of 10 at iteration $300K$ and $340K$.

The resulting real-time models are shown in Table 3.9. With ResNet-50, FCOS-RT can achieve 40.2% AP at 38 FPS on a single 1080Ti GPU card. We further replace ResNet-50 with the backbone DLA-34 Yu et al., 2018, which results in a better speed/accuracy trade-off (40.3% AP at 46 FPS). In order to compare with CenterNetZhou, Wang, and Krähenbühl, 2019, we share the towers (*i.e.*, 4× conv. layers shown in Fig. 3.2) between the classification and regression branches, which improves the speed from 46 FPS to 52 FPS but deteriorate the performance by 1.2% AP. However, as shown in Table 3.9, the model still outperforms CenterNet Zhou, Wang, and Krähenbühl, 2019 by 1.7% AP at the same speed. For the real-time models, we also replace FPN with BiFPN as in Section 3.3.3, resulting 1.8% AP improvement (from 40.3% to 42.1%) at a similar speed. A speed/accuracy comparison between FCOS and a few recent detection methods is shown in Fig. 3.3.

3.3.5 FCOS on CrowdHuman

	AP	MR^{-2}	JI
RetinaNet w/ imprv.	81.60	57.36	72.88
FCOS w/o ctr.-ness	83.16	59.04	73.09
FCOS w/ ctr.-ness	85.0	51.34	74.97
+ MIP ($K = 2$)	85.19	51.60	75.14
+ Set NMS Chu et al., 2020	**87.28**	**51.21**	**77.34**

TABLE 3.10. **FCOS for crowded object detection on the CrowdHuman dataset**. Even on the highly crowded benchmark, FCOS still attains even better performance than anchor-based RetinaNet. Note that lower MR^{-2} is better. "MIP w. set NMS": Multiple Instance Prediction, which predicts multiple instances from a single location as proposed by Chu et al., 2020. Note that we are not pursuing the state-of-the-art performance on the benchmark. We only show that the anchor boxes are not necessary even on the highly-crowded benchmark.

We also conduct experiments on the highly crowded dataset CrowdHuman Shao et al., 2018. CrowdHuman consists of $15K$ images for training, $4,370$ for validation and $5,000$ images for testing. Following previous works on crowded benchmark Chu et al., 2020; Shao et al., 2018, we use AP, *long-average Miss Rate on False Positive Per Image in* $[10^{-2}, 100]$ (MR^{-2}) Dollar et al., 2011 and *Jaccard Index* (JI) as the evaluation metrics. Note that lower MR^{-2} is better. Following Chu et al., 2020, all experiments here are trained on the `train` split for 30 epochs with batch size 16 and then evaluated on the `val` split. Two small changes are made when FCOS is applied to the benchmark. First, the NMS threshold is set as 0.5 instead of 0.6. We find that it has large impact on MR^{-2} and JI. Second, when a location is supposed to be associated to multiple ground-truth boxes, on COCO, we choose the object with minimal area as the target for the location. On CrowdHuman, we instead choose the target with minimal distance to the location. The distance between a location and an object is defined as the distance from the location to the center of the object. On COCO, both schemes result in similar performance. However, the latter has much better performance than the former on the highly crowded dataset. Other settings are the same as that of COCO.

First, we count the ambiguous sample ratios on CrowdHuman `val` set. With FPN-based FCOS, there are 84.47% unambiguous positive samples (with one ground-truth box), 13.63% with two ground-truth boxes, 1.69% with three ground-truth boxes and the rest ($< 0.3\%$) with more than three ground-truth boxes. Given the much higher ambiguous sample ratio than COCO, it is expected that FCOS will have inferior performance on the highly crowded dataset.

We compare FCOS without center-ness with the the improved RetinaNet (*i.e.*, "RetinaNet w/ imprv."). To our surprise, even without center-ness, FCOS can already achieve decent performance. As shown in Table 3.10, FCOS compares favorably with its anchor-based counterpart RetinaNet on two out of three metrics (AP and JI), *which suggests that anchor-based detectors have no large advantages even under the highly crowded scenario.* The higher MR^{-2} of FCOS denotes that FCOS might have a large number of false positives with high confidence. By using the center-ness, MR^{-2} can be significantly reduced from 59.04% to 51.34%. As a result, FCOS can achieve better results under all the three metrics.

Furthermore, as shown in Chu et al., 2020, it is more reasonable to let one proposal make multiple predictions under the highly crowded scenario (*i.e.*, multiple instance prediction (MIP)). After that, these predictions are merged by Set NMS Chu et al., 2020, which skips the suppression for the boxes from the same location. A similar idea can be easily incorporated into FCOS. To be specific, if a location should be associated to multiple objects, instead of choosing a single target (*i.e.*, the closest one to the location), the location's targets are set as the K-closest objects. Accordingly, the network is required to make K predictions per location. Moreover, we do not make use of the earth mover's distance (EMD) loss for simplicity. Finally, the results

are merged by Set NMS Chu et al., 2020. As shown in Table 3.10, with MIP and Set NMS, improved performance is achieved under all the three metrics. Some qualitative results are shown in Fig. 3.7.

3.4 Conclusions

In this work, we have proposed an anchor-free and proposal-free one-stage detector FCOS. Our experiments demonstrate that FCOS compares favourably against the widely-used anchor-based one-stage detectors, including RetinaNet, YOLO and SSD, but with much less design complexity. FCOS completely avoids all computation and hyper-parameters related to anchor boxes and solves the object detection in a per-pixel prediction fashion, similar to other dense prediction tasks such as semantic segmentation. Given its effectiveness and efficiency, we hope that FCOS can serve as a strong and simple alternative of current mainstream anchor-based detectors.

Chapter 4

DirectPose: Direct End-to-End Multi-Person Pose Estimation

4.1 Introduction

Multi-person pose estimation (a.k.a. keypoint detection) is a crucial step in the understanding of human behavior in images and videos. Previous methods for the task can be roughly categorized into bottom-up Cao et al., 2017; Newell, Huang, and Deng, 2017; Papandreou et al., 2018; Pishchulin et al., 2016 and top-down He et al., 2017; Sun et al., 2019; Fang et al., 2017; Chen et al., 2018 methods. Bottom-up methods first detect all the possible keypoints in an input image in an instance-agnostic fashion, which are followed by a grouping or assembling process to produce the final instance-aware keypoints. The grouping process is often heuristic and many tricks are involved to achieve a good performance. In contrast, top-down methods first detect each individual instance with a box and then reduce the task to single-instance keypoint detection. Although top-down methods can avoid the heuristic grouping process, they come with the price of long computational time since they cannot fully leverage the sharing computation mechanism of convolutional neural networks (CNNs). Moreover, the running time of top-down methods depends on the number of instances in the image, making them unreliable in some instant applications such as autonomous vehicles. Importantly, *both bottom-up and top-down methods are not end-to-end*[1], which is in conflict with deep learning's philosophy of learning everything together.

As shown before, the anchor-free detector FCOS has demonstrated superior performance than previous anchor-based ones. It directly regresses two corners of a target box, without using pre-defined anchor boxes. The straightforward and effective solution for object detection gives rise to a question: *can keypoint detection be solved with this simple framework as well?* It is easy to see that the keypoints for an instance can be considered as a special box with more than two corner points, and thus the task could be solved by attaching more output heads to the object detection networks.

[1]Here we mean 'direct end-to-end'; *i.e.*, the model is trained end-to-end with keypoint annotations solely during training, and for inference, the model is able to map an input to keypoints for each individual instance without box detection and grouping post-processing.

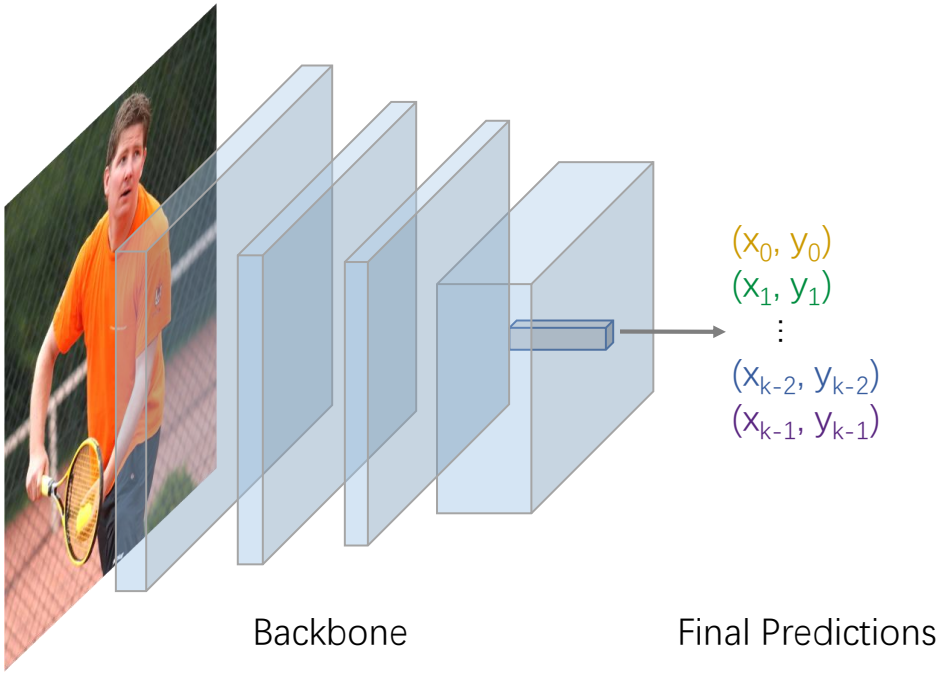

FIGURE 4.1. **The naive direct end-to-end keypoint detection framework.** As shown in the figure, the framework requires a single feature vector on the final feature maps to encode all the essential information of an instance (*e.g.*, the precise locations of some keypoints for the instance, denoted as $(x_0, y_0), ...(x_{k-1}, y_{k-1})$).

This solution is intriguing since 1) it is end-to-end trainable (*i.e.*, directly mapping a raw input image to the desired instance-aware keypoints). 2) It can avoid the shortcomings of both top-down and bottom-up methods as it needs neither grouping or box detection. 3) It can unify object detection and keypoint detection in a single simple and elegant framework.

However, we show that such a naive approach performs unsatisfactorily, mainly due to the fact that these object detectors resort to a single feature vector to regress all the keypoints of interest for an instance, with the hope that the single feature vector can faithfully preserve the essential information (*e.g.*, the precise locations of all the keypoints) in its receptive field, as shown in Fig. 4.1. While the single feature vector may be sufficiently good to carry information for simple box detection as shown in Tian et al., 2019b, where only two corner points are involved in a box, it has difficulties in encoding rich information for the more challenging keypoint detection. As shown in our experiments, this straightforward approach yields inferior performance.

In this work, we propose a keypoint alignment (KPAlign) mechanism to largely overcome the aforementioned problem of the solution. Instead of using a single feature vector to regress all the keypoints for an instance, the proposed KPAlign aligns the convolutional features with a target keypoint (or a group of keypoints) as possible as it can, and then predicts the location of the target keypoint(s) with the aligned features. Since the target keypoints and the used features are roughly aligned, the

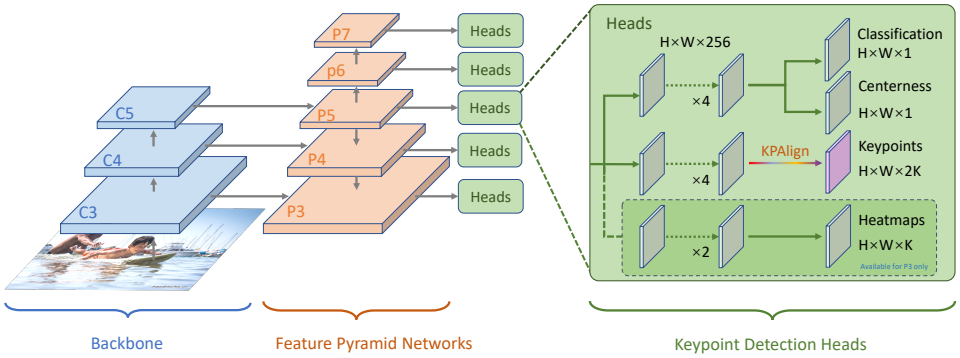

FIGURE 4.2. **The proposed direct end-to-end multi-person pose estimation framework.** The framework shares a similar architecture with one-stage object detectors such as FCOS Tian et al., 2019b but the box branch is replaced with a keypoint branch. KPAlign: the proposed keypoint alignment module, as described in Sec. 4.2.2. Heatmaps: the branch for jointly heatmap-based learning and will be removed when testing. Keypoints: the branch for keypoint detection. Classification is from FCOS and used to classify the locations on the feature maps into "person" or "not person". Center-ness is also from FCOS.

features are only required to encode the information in its neighborhood. It is evident that encoding the neighborhood is much easier than encoding the whole receptive field, which thus results in an improved performance. Moreover, the KPAlign module is differentiable, thus keeping the model end-to-end trainable. Additionally, it is well-known that learning a regression-based model is difficult. However, in this work, we find the regression task can largely benefit from a heatmap-based learning. As a result, we propose to jointly learn the two tasks during training. When testing, the heatmap-based branch is disabled and thus does not impose any overheads to the framework.

To summarize, the proposed one-stage regression-based keypoint detection enjoys the followings advantages over previous top-down or bottom-up approaches.

- The proposed framework is direct, totally end-to-end trainable. To predict, it maps an input image to keypoints for each individual instance *directly*, relying on neither intermediate operators like RoI feature cropping, nor grouping post-processing, which sets our work apart from previous frameworks He et al., 2017; Cao et al., 2017 with multiple steps.
- Our proposed framework can bypass the major shortcomings of both top-down and bottom-up methods. For example, compared to top-down methods, our framework can avoid the issue of early commitment and decouple computational complexity from the number of instances in an input image. Compared to bottom-up methods, our framework eliminates the heuristic post-processing assembling the detected keypoints into full-body instances.
- Finally, the framework suggests that the keypoint detection task can also be

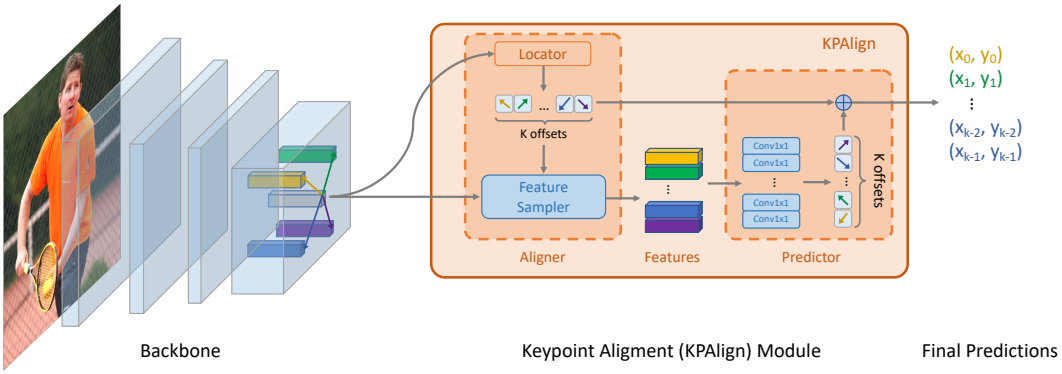

FIGURE 4.3. **The proposed keypoint detection framework with the Keypoint Alignment (KPAlign) module.** Feature pyramid networks (FPNs) are not shown here. The aligner consists a locator and a sampler. The locator is essentially a 3×3 convolution layer and predicts the rough locations of the keypoints. Next, the feature sampler samples feature vectors at these locations. Thus, the aligner can roughly align the features and the predicted keypoints. The predictor employs these aligned feature vectors to make the final keypoint predictions.

solved with the same methodology as box detection (*i.e.*, directly regressing all the keypoints or the corners of boxes), resulting in a unifying framework for both tasks.

4.2 Our Approach

In this section, we first show how FCOS can be extended to keypoint detection. Next, we illustrate our proposed KPAlign module, which allows the framework to leverage the feature-prediction alignment and improves the performance by a large margin. Finally, we present that how the jointly learning of the regression-based task and a heatmap-based task can be used to further boost the precision of keypoint localization.

4.2.1 End-to-End Multi-Person Pose Estimation

Extending FCOS to Keypoint Detection. It is straightforward to extend the FCOS to keypoint detection. Specifically, we increase the scalars that each pixel regresses from 4 to $2K$, where K is the number of keypoints for each instance. Similarly, the $2K$ scalars denote the relative coordinates to the current pixel. In other words, we regard keypoints as a special box with K corner points.

Our End-to-End Framework. As shown in Fig. 4.2, the framework is implemented by applying a convolutional branch on all levels of the output feature maps of FPN Lin et al., 2017a (*i.e.*, P_3, P_4, P_5, P_6 and P_7). The downsampling ratios of these feature maps to the input image are 8, 16, 32, 64 and 128, respectively. Note that the parameters of the branch are shared between FPN levels as in the box detection branch of the FCOS detector. The output channels of the branch is $2K$, where K is

the number of keypoints for each instance. The original box branch can be kept for simultaneous keypoint and box detection. Moreover, for keypoint-only detection, it is worth noting that we only use keypoint annotations without boxes. However, during training, FCOS requires a box for each instance to determine a positive or negative label for each location on the FPN feature maps. Here, we employ the minimum enclosing rectangles of keypoints of the instances as pseudo-boxes for computing the training labels.

4.2.2 Keypoint Alignment (KPAlign) Module

We conduct preliminary experiments with the aforementioned naive keypoint detection framework. However, as shown in our experiments, it has inferior performance. We attribute the inferior performance to the lack of the alignment between the features and the predicted keypoints. Essentially, the naive framework makes use of a single feature vector at a location on the input feature maps to regress all the keypoints for an instance. As a result, the single feature vector is required to encode all the required information for the instance. This is difficult because many keypoints are far away from the center of the feature vector's receptive field and it has been shown in Luo et al., 2016 that the intensity of the feature's response decays quickly as the input signal deviates from the center of its receptive field. As shown in many FCN-based frameworks He et al., 2017; Long, Shelhamer, and Darrell, 2015, keeping the feature and prediction aligned is crucial to good performance. Thus the feature only needs to encode the information in a local patch, which is much easier.

In this work, we propose a keypoint alignment (KPAlign) module to recover the feature-prediction alignment in the framework. KPAlign is used to replace the convolutional layer for the final keypoint detection in the naive framework and take as input the same feature maps, denoted as $\mathbf{F} \in \mathbb{R}^{H \times W \times C}$, where C being 256 is the number of channels of the feature maps. Analogous to a convolution operation, KPAlign is densely slid through the input feature maps $\mathbf{F}$. For simplicity, we take as an example a specific location (i, j) on $\mathbf{F}$ to illustrate how KPAlign works. As shown in Fig. 4.3, KPAlign consists of two components — an aligner ζ and a predictor ϕ. The aligner consists of a locator and a feature sampler, and outputs the aligned feature vectors. The aligner can be formulated as,

$$\boldsymbol{o}_0, \boldsymbol{o}_1, ..., \boldsymbol{o}_{K-1}, \boldsymbol{v}_0, \boldsymbol{v}_1, ..., \boldsymbol{v}_{K-1} = \zeta(\mathbf{F}), \tag{4.1}$$

where $\boldsymbol{o}_t \in \mathbb{R}^2$, produced by the locator in Fig. 4.3, is the location where the feature vector used to predict the t-th keypoint of an instance should be sampled. $\boldsymbol{v}_t \in \mathbb{R}^C$ is the sampled feature vector. Note that the location $\boldsymbol{o}_t$ is defined over $\mathbb{R}^2$ and thus it can be fractional. Following Dai et al., 2017; He et al., 2017, we make use of bilinear interpolation to compute the features at a fractional location. Additionally, the location is encoded as the coordinates relative to (i, j) and thus is translation invariant.

Next, the predictor ϕ takes the outputs of the aligner as inputs to predict the final coordinates of the keypoints. As shown in Fig. 4.3, the predictor includes K convolution layers (*i.e.*, one for each keypoint). Let us assume that we are looking for the t-th keypoint for the instance and let ϕ_t denote the t-th convlutional layer in the predictor. ϕ_t takes $\boldsymbol{v}_t$ as input and predicts the coordinates of the t-th keypoint relative to the location where $\boldsymbol{v}_t$ is sampled (*i.e.*, $\boldsymbol{o}_t$). Finally, the coordinates of the t-th keypoint, denoted as $\boldsymbol{x}_t$, are the sum of the two sets of coordinates. Formally,

$$\boldsymbol{x}_t = \phi_t(\boldsymbol{v}_t) + \boldsymbol{o}_t, \qquad t = 0, 1, ..., K-1. \tag{4.2}$$

Note that the coordinates need to be re-scaled by the down-sampling ratio of $\mathbf{F}$. We omit the re-scaling operator here for simplicity. Note that all the operations in KPAlign module are differentiable and therefore *the whole model can be trained in an end-to-end fashion with standard back-propagation*, which sets our work apart from previous bottom-up or top-down keypoint detection frameworks such as CMU-Pose Cao et al., 2017 or Mask R-CNN He et al., 2017. *Being end-to-end trainable also makes the locator be able to learn to localize the keypoints without explicit supervision, which is critically important to KPAlign.*

Grouped KPAlign. The aforementioned KPAlign module is required to sample K feature vectors for K keypoints. This is actually not necessary because some keypoints (*e.g.*, nose, eyes and ears) always populate in a local area. Therefore, we propose to group the keypoints and the keypoints in the same group will use the same feature vector, which reduces the number of sampled feature vectors from K to G and achieves a similar performance, where G is the number of groups.

Using Separate Convolutional Features. In the KPAlign described before, all of the keypoint groups use the feature maps $\mathbf{F}$ as the input. However, we find that the performance can be improved *remarkably* if we use separate feature maps for the G keypoint groups (*i.e.*, using $\mathbf{F}_t, t = 0, 1, ..., G-1$). In that way, the demand for the information encoded in a single $\mathbf{F}_t$ can be further mitigated. In order to reduce the computational complexity, the number of channels of each $\mathbf{F}_t$ is set as $\frac{C}{4}$ (*i.e.*, from 256 to 64).

Where to Sample Features? For the sake of convenience, the sampler in the aforementioned aligner samples features on the input feature maps of the locator, and therefore the predictor and locator take as inputs the same feature maps. However, it is not reasonable as the locator and predictor require different levels of feature maps. The locator predicts the initial but imprecise locations for all the keypoints (or keypoint groups) of an instance and thus requires high-level features with a larger receptive field. In contrast, the predictor needs to make precise predictions but only for the keypoints in a local area because the features have been aligned by the aligner. As a result, the predictor prefers high-resolution low-level features with a smaller receptive field. To this end, we feed lower levels of feature maps into the sampler.

Specifically, if a locator uses feature maps P_L and P_L is not the finest feature maps, the sampler will take P_{L-1} as the input. If P_L is already the finest feature maps, the sampler will still sample on it.

4.2.3 Regularization from Heatmap Learning

It is well-known that regression-based tasks are difficult to learn Glorot and Bengio, 2010; Sun et al., 2018 and have poor generalization. As a result, we need to seek a way that can make the regression-based task easier to learn and generalize. To this end, given the fact that heatmap-based learning is much easier, we use the heatmap-based prediction task as an auxiliary task. Thus, the heatmap-based task can serve as a hint for the regression-based task and thus can regularize the task. In our experiments, the jointly learning significantly boosts the performance of the regression-based task. Note that *the heatmap-based task is only used as an auxiliary loss during training.* It is removed when testing.

Heatmap Prediction. As shown in Fig. 4.2, the heatmap prediction task takes as input the FPN feature maps P_3 with downsampling ratio being 8. Afterwards, two 3×3 conv layers with channel being 128 are applied here, which are followed by another 3×3 conv layer with output channel being K for the final heatmap prediction, where K is the number of keypoints for each instance.

Ground-truth Heatmaps and Loss Function. The ground-truth heatmaps are generated as follows. On the heatmaps, if a location is the nearest location to a keypoint with type t, the classification label for the location is set as t, where $t \in \{1, 2, ..., K\}$. Otherwise, the label is 0.[2] Finally, in order to overcome the imbalance between positive and negative samples, we use focal loss Lin et al., 2017b as the loss function.

4.3 Experiments

Our experiments are conducted on human keypoint detection task of the large-scale benchmark COCO dataset Lin et al., 2014. The dataset contains more than $250K$ person instances with 17 annotated keypoints. Following the common practice Cao et al., 2017; He et al., 2017, we use the COCO `trainval35k` split ($57K$ images) for training and `minival` split ($5K$ images) as validation for our ablation study. We report our main results on the `test-dev` split ($20K$ images). Unless specified, we only make use of the human keypoint annotations without boxes. The performance is computed with Average Percision (AP) based on Object Keypoint Similarity (OKS).

Implementation Details. Unless specified, ResNet-50 He et al., 2016 is used as our backbone networks. We use two training schedules. The first is quick and used to train a fast prototype of our models in ablation experiments. Specifically, the

[2] Strictly speaking, the generated ground-truth is a set of binary labels, rather than the conventional real-valued heatmap. We slightly abuse the term here.

	AP^{kp}	AP^{kp}_{50}	AP^{kp}_{75}	AP^{kp}_{M}	AP^{kp}_{L}
Baseline	43.4	73.8	45.1	38.9	50.9
w/ KPAlign†	43.0	74.2	43.9	39.0	49.6
w/ KPAlign	**50.5**	**77.6**	**54.9**	**44.4**	**60.0**

TABLE 4.1. **Ablation experiments on COCO `minival` for the proposed KPAlign module.** Baseline: the naive keypoint detection framework, as shown in Fig. 4.1. "w/ KPAlign†": using the KPAlign module in the naive framework but disabling the aligner in it. "w/ KPAlign": using the full-featured KPAlign module.

models are trained with stochastic gradient descent (SGD) on 8 V100 GPUs for 25 epochs with a mini-batch of 16 images. For the main results on `test-dev` split, we use a longer training schedule; the models are trained for 100 epochs with a mini-batch of 32 images. We set the initial learning rate to 0.01 and use a linear schedule $base\_lr \times (1 - \frac{iter}{max\_iter})$ to decay it. Weight decay and momentum are set as 0.0001 and 0.9, respectively. We initialize our backbone networks with the weights pre-trained on ImageNet Deng et al., 2009. For the newly added layers, we initialize them as in Lin et al., 2017b. When training, the images are randomly resized and horizontally flipped with probability being 0.5, and the images are also randomly cropped into 800×800 patches. When testing, we run inference on the whole image and the testing images are resized to have its shorter side being 800 and their longer side less or equal to 1333. If box detection is available, NMS is applied to the detected boxes. Otherwise, we do NMS on the minimum enclosing rectangles of keypoints of the instances. The NMS threshold is set as 0.5 for all experiments.

4.3.1 Ablation Experiments

Baseline: the naive end-to-end framework

We first experiment with the naive end-to-end keypoint detection framework in Fig. 4.1 by replacing the box head in FCOS with the keypoint detection head. Moreover, as described before, we use pseudo-boxes to compute the label for each location on FPN feature maps during training. As shown in Table 4.1, the naive framework can only obtain low performance (43.4% in AP^{kp}). As mentioned before, the low performance is due to the misalignment between the features and keypoint predictions. In the following experiments, we will show that our proposed KPAlign can overcome the issue.

Keypoint alignment (KPAlign) module

In this section, we equip the above naive framework with our proposed KPAlign module. As shown in Fig. 4.2, KPAlign serves as the final prediction layer, which was a standard convolutional layer in the naive framework. As shown in Table 4.1, KPAlign improves the keypoint detection performance by a large margin (more than 7 points in AP^{kp}). In order to demonstrate that the improvement is indeed due to the

	AP^{kp}	AP^{kp}_{50}	AP^{kp}_{75}	AP^{kp}_{M}	AP^{kp}_{L}
KPAlign	50.5	77.6	54.9	44.4	60.0
+ Grouped	50.6	77.5	55.4	44.3	60.2
+ Sep. features	51.4	78.2	55.6	45.6	60.6
+ Better sampling	**52.2**	**78.3**	**56.6**	**46.3**	**61.7**

TABLE 4.2. **Ablation experiments on COCO `minival` for the design choices in KPAlign.** "+ Grouped": using Grouped KPAlign. "+ Sep. features": using separate (but slimmer) feature maps for different keypoint groups. "+ Better sampling": the predictor samples features on finer feature maps (*i.e.*, from P_L to P_{L-1}).

retained alignment between the features and keypoint predictions rather than other factors (*e.g.*, slightly more network parameters), we conduct another experiment in which the aligner of KPAlign is disabled. In other words, the offsets predicted by the locator are ignored and thus all the keypoints of an instance are predicted with the same features as in the naive framework. As shown in Table 4.1, without the aligner, the performance drops dramatically to 43.0% in AP^{kp}, which is nearly the same as the performance of the naive framework. Therefore, it is safe to claim that the improvement is due to the retained alignment.

Grouped KPAlign

As described before, it is not necessary to sample K (*i.e.*, 17 on COCO) feature vectors (one feature vector per keypoint) as some keypoints are always together and thus can be predicted with the same feature vector. In this experiments, we divide the keypoints into 9 groups[3], which reduces the number of the sampled feature vectors from 17 to 9 and makes the module faster. As shown in Table 4.2, the Grouped KPAlign can achieve slightly better performance than the original KPAlign. Therefore, in the sequel, we will use the Grouped KPAlign for all the following experiments. We also attempted other ways forming the groups but they achieve a similar performance.

Using separate convolutional features

As shown in Table 4.2, using separate feature maps can boost the performance by 0.8% in AP^{kp} (from 50.6% to 51.4%). Note that the number of channels of these separate feature maps is reduced from 256 to 64, and thus the model has similar computational complexity to the original one.

Where to sample features in KPAlign?

In this experiment, the sampler samples on finer feature maps, as described in Sec. 4.2.2. As shown in Table 4.2 ("+ Better Sampling"), using the sampling strategy can improve the performance to 52.2%. Note that using the sampling strategy does not

[3]These groups respectively include (nose, left eye, right eye, left ear, right ear), (left shoulder,), (left elbow, left wrist), (right shoulder,), (right elbow, right wrist), (left hip,), (left knee, left ankle), (right hip,) and (right knee, right ankle).

	AP^{kp}	AP^{kp}_{50}	AP^{kp}_{75}	AP^{kp}_{M}	AP^{kp}_{L}
Baseline	52.2	78.3	56.6	46.3	61.7
w/ 16× Heatmap	57.7	82.8	63.1	51.8	66.9
w/ 8× Heatmap	58.0	82.5	63.3	52.7	66.6
+ Longer sched.	**63.1**	**85.6**	**68.8**	**57.7**	**71.3**

TABLE 4.3. **Ablation experiments on COCO `minival` for DirectPose with heatmap prediction.** Baseline: without the heatmap learning. "16× Heatmaps": predicting the heatmaps with downsampling ratio being 16. "8× Heatmaps": predicting the heatmaps with downsampling ratio being 8 (*i.e.*, using P_3). "+ Long sched.": increasing the number of training epochs from 25 to 100.

increase the computational complexity of the model. Moreover, the better sampling strategy improves the AP^{kp}_{50} and AP^{kp}_{75} by 0.1% and 1.0%, respectively, which implies that the sampling strategy can result in more accurate keypoint predictions because the improvement mainly comes from the AP^{kp}s at higher thresholds.

Regularization from heatmap learning

As shown in Table 4.3 ("w/ 8× Heatmaps"), by jointly learning the regression-based model with a heatmap prediction task, the performance of the regression-based task can be largely improved from 52.2% to 58.0%. Note that the heatmap prediction is only used during training to provide the multi-task regularization. Moreover, we also conduct experiments with the heatmap prediction with a lower resolution (*i.e.*, "w/ 16× Heatmaps"). As shown in Table 4.3, even with the low-resolution heatmaps, the model can still yield a similar performance. This suggests that our method is not sensitive to the design choices for the heatmap learning. In order to demonstrate the impact of the heatmap learning, we plot the loss curves of training with or without the heatmap learning in Fig. 4.4. As shown in the figure, the heatmap learning can greatly help the training of the model and make the model achieve a much lower loss value, thus resulting in much better performance.

Moreover, we find that our method is highly under-fitting and previous methods such as Sun et al., 2019 with heatmaps learning are trained with much more epochs than ours, and therefore we increase the number of epochs from 25 to 100. As shown in Table 4.3, this improves the performance by 5.1% in AP^{kp}.

4.3.2 Combining with Bounding Box Detection

w/ BBox	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{kp}	AP^{kp}_{50}	AP^{kp}_{75}
	-	-	-	**63.1**	**85.6**	**68.8**
✓	55.3	81.5	59.9	61.5	84.3	67.5

TABLE 4.4. **Our framework with person box detection on COCO `minival`.** The proposed framework can achieve reasonable person detection results (55.3% in AP). As a reference, the Faster R-CNN person detector in Mask R-CNN He et al., 2017 achieves 53.7% in AP.

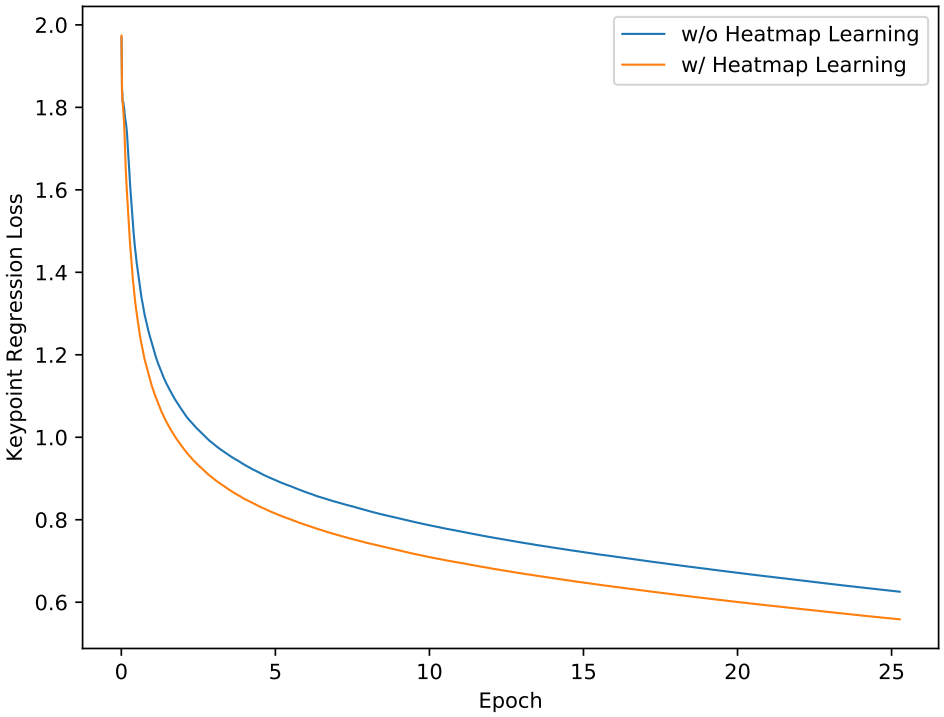

FIGURE 4.4. **The loss curves of training with or without the heatmap learning.** As shown in the figure, with the heatmap learning, the model can achieve a significantly lower loss value and thus much better performance.

As mentioned before, by simply adding a box branch, the proposed framework can simultaneously detect bounding boxes and keypoints. Here we confirm it by the experiment. The box detection is implemented by adding the original box detection head of FCOS to the framework. As shown in Table 4.4, our framework can achieve a reasonable person detection performance, which is similar to the Faster R-CNN detector in Mask R-CNN (55.3% vs. 53.7%). Although Mask R-CNN can also simultaneously detect boxes and keypoints, we further unify the two tasks into the same methodology.

4.3.3 Comparisons with State-of-the-art Methods

In this section, we evaluate the proposed end-to-end keypoint detection framework on MS-COCO `test-dev` split and compare it with previous bottom-up and top-down ones. We make use of the best model in ablation experiments. As shown in Table 4.5, without any bells and whistles (*e.g.*, multi-scale and flipping testing, the refining in Cao et al., 2017; Newell, Huang, and Deng, 2017, and any other tricks), the end-to-end framework achieves 62.2% and 63.3% in AP^{kp} on COCO `test-dev` split, with ResNet-50 and ResNet-101 as the backbone, respectively. With multi-scale testing, our framework can achieve 63.0% and 64.8% with ResNet-50 and ResNet-101, respectively.

Compared to Bottom-up Methods. The performance of our ResNet-50 based

method	AP^{kp}	AP^{kp}_{50}	AP^{kp}_{75}	AP^{kp}_{M}	AP^{kp}_{L}
Top-down Methods					
Mask R-CNN He et al., 2017	62.7	87.0	68.4	57.4	71.1
CPN Chen et al., 2018	72.1	91.4	80.0	68.7	77.2
RMPE Fang et al., 2017	72.3	89.2	79.1	68.0	78.6
CFN Huang, Gong, and Tao, 2017	72.6	86.1	69.7	78.3	64.1
HRNet-W48 Sun et al., 2019	**75.5**	**92.5**	**83.3**	**71.9**	**81.5**
Bottom-up Methods					
CMU-Pose*† Cao et al., 2017	61.8	84.9	67.5	57.1	68.2
AE Newell, Huang, and Deng, 2017	56.6	81.8	61.8	49.8	67.0
AE*	62.8	84.6	69.2	57.5	70.6
AE*†	65.5	86.8	72.3	60.6	72.6
PersonLab Papandreou et al., 2018	65.5	87.1	71.4	61.3	71.5
PersonLab†	**67.8**	**89.0**	**75.4**	**64.1**	**75.5**
Direct End-to-end Methods					
Ours (R-50)	62.2	86.4	68.2	56.7	69.8
Ours (R-50)†	63.0	86.8	69.3	59.1	69.3
Ours (R-101)	63.3	86.7	69.4	57.8	71.2
Ours (R-101)†	**64.8**	**87.8**	**71.1**	**60.4**	**71.5**

TABLE 4.5. **The performance of our proposed end-to-end framework on COCO `test-dev` split.** * and † respectively denote using refining and multi-scale testing.

end-to-end framework is better (62.2% vs. 61.8%) than the strong baseline CMU-Pose Cao et al., 2017 that uses multi-scale testing and post-processing with CPM Wei et al., 2016, and filters the results with an object detector. Our framework also achieves much better performance than the bottom-up method AE Newell, Huang, and Deng, 2017 (63.3% vs. 56.6%) and is even better than the method with refining. Compared to PersonLab, with the same backbone ResNet-101 and single-scale testing, our proposed framework also has a competitive performance with it (63.3% vs. 65.5%). Note that our proposed framework is much simpler than these bottom-up methods, in both training and testing.

Compared to Top-down Methods. With the same backbone ResNet-50, the proposed method has a similar performance with previous strong baseline Mask R-CNN (62.2% vs. 62.7%). Our model is still behind other top-down methods. However, it is worth noting that these methods often employ a separate box detector to obtain person instances. These instances are then cropped from the original image and a single person pose estimation method is separately applied to each the cropped image to obtain the final results. As noted before, this strategy is slow as it cannot take advantage of the sharing computation mechanism in CNNs. In contrast, our proposed end-to-end framework is much simpler and faster since it directly maps the raw input images to the final instance-aware keypoint detections with a fully convolutional network.

Timing. The averaged inference time of our model on COCO `minival` split is 74ms and 87ms per image with ResNet-50 and ResNet-101, respectively, which is slightly faster than Mask R-CNN with the same hardware and backbones (Mask R-CNN takes

FIGURE 4.5. **Visualization results of KPAlign on MS-COCO `minival`.** The first image in each group shows the outputs of the locator in KPAlign (*i.e.*, the locations where the sampler samples the features used to predict the keypoints). The orange point denotes the original location where the features will be used if KPAlign is not used. The second image shows the final keypoint detection results. As shown in the figure, the proposed KPAlign can make use of the features near the keypoints to predict them. The final image shows that the ground-truth keypoints. Zoom in for a better look.

78ms per image with ResNet-50). Additionally, the running time of Mask R-CNN depends on the number of the instances while our model, similar to one-stage object detectors, has nearly constant inference time for any number of instances.

4.3.4 Visualization of KPAlign

The visualization results of KPAlign are shown in Fig. 4.5. As shown in the figure, the proposed KPAlign can make use of the features near the keypoints to predict them. Thus, the feature vectors can avoid encoding the keypoints far from their spatial location, which results in improved performance.

4.3.5 Visualization of Keypoint Detections

We show more visualization results of DirectPose in Fig. 4.6. As shown in the figure, the proposed DirectPose can directly detect all the desired instance-aware keypoints

FIGURE 4.6. **Visualization results of the proposed DirectPose on MS-COCO `minival`.** DirectPose can directly detect a wide range of poses. Note that some small-scale people do not have ground-truth keypoint annotations in the training set of MS-COCO, thus they might be missing when testing.

without the need for the grouping post-processing or box detection. The results of the proposed DirectPose with simultaneous box detection are also shown in Fig. 4.7.

4.4 Conclusions

We have proposed the first direct end-to-end human pose estimation framework, termed DirectPose. Our proposed model is end-to-end trainable and can directly map a raw input image to the desired instance-aware keypoint detections with almost constant inference time, eliminating the grouping post-processing in bottom-up methods or the box detection and RoI operations in top-down ones. Experiments demonstrate that the new end-to-end method can obtain competitive or better performance than previous bottom-up and top-down methods.

FIGURE 4.7. **Visualization results of the proposed DirectPose with the simultaneous box detection on MS-COCO `minival`.**

Chapter 5

Conditional Convolutions for Instance Segmentation

5.1 Introduction

Instance segmentation is a fundamental yet challenging task in computer vision, which requires an algorithm to predict a per-pixel mask with a category label for each instance of interest in an image. Despite a few works being proposed recently, the dominant framework for instance segmentation is still the two-stage method Mask R-CNN He et al., 2017, which casts instance segmentation into a two-stage detection-and-segmentation task. Mask R-CNN first employs an object detector Faster R-CNN to predict a box for each instance. Then for each instance, regions-of-interest (ROIs) are cropped from the networks' feature maps using the ROIAlign operation. To predict the final masks for each instance, a compact fully convolutional network (FCN) (*i.e.*, mask head) is applied to these ROIs to perform foreground/background segmentation. However, this ROI-based method may have the following drawbacks. 1) Since ROIs are often axis-aligned boxes, for objects with irregular shapes, they may contain an excessive amount of irrelevant image content including background and other instances. This issue may be mitigated by using rotated ROIs, but with the price of a more complex pipeline. 2) In order to distinguish between the foreground instance and the background stuff or instance(s), the mask head requires a relatively larger receptive field to encode sufficiently large context information. As a result, a stack of 3×3 convolutions is needed in the mask head (*e.g.*, four 3×3 convolutions with 256 channels in Mask R-CNN). It considerably increases computational complexity of the mask head, resulting that the inference time significantly varies in the number of instances. 3) ROIs are typically of different sizes. In order to use effective batched computation in modern deep learning frameworks A. Paszke et al., 2019; M. Abadi et al., 2016, a resizing operation is often required to resize the cropped regions into patches of the same size. For instance, Mask R-CNN resizes all the cropped regions to 14×14 (upsampled to 28×28 using a deconvolution), which restricts the output resolution of instance segmentation, as large instances would require higher resolutions to retain details at the boundary.

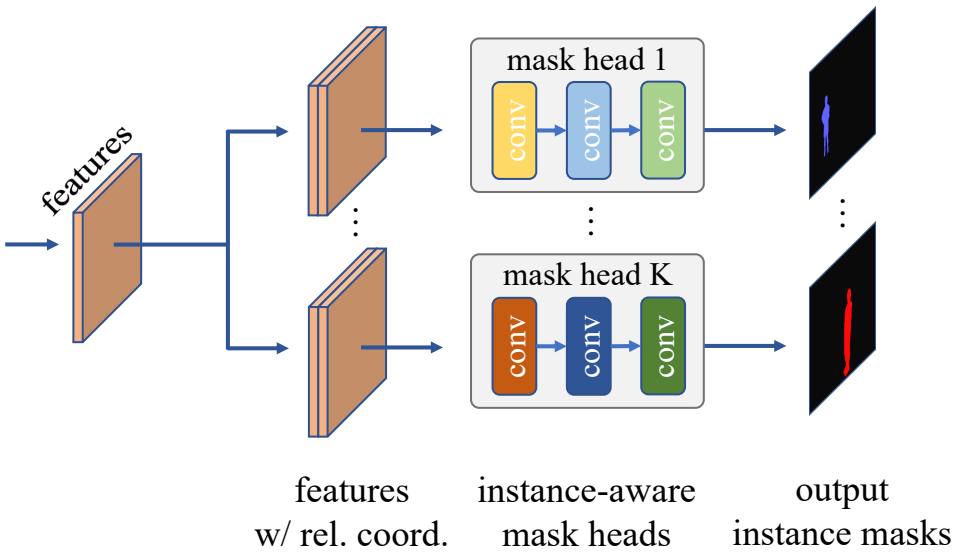

FIGURE 5.1. **CondInst uses instance-aware mask heads to predict the masks for each instance.** K is the number of instances to be predicted. The filters in the mask head vary with different instances, which are conditioned on the target instance. ReLU is used as the activation function (excluding the last conv. layer).

In computer vision, the closest task to instance segmentation is semantic segmentation, for which fully convolutional networks (FCNs) have shown dramatic success Long, Shelhamer, and Darrell, 2015; Chen et al., 2017a; Tian et al., 2019a; He et al., 2019b; Liu et al., 2020a. FCNs also have shown excellent performance on many other per-pixel prediction tasks ranging from low-level image processing such as denoising, super-resolution; to mid-level tasks such as optical flow estimation and contour detection; and high-level tasks including the single-shot object detection such as FCOS, monocular depth estimation Liu et al., 2016a; Yin et al., 2019; Yin et al., 2020; Bian et al., 2019; Bian et al., 2020 and counting Boominathan, Kruthiventi, and Babu, 2016. However, almost all the instance segmentation methods based on FCNs[1] lag behind state-of-the-art ROI-based methods. Why do the versatile FCNs perform unsatisfactorily on instance segmentation? We observe that the major difficulty of applying FCNs to instance segmentation is that the similar image appearance may require different predictions but FCNs struggle at achieving this. For example, if two persons A and B with the similar appearance are in an input image, when predicting the instance mask of A, the FCN needs to predict B as background w.r.t. A, which can be difficult as they look similar in appearance. Therefore, an ROI operation is used to crop the person of interest, *i.e.*, A; and filter out B. The ROI operation is the *de facto* core operation making the model attend to instances. In contrast, CondInst attends to the instances by using *instance-sensitive convolutional filters*.

Specifically, we advocate a new solution that uses instance-aware FCNs for instance

[1]By FCNs, we mean the vanilla FCNs in Long, Shelhamer, and Darrell, 2015 that only involve convolutions and pooling.

mask prediction. In other words, instead of using a standard ConvNet with a fixed set of convolutional filters as the mask head for predicting all instances, the network parameters are adapted according to the instance to be predicted. Inspired by dynamic filtering networks Jia et al., 2016 and CondConv Yang et al., 2019a, for each instance, a controller sub-network (see Fig. 5.3) dynamically generates the mask FCN network parameters (conditioned on the center area of the instance), which is then used to predict the mask of this instance. It is expected that the network parameters can encode the characteristics (*e.g.*, relative position, shape and appearance) of this instance, and only fires on the pixels of this instance, which thus bypasses the difficulty mentioned above. These conditional mask heads are applied to the whole feature maps, *eliminating the need for ROI operations*. At the first glance, the idea may not work well as instance-wise mask heads may incur a large number of network parameters provided that some images contain as many as dozens of instances. However, we show that a very compact FCN mask head with dynamically-generated filters can already outperform previous ROI-based Mask R-CNN, resulting in much reduced computational complexity per instance than that of the mask head in Mask R-CNN.

We summarize our main contributions as follow.

- We attempt to solve instance segmentation from a new perspective. To this end, we propose the CondInst instance segmentation framework, which achieves improved instance segmentation performance than existing methods such as Mask R-CNN while being faster. To our knowledge, this is the first time that a new instance segmentation framework outperforms recent state-of-the-art both in accuracy and speed.
- CondInst is fully convolutional and avoids the aforementioned resizing operation used in many existing methods, as CondInst does not rely on ROI operations. Without having to resize feature maps leads to high-resolution instance masks with more accurate edges.
- Unlike previous methods, in which the filters in its mask head are fixed for all the instances once trained, the filters in our mask head are dynamically generated and conditioned on instances. As the filters are only asked to predict the mask of only one instance, it largely eases the learning requirement and thus reduces the load of the filters. As a result, the mask head can be extremely light-weight, significantly reducing the inference time per instance. Compared with the bounding box detector FCOS, CondInst needs only $\sim$10% more computational time, even processing the maximum number of instances per image (*i.e.*, 100 instances).
- Without resorting to longer training schedules as needed in recent works Chen et al., 2019b; Bolya et al., 2019a, CondInst achieves state-of-the-art performance while being faster in inference. We hope that CondInst can be a new

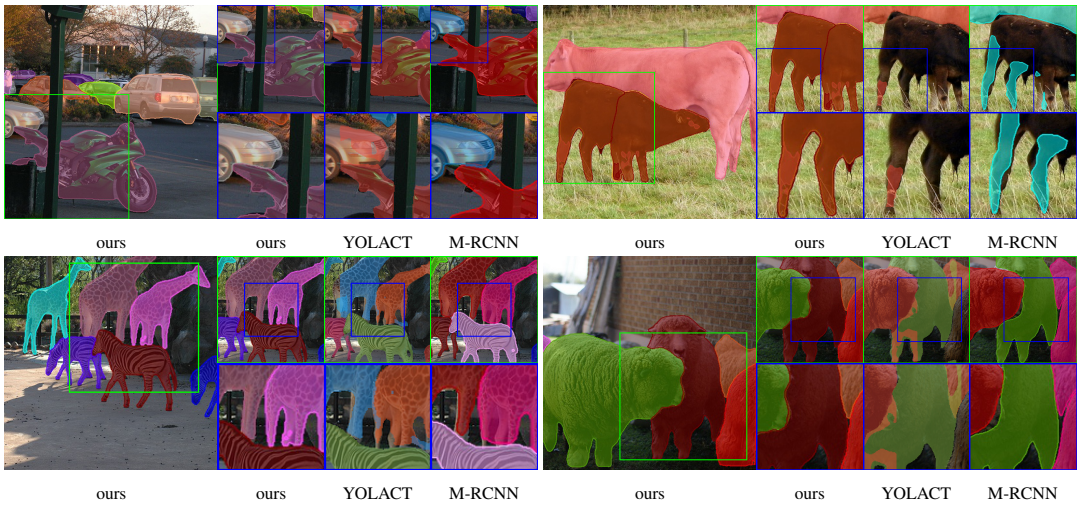

FIGURE 5.2. **Qualitative comparisons with other methods.** We compare the proposed CondInst against YOLACT Bolya et al., 2019a and Mask R-CNN He et al., 2017. Our masks are generally of higher quality (*e.g.*, preserving more details).

strong alternative to popular methods such as Mask R-CNN for the instance segmentation task.

Moreover, CondInst can be immediately applied to panoptic segmentation due to its flexible design. We believe that with minimal re-design effort, the proposed CondInst can be used to solve all instance-level recognition tasks that were previously solved with an ROI-based pipeline.

5.2 Instance Segmentation with CondInst

5.2.1 Overall Architecture

Given an input image $I \in \mathbb{R}^{H \times W \times 3}$, the goal of instance segmentation is to predict the pixel-level mask and the category of each instance of interest in the image. The ground-truths are defined as $\{(M_i, c_i)\}$, where $M_i \in \{0, 1\}^{H \times W}$ is the mask for the i-th instance and $c_i \in \{1, 2, ..., C\}$ is the category. C is 80 on MS-COCO Lin et al., 2014. Unlike semantic segmentation, which only requires to predict one mask for an input image, instance segmentation needs to predict a variable number of masks, depending on the number of instances in the image. This poses a challenge when applying traditional FCNs Long, Shelhamer, and Darrell, 2015 to instance segmentation. In this work, our core idea is that for an image with K instances, K different mask heads will be dynamically generated, and each mask head will contain the characteristics of its target instance in their filters. As a result, when the mask is applied to an input, it will only fire on the pixels of the instance, thus producing the mask prediction of the instance. We illustrate the process in Fig. 5.1.

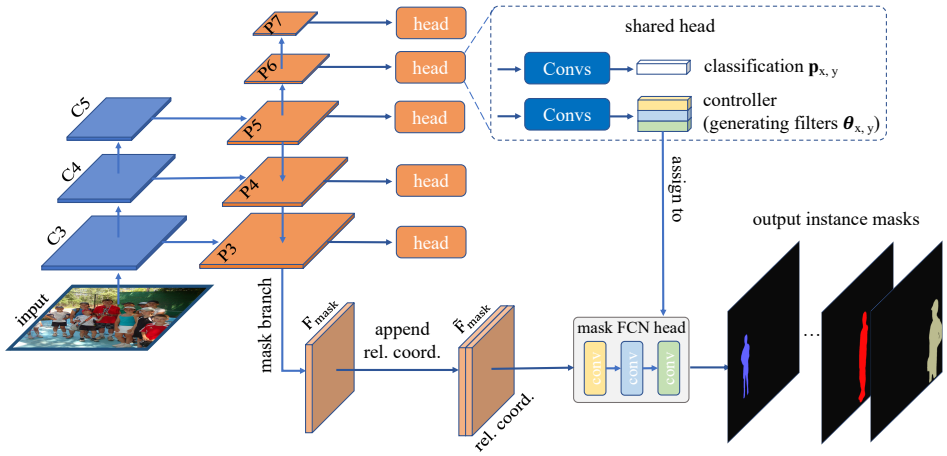

FIGURE 5.3. **The overall architecture of CondInst.** C_3, C_4 **and** C_5 **are the feature maps of the backbone network (*e.g.*, ResNet-50).** P_3 to P_7 are the FPN feature maps as in FCOS. $\mathbf{F}_{mask}$ is the mask branch's output and $\tilde{\mathbf{F}}_{mask}$ is obtained by concatenating the relative coordinates to $\mathbf{F}_{mask}$. The classification head is the same as in FCOS. The controller generates the filter parameters $\boldsymbol{\theta}_{x,y}$ of the mask head for the instance. Note that the heads in the dashed box are repeatedly applied to $P_3 \cdots P_7$. The mask head is instance-aware, and is applied to $\tilde{\mathbf{F}}_{mask}$ as many times as the number of instances in the image (refer to Fig. 5.1).

Recall that Mask R-CNN employs an object detector to predict the boxes of the instances in the input image. The boxes are actually the way that Mask R-CNN represents instances. Similarly, CondInst employs the instance-aware filters to represent the instances. In other words, instead of encoding the instance concept into the boxes, CondInst implicitly encodes it into the parameters of the mask heads, which is a much more flexible way. For example, it can easily represent the irregular shapes that are hard to be tightly enclosed by a box. This is one of CondInst's advantages over the previous ROI-based methods.

Similar to the way that ROI-based methods obtain boxes, the instance-aware filters can also be obtained with an object detector. In this work, we build CondInst on the object detector FCOS. The elimination of anchor-boxes in FCOS can also save the number of parameters and the amount of computation of CondInst. As shown in Fig. 5.3, following FCOS, we make use of the feature maps $\{P_3, P_4, P_5, P_6, P_7\}$ of feature pyramid networks (FPNs) Lin et al., 2017a, whose down-sampling ratios are 8, 16, 32, 64 and 128, respectively. As shown in Fig. 5.3, on each feature level of the FPN, some functional layers (in the dash box) are applied to make instance-related predictions. For example, the class of the target instance and the dynamically-generated filters for the instance. In this sense, CondInst can be viewed as the same as Mask R-CNN, both of which first attend to instances in an image and then predict the pixel-level masks of the instances (*i.e.*, instance-first).

Besides the detector, as shown in Fig. 5.3, there is also a mask branch, which provides the feature maps that our generated mask heads take as inputs to predict the desired instance mask. The feature maps are denoted by $\mathbf{F}_{mask} \in \mathbb{R}^{H_{mask} \times W_{mask} \times C_{mask}}$. The mask branch is connected to FPN level P_3 and thus its output resolution is $\frac{1}{8}$ of the input image resolution. The mask branch has four 3×3 convolutions with 128 channels before the last layer. Afterwards, in order to reduce the number of the generated parameters, the last layer of the mask branch reduces the number of channels from 128 to 8 (*i.e.*, $C_{mask} = 8$). Surprisingly, using $C_{mask} = 8$ can already achieve superior performance and using a larger C_{mask} here (*e.g.*, 16) cannot improve the performance, as shown in our experiments. Even more aggressively, using $C_{mask} = 2$ only degrades the performance by $\sim 0.3\%$ in mask AP. Moreover, as shown in Fig. 5.3, $\mathbf{F}_{mask}$ is combined with a map of the coordinates, which are relative coordinates from all the locations on $\mathbf{F}_{mask}$ to the location (x, y) (*i.e.*, where the filters of the mask head are generated). Then, the combination is sent to the mask head to predict the instance mask. The relative coordinates provide a strong cue for predicting the instance mask, as shown in our experiments. Moreover, a single sigmoid is used as the final output of the mask head, and thus the mask prediction is class-agnostic. The class of the instance is predicted by the classification head in parallel with the controller, as shown in Fig. 5.3.

The resolution of the original mask prediction is same as the resolution of F_{mask}, which is $\frac{1}{8}$ of the input image resolution. In order to produce high-resolution instance masks, a bilinear upsampling is used to upsample the mask prediction by 4, resulting in 400×512 mask prediction (if the input image size is 800×1024). We will show that the upsampling is crucial to the final instance segmentation performance of CondInst in experiments. Note that the mask's resolution is much higher than that of Mask R-CNN (only 28×28 as mentioned before).

5.2.2 Network Outputs and Training Targets

Similar to FCOS, each location on the FPN's feature maps P_i either is associated with an instance, thus being a positive sample, or is considered a negative sample. The associated instance and label for each location are determined as follows. Let us consider the feature maps $P_i \in \mathbb{R}^{H \times W \times C}$ and let s be its down-sampling ratio. As shown in previous works Ren et al., 2015; He et al., 2015, a location (x, y) on the feature maps can be mapped back onto the input image as $(\lfloor \frac{s}{2} \rfloor + xs, \lfloor \frac{s}{2} \rfloor + ys)$. If the mapped location falls in the center region of an instance, the location is considered to be responsible for the instance. Any locations outside the center regions are labeled as negative samples. The center region is defined as the box $(c_x - rs, c_y - rs, c_x + rs, c_y + rs)$, where (c_x, c_y) denotes the mass center of the instance, s is the down-sampling ratio of P_i and r is a constant scalar being 1.5 as in FCOS. As shown in Fig. 5.3, at a location (x, y) on P_i, CondInst has the following output heads.

Classification Head. The classification head predicts the class of the instance associated with the location. The ground-truth target is the instance's class c_i or 0 (*i.e.*, background). As in FCOS, the network predicts a C-D vector $\boldsymbol{p}_{x,y}$ for the classification and each element in $\boldsymbol{p}_{x,y}$ corresponds to a binary classifier, where C is the number of categories.

Controller Head. The controller head, which has the same architecture as the above classification head, is used to predict the parameters of the mask head for the instance at the location. The mask head predicts the mask of this particular instance. This is the core contribution of our work. To predict the parameters, we concatenate all the parameters of the filters (*i.e.*, weights and biases) together as an N-D vector $\boldsymbol{\theta}_{x,y}$, where N is the total number of the parameters. Accordingly, the controller head has N output channels. The mask head is a very compact FCN architecture, which has three 1×1 convolutions, each having 8 channels and using ReLU as the activation function except for the last one. No normalization layer such as batch normalization Ioffe and Szegedy, 2015 is used here. The last layer has 1 output channel and uses sigmoid to predict the probability of being foreground. The mask head has 169 parameters in total ($\#weights = (8+2) \times 8(conv1) + 8 \times 8(conv2) + 8 \times 1(conv3)$ and $\#biases = 8(conv1) + 8(conv2) + 1(conv3)$). As mentioned before, the generated filters contain information about the instance at the location, and thus, ideally, the mask head with the filters will only fire on the pixels of the instance, even taking as the input the whole feature maps.

Box Head. The box head is the same as that in FCOS, which predicts a 4-D vector encoding the four distances from the location to the four boundaries of the bounding-box of the target instance. Conceptually, CondInst can eliminate the box head since CondInst needs no ROIs. However, we find that if we make use of box-based NMS, the inference time will be much reduced. Thus, we still predict boxes in CondInst. We would like to highlight that the predicted boxes are *only* used in NMS and do not involve any ROI operations. Moreover, as shown in Table 5.6, the box prediction can be removed if no box information is used (*e.g.*, mask NMS Wang et al., 2020c). This is fundamentally different from previous ROI-based methods, in which the box prediction is mandatory.

Center-ness Head. Like FCOS, at each location, we also predict a center-ness score. The center-ness score depicts how the location deviates from the center of the target instance. In inference, it is used to down-weight the boxes predicted by the locations far from the center, which might be unreliable. We refer readers to Chapter 3 for the details.

5.2.3 Loss Functions

Formally, the overall loss function of CondInst can be formulated as,

$$L_{overall} = L_{fcos} + \lambda L_{mask}, \tag{5.1}$$

where L_{fcos} and L_{mask} denote the original loss of FCOS and the loss for instance masks, respectively. λ being 1 in this work is used to balance the two losses. We refer readers to FCOS for the details of L_{fcos}. L_{mask} is defined as,

$$L_{mask}(\{\boldsymbol{\theta}_{x,y}\}) = \frac{1}{N_{\text{pos}}} \sum_{x,y} \mathbb{1}_{\{c^*_{x,y}>0\}} L_{dice}(MaskHead(\tilde{\mathbf{F}}_{x,y}; \boldsymbol{\theta}_{x,y}), \mathbf{M}^*_{x,y}), \tag{5.2}$$

where $c^*_{x,y}$ is the classification label of location (x, y), which is the class of the instance associated with the location or 0 (*i.e.*, background) if the location is not associated with any instance. N_{pos} is the number of locations where $c^*_{x,y} > 0$. $\mathbb{1}_{\{c^*_{x,y}>0\}}$ is the indicator function, being 1 if $c^*_{x,y} > 0$ and 0 otherwise. $\boldsymbol{\theta}_{x,y}$ is the generated filters' parameters at location (x, y). $\tilde{\mathbf{F}}_{x,y} \in \mathbb{R}^{H_{mask} \times W_{mask} \times (C_{mask}+2)}$ is the combination of $\mathbf{F}_{mask}$ and a map of coordinates $\mathbf{O}_{x,y} \in \mathbb{R}^{H_{mask} \times W_{mask} \times 2}$. As described before, $\mathbf{O}_{x,y}$ is the relative coordinates from all the locations on $\mathbf{F}_{mask}$ to (x, y) (*i.e.*, the location where the filters are generated). $MaskHead$ denotes the mask head, which consists of a stack of convolutions with dynamic parameters $\boldsymbol{\theta}_{x,y}$. $\mathbf{M}^*_{x,y} \in \{0, 1\}^{H \times W \times C}$ is the mask of the instance associated with location (x, y). L_{dice} is the dice loss as in Milletari, Navab, and Ahmadi, 2016, which is used to overcome the foreground-background sample imbalance. We do not employ focal loss here as it requires special initialization, which cannot be realized if the parameters are dynamically generated. Note that, in order to compute the loss between the predicted mask and the ground-truth mask $\mathbf{M}^*_{x,y}$, they are required to have the same size. As mentioned before, the prediction is upsampled by 4 and thus the resolution of the final prediction is half of that of the ground-truth mask $\mathbf{M}^*_{x,y}$. We downsample $\mathbf{M}^*_{x,y}$ by 2 to make the sizes equal. These operations are omitted in Eq. (5.2) for clarification.

Moreover, as shown in YOLACT and BlendMask Bolya et al., 2019a; Chen et al., 2020, the instance segmentation task can benefit from a joint semantic segmentation task. Thus, we also conduct experiments with the joint semantic segmentation task. However, unless explicitly specified, all the experiments in the paper are *without* the semantic segmentation task. If used, the semantic segmentation loss is added to $L_{overall}$.

5.2.4 Inference

Given an input image, we forward it through the network to obtain the outputs including classification confidence $\boldsymbol{p}_{x,y}$, center-ness scores, box prediction $\boldsymbol{t}_{x,y}$ and the generated parameters $\boldsymbol{\theta}_{x,y}$. We first follow the steps in FCOS to obtain the box detections. Afterwards, box-based NMS with the threshold being 0.6 is used to remove duplicated detections and then the top 100 boxes are used to compute masks. Different from FCOS, these boxes are also associated with the filters generated by the controller. Let us assume that K boxes remain after the NMS, and thus we have K groups of the generated filters. The K groups of filters are used to produce K instance-specific mask heads. These instance-specific mask heads are applied, in the fashion of FCNs,

depth	time	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
1	**2.2**	30.9	52.9	31.4	14.0	33.3	45.1
2	3.3	35.5	56.1	37.8	17.0	38.9	50.8
3	4.5	**35.7**	**56.3**	37.8	17.1	**39.1**	50.2
4	5.6	**35.7**	56.2	**37.9**	**17.2**	38.7	**51.5**

TABLE 5.1. **Instance segmentation results by varying the depth of the mask head (width = 8) on MS-COCO `val2017` split.** "depth": the number of layers in the mask head. "time": the milliseconds that the mask head takes for processing 100 instances.

width	time	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
2	**2.5**	34.1	55.4	35.8	15.9	37.2	49.1
4	2.6	35.6	**56.5**	**38.1**	17.0	**39.2**	**51.4**
8	4.5	**35.7**	**56.3**	37.8	17.1	39.1	50.2
16	4.7	35.6	56.2	37.9	**17.2**	38.8	50.8

TABLE 5.2. **Instance segmentation results by varying the width of the mask head (with depth = 3) on MS-COCO `val2017` split.** "time": the milliseconds that the mask head takes for processing 100 instances.

to the $\tilde{\mathbf{F}}_{x,y}$ (*i.e.*, the combination of $\mathbf{F}_{mask}$ and $\mathbf{O}_{x,y}$) to predict the masks of the instances. Since the mask head is a very compact network (three 1×1 convolutions with 8 channels and 169 parameters in total), the overhead of computing masks is extremely small. For example, even with 100 detections (*i.e.*, the maximum number of detections per image on MS-COCO), only less 5 milliseconds in total are spent on the mask heads, which only adds $\sim 10\%$ computational time to the base detector FCOS. In contrast, the mask head of Mask R-CNN has four 3×3 convolutions with 256 channels, thus having more than 2.3M parameters and taking longer computational time.

5.3 Experiments

We evaluate CondInst on the large-scale benchmark MS-COCO Lin et al., 2014. Following the common practice He et al., 2017; Lin et al., 2017b, our models are trained with split `train2017` (115K images) and all the ablation experiments are evaluated on split `val2017` (5K images). Our main results are reported on the `test-dev` split (20K images).

5.3.1 Implementation Details

Unless specified, we make use of the following implementation details. Following FCOS, ResNet-50 He et al., 2016 is used as our backbone network and the weights pre-trained on ImageNet Deng et al., 2009 are used to initialize it. For the newly added layers, we initialize them as in FCOS. Our models are trained with stochastic gradient descent (SGD) over 8 V100 GPUs for 90K iterations with the initial learning rate being 0.01 and a mini-batch of 16 images. The learning rate is reduced by a factor

C_{mask}	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
1	34.8	55.9	36.9	16.7	38.0	50.1
2	35.4	56.2	37.6	16.9	38.9	50.4
4	35.5	56.2	**37.9**	17.0	39.0	50.8
8	**35.7**	**56.3**	37.8	**17.1**	**39.1**	50.2
16	35.5	56.1	37.7	16.4	**39.1**	**51.2**

TABLE 5.3. **The instance segmentation results by varying the number of channels of the mask branch output (*i.e.*, C_{mask}) on MS-COCO `val2017` split.**

of 10 at iteration $60K$ and $80K$, respectively. Weight decay and momentum are set as 0.0001 and 0.9, respectively. Following Detectron2 Wu et al., 2019, the input images are resized to have their shorter sides in $[640, 800]$ and their longer sides less or equal to 1333 during training. Left-right flipping data augmentation is also used during training. When testing, we do not use any data augmentation and only the scale of the shorter side being 800 is used. The inference time in this work is measured on a single V100 GPU with 1 image per batch.

w/ abs. coord.	w/ rel. coord.	w/ $\mathbf{F}_{mask}$	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AR_1	AR_{10}	AR_{100}
		✓	31.4	53.5	32.1	15.6	34.4	44.7	28.4	44.1	46.2
	✓		31.3	54.9	31.8	16.0	34.2	43.6	27.1	43.3	45.7
✓		✓	32.0	53.3	32.9	14.7	34.2	46.8	28.7	44.7	46.8
	✓	✓	**35.7**	**56.3**	**37.8**	**17.1**	**39.1**	**50.2**	**30.4**	**48.8**	**51.5**

TABLE 5.4. **Ablation study of the input to the mask head on MS-COCO `val2017` split.** As shown in the table, without the relative coordinates, the performance drops significantly from 35.7% to 31.4% in mask AP. Using the absolute coordinates cannot improve the performance remarkably. If the mask head only takes as input the relative coordinates (*i.e.*, no appearance in this case), CondInst also achieves modest performance.

5.3.2 Architectures of the Mask Head

In this section, we discuss the design choices of the mask head in CondInst. To our surprise, the performance is insensitive to the architectures of the mask head. Our baseline is the mask head of three 1×1 convolutions with 8 channels (*i.e.*, width $= 8$). As shown in Table 5.1 (3rd row), it achieves 35.7% in mask AP. Next, we first conduct experiments by varying the depth of the mask head. As shown in Table 5.1, apart from the mask head with depth being 1, all other mask heads (*i.e.*, depth $= 2, 3$ and 4) attain similar performance. The mask head with depth being 1 achieves inferior performance as in this case the mask head is actually a linear mapping, which has overly weak capacity. Moreover, as shown in Table 5.2, varying the width (*i.e.*, the number of the channels) does not result in a remarkable performance change either as long as the width is in a reasonable range. We also note that our mask head is extremely light-weight as the filters in our mask head are dynamically generated. As shown in Table 5.1, our baseline mask head only takes 4.5 ms per 100 instances (the maximum number of instances on MS-COCO), which suggests that our mask head

factor	resolution	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
1	1/8	34.4	55.4	36.2	15.1	38.4	50.8
2	1/4	**35.8**	**56.4**	**38.0**	17.0	**39.3**	**51.1**
4	1/2	35.7	56.3	37.8	**17.1**	39.1	50.2

TABLE 5.5. **The instance segmentation results on MS-COCO `val2017` split by changing the factor used to upsample the mask predictions.** "resolution" denotes the resolution ratio of the mask prediction to the input image.

only adds small computational overhead to the base detector. Moreover, our baseline mask head only has 169 parameters in total. In sharp contrast, the mask head of Mask R-CNN He et al., 2017 has more than 2.3M parameters and takes $\sim 2.5\times$ computational time (11.4 ms per 100 instances).

5.3.3 Design Choices of the Mask Branch

We further investigate the impact of the mask branch. We first change C_{mask}, which is the number of channels of the mask branch's output feature maps (*i.e.*, $\mathbf{F}_{mask}$). As shown in Table 5.3, as long as C_{mask} is in a reasonable range (*i.e.*, from 2 to 16), the performance keeps almost the same. $C_{mask} = 8$ is optimal and thus we use $C_{mask} = 8$ in all other experiments by default.

As mentioned before, before taken as the input of the mask heads, the mask branch's output $\mathbf{F}_{mask}$ is concatenated with a map of relative coordinates, which provides a strong cue for the mask prediction. As shown in Table 5.4 (2nd row), the performance drops significantly if the relative coordinates are removed (35.7% vs. 31.4%). The significant performance drop implies that the generated filters not only encode the appearance cues but also encode the shape (and relative position) of the target instance. It can also be evidenced by the experiment only using the relative coordinates. As shown in Table 5.4 (2rd row), only using the relative coordinates can also obtain decent performance (31.3% in mask AP). We would like to highlight that unlike Mask R-CNN, which encodes the shape of the target instance by a box, CondInst implicitly encodes the shape into the generated filters, which can easily represent any shapes including irregular ones and thus is much more flexible. We also experiment with the absolute coordinates, but it cannot largely boost the performance as shown in Table 5.4 (32.0%). This suggests that the generated filters mainly carry translation-invariant cues such as shapes and relative position, which is preferable.

5.3.4 How Important to Upsample Mask Predictions?

As mentioned before, the original mask prediction is upsampled and the upsampling is of great importance to the final performance. We confirm this in the experiment. As shown in Table 5.5, without using the upsampling (1st row in the table), in this case CondInst can produce the mask prediction with $\frac{1}{8}$ of the input image resolution, which merely achieves 34.4% in mask AP because most of the details (*e.g.*, the boundary) are lost. If the mask prediction is upsampled by factor = 2, the performance can be

NMS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
box	**35.7**	56.3	**37.8**	17.1	39.1	50.2
mask	**35.7**	**56.7**	37.7	**17.2**	**39.2**	**50.5**

TABLE 5.6. **Instance segmentation results with different NMS algorithms.** Mask-based NMS can obtain the same overall performance as box-based NMS, which suggests that CondInst can totally eliminate the box detection.

significantly improved by 1.4% in mask AP (from 34.4% to 35.8%). In particular, the improvement on small objects is large (from 15.1% to 17.0), which suggests that the upsampling can greatly retain the details of objects. Increasing the upsampling factor to 4 slightly worsens the performance (from 35.8% to 35.7% in mask AP), probably due to the relatively low-quality annotations of MS-COCO. We use factor = 4 in all other models as it has the potential to produce high-resolution instance masks.

5.3.5 CondInst without box Detection

Although we still keep the box detection branch in CondInst, it is conceptually feasible to totally eliminate it if we make use of the NMS using no boxes. In this case, all the foreground samples (determined by the classification head) will be used to compute instance masks, and the duplicated masks will be removed by mask-based NMS. As shown in Table 5.6, with the mask-based NMS, the same overall performance can be obtained as box-based NMS (35.7% vs. 35.7% in mask AP).

5.3.6 Comparisons with State-of-the-art Methods

We compare CondInst against previous state-of-the-art methods on MS-COCO `test-dev` split. As shown in Table 5.7, with 1× learning rate schedule (*i.e.*, $90K$ iterations), CondInst outperforms the original Mask R-CNN by 0.8% (35.4% vs. 34.6%). CondInst also achieves a much faster speed than the original Mask R-CNN (49ms vs. 65ms per image on a single V100 GPU). To our knowledge, it is the first time that a new and simpler instance segmentation method, without any bells and whistles outperforms Mask R-CNN both in accuracy and speed. CondInst also obtains better performance (35.9% vs. 35.5%) and on-par speed (49ms vs 49ms) than the well-engineered Mask R-CNN in `Detectron2` (*i.e.*, Mask R-CNN* in Table 5.7). Furthermore, with a longer training schedule (*e.g.*, 3×) or a stronger backbone (*e.g.*, ResNet-101), a consistent improvement is achieved as well (37.8% vs. 37.5% with ResNet-50 3× and 39.1% vs. 38.8% with ResNet-101 3×). Moreover, as shown in Table 5.7, with the auxiliary semantic segmentation task, the performance can be boosted from 37.8% to 38.8% (ResNet-50) or from 39.1% to 40.1% (ResNet-101), without increasing the inference time. For fair comparisons, all the inference time here is measured by ourselves on the same hardware with the official codes.

We also compare CondInst with the recently-proposed instance segmentation methods. Only with half training iterations, CondInst surpasses TensorMask Chen et al.,

method	backbone	aug.	sched.	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Mask R-CNN He et al., 2017	R-50-FPN		1×	34.6	**56.5**	36.6	15.4	36.3	**49.7**
CondInst	R-50-FPN		1×	**35.4**	56.4	**37.6**	**18.4**	**37.9**	46.9
Mask R-CNN*	R-50-FPN	✓	1×	35.5	57.0	37.8	19.5	37.6	46.0
Mask R-CNN*	R-50-FPN	✓	3×	37.5	59.3	40.2	21.1	39.6	48.3
TensorMask Chen et al., 2019b	R-50-FPN	✓	6×	35.4	57.2	37.3	16.3	36.8	49.3
CondInst	R-50-FPN	✓	1×	35.9	56.9	38.3	19.1	38.6	46.8
CondInst	R-50-FPN	✓	3×	37.8	59.1	40.5	21.0	40.3	48.7
CondInst w/ sem.	R-50-FPN	✓	3×	38.8	60.4	41.5	21.1	41.1	51.0
Mask R-CNN	R-101-FPN	✓	6×	38.3	61.2	40.8	18.2	40.6	**54.1**
Mask R-CNN*	R-101-FPN	✓	3×	38.8	60.9	41.9	21.8	41.4	50.5
YOLACT-700	R-101-FPN	✓	4.5×	31.2	50.6	32.8	12.1	33.3	47.1
TensorMask	R-101-FPN	✓	6×	37.1	59.3	39.4	17.4	39.1	51.6
CondInst	R-101-FPN	✓	3×	39.1	60.9	42.0	21.5	41.7	50.9
CondInst w/ sem.	R-101-FPN	✓	3×	**40.1**	**62.1**	**43.1**	**21.8**	**42.7**	52.6

TABLE 5.7. **Comparisons with state-of-the-art methods on MS-COCO `test-dev`.** "Mask R-CNN" is the original Mask R-CNN He et al., 2017 and "Mask R-CNN*" is the improved Mask R-CNN in `Detectron2` Wu et al., 2019. "aug.": using multi-scale data augmentation during training. "sched.": the used learning rate schedule. 1× is $90K$ iterations. 'w/ sem": using the auxiliary semantic segmentation task.

2019b by a large margin (38.8% vs. 35.4% for ResNet-50 and 39.1% vs. 37.1% for ResNet-101). CondInst is also $\sim$ 8× faster than TensorMask (49ms vs 380ms per image on the same GPU) with similar performance (37.8% vs. 37.1%). Moreover, CondInst outperforms YOLACT-700 Bolya et al., 2019a by a large margin with the same backbone ResNet-101 (40.1% vs. 31.2% and both with the auxiliary semantic segmentation task). Moreover, as shown in Fig. 5.2, compared with YOLACT-700 and Mask R-CNN, CondInst can preserve more details and produce higher-quality instance segmentation results. More qualitative results are shown in Fig. 5.4.

5.4 Conclusions

We have proposed a new and simpler instance segmentation framework, named CondInst. Unlike previous method such as Mask R-CNN, which employs the mask head with fixed weights, CondInst conditions the mask head on instances and dynamically generates the filters of the mask head. This not only reduces the parameters and computational complexity of the mask head, but also eliminates the ROI operations, resulting in a faster and simpler instance segmentation framework. To our knowledge, CondInst is the first framework that can outperform Mask R-CNN both in accuracy and speed, without longer training schedules needed. We believe that CondInst can be a new strong alternative to Mask R-CNN for instance segmentation.

FIGURE 5.4. **More qualitative results of CondInst.** Best viewed on screen.

Chapter 6

High-Performance Instance Segmentation with Box Annotations

6.1 Introduction

Instance segmentation requires the algorithm to predict the pixel-wise masks and categories of instances of interest, and is one of the most fundamental tasks in computer vision. The performance of instance segmentation has been significantly advanced by a number of successful methods He et al., 2017; Huang et al., 2019; Wang et al., 2020d; Wang et al., 2020b; Cheng et al., 2020b; Chen et al., 2020. These methods have almost made the previously much more challenging instance segmentation task be as simple and fast as box object detection. For example, built on the detector FCOS, our CondInst only adds very compact dynamic mask heads to predict instance masks, and thus only introduces less than 10% computation overhead, compared to FCOS. Instance segmentation is able to provide more accurate and fine mask-level object location than detection. Thus, given that the extra computation cost is negligible, instance segmentation should be preferred over bounding box detection in many cases. For example, if a robot wants to grasp an object, an accurate mask will be much more helpful than a box. Now the only obstacle that impedes instance segmentation replacing box detection is the significantly heavier pixel-wise mask annotations. Compared to box-level annotations required by object detection, annotating pixel-level masks is notoriously time-consuming. As shown in Bearman et al., 2016; Everingham et al., 2010; Kulharia et al., 2020, pixel-level mask annotations are about 35 times more expensive than box-level annotations. Here we aim to eliminate this obstacle by training instance segmentation using box annotations only.

A few works Song et al., 2019; Dai, He, and Sun, 2015; Kulharia et al., 2020; Papandreou et al., 2015; Hsu et al., 2019; Khoreva et al., 2017; Rajchl et al., 2016; Arun, Jawahar, and Kumar, 2020 attempted to obtain (semantic or instance-level) mask prediction with box-level annotations. Among them, most methods such as

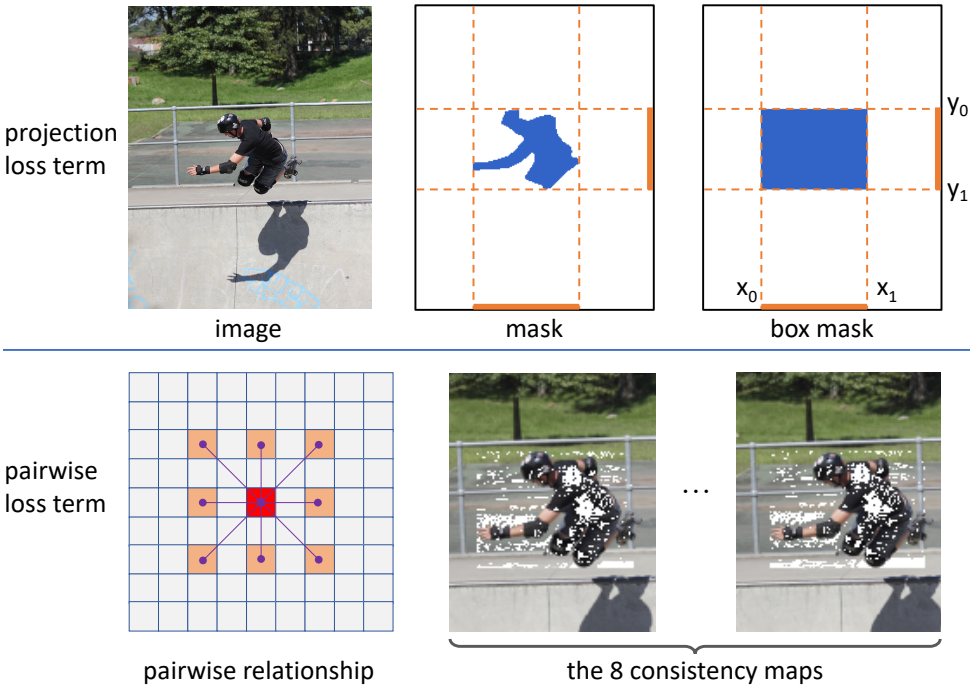

FIGURE 6.1. **The two proposed loss terms.** Top row: the projections onto x-axis and y-axis of the mask and the box, and the projections should be the same, where (x_0, y_0) and (x_1, y_1) are the two corners of the box. Bottom row: the pairwise term. For each pixel, we compute the pairwise label consistency between the pixel and its 8 neighbours (with dilation rate 2). Thus each pixel has 8 edges and we have 8 consistency maps in the right. The white locations in the right figure are the edges we have the supervision derived from the color similarity, and other edges are discarded in the loss computation.

BoxSup Dai, He, and Sun, 2015 and Box2Seg Kulharia et al., 2020 rely on the region proposals that are generated by MCG Pont-Tuset et al., 2016 or GrabCut Rother, Kolmogorov, and Blake, 2004. One drawback might be the slow training procedure since these algorithms are hard to be parallelized by modern GPUs. Moreover, in order to achieve good performance, some methods often require iterative training, resulting in a complicated training pipeline and more hyper-parameters. Most importantly, *none of these methods is able to show strong weakly-supervised performance on large benchmarks such as COCO Lin et al., 2014*. Thus almost all of them are only evaluated on small datasets such as Pascal VOC Everingham et al., 2010.

In this work, we propose a simple, single-shot and high-performance box-supervised instance segmentation method, built upon the fully convolutional instance segmentation framework CondInst. Our core idea is to replace the original pixel-wise mask losses in CondInst with a carefully designed mask loss consisting of two terms. The first term minimizes the discrepancy between the horizontal and vertical projections of the predicted mask and the ground-truth box (see Fig. 6.1 top). *This essentially ensures that the tightest box covering the predicted mask matches the ground-truth box.* Since the ground-truth mask and ground-truth box have the same projections on the

FIGURE 6.2. **Some qualitative results of BoxInst** with the ResNet-101 based model achieving 33.0% mask AP on COCO `val2017`. The model is trained without any mask annotations and can infer at 10 FPS on a 1080Ti GPU. Best viewed on screen.

two axes[1], this can be also viewed as a surrogate term that minimizes the discrepancy between the projections of the predicted mask and ground-truth mask. This loss term can be computed when we only have box annotations. Clearly, with this projection term, multiple masks can be projected to a same box. Therefore the projection loss alone would not suffice. Thus, we introduce the second loss term, encouraging the prediction and ground-truth masks have the same pairwise similarity in proximal pixels (Fig. 6.1 bottom). At first glance, the pairwise similarity of the ground-truth masks cannot be computed if we do not have the mask annotations. With only box annotations available, in principle this pairwise supervision signal is inevitably noisy. However, *an important observation is that the proximal pixels with similar colors are very likely to have the same label.* Thus, we show that it is empirically plausible to determine a color similarity threshold such that only confident pairs of pixels having a same label are used in the loss computation (the white regions in the bottom right of Fig. 6.1), thus largely eliminating supervision noises. Using these two loss terms, we achieve stunning instance segmentation results *without using any mask annotations*. Some qualitative results are shown in Fig. 6.2.

Even though ideas that are relevant to either of our two observations mentioned above were studied more or less in the literature, ranging from non-deep learning methods such as CRF Krähenbühl and Koltun, 2011 and GrabCut Rother, Kolmogorov, and Blake, 2004 to deep learning-based methods such as Box2Seg Kulharia et al., 2020 and BBTP Hsu et al., 2019, none of these works effectively incorporates them into a simple and appropriate framework. As a result, and more importantly, performance of existing methods on large challenging datasets (*e.g.*, COCO) is far away from that of the full potential of box-supervised instance segmentation that is achievable, as we are going to reveal here. In summary, our method, termed **BoxInst**, enjoys the following advantages.

[1] This may not hold if the instance mask consists multiple disjointed regions.

- The proposed method can achieve instance segmentation with box supervision by introducing two loss terms to the instance segmentation framework CondInst. BoxInst is *simple* as it does not modify the network model of CondInst at all, only using different loss terms. This means that the inference process of the proposed BoxInst is exactly the same as CondInst, thus naturally inheriting all desirable properties of CondInst.
- BoxInst attains excellent instance segmentation performance on large-scale instance segmentation benchmark COCO. With the ResNet-101 backbone and 3× training schedule, our BoxInst achieves 33.2% mask AP on COCO with no mask annotations used in training, *outperforming a few recent fully supervised methods using the same backbone and trained with mask annotations*, including YOLACT Bolya et al., 2019b (31.2% AP) and PolarMask Xie et al., 2020 (32.1% AP). mask AP of BoxInst can be further improved, as expected (§6.3.6).
- Since instance masks can provide much more precise localization than boxes, we envision that BoxInst can be used in many downstream tasks to boost their performance without extra effort of annotating ground-truth masks.

Instance segmentation has long been believed to be much more challenging to solve than bounding box detection. Our strong performance of instance segmentation using only box supervision shows that it may not necessarily be the case.

6.2 Approach

6.2.1 Projection and Pairwise Affinity Mask Loss

Projection loss term. As mentioned before, the first term supervises the horizontal and vertical projections of the predicted mask using the ground-truth box annotation, which ensures that the tightest box covering the predicted mask matches the ground-truth box. Formally, let $\boldsymbol{b} \in \{0,1\}^{H \times W}$ be the mask generated by assigning 1 to the locations in the ground-truth box and 0 otherwise, as shown in Fig. 6.1 (top-right). Then we have

$$\text{Proj}_x(\boldsymbol{b}) = \boldsymbol{l}_x, \quad \text{Proj}_y(\boldsymbol{b}) = \boldsymbol{l}_y, \tag{6.1}$$

where $\text{Proj}_x \colon \mathbb{R}^{H \times W} \to \mathbb{R}^W$ and $\text{Proj}_y \colon \mathbb{R}^{H \times W} \to \mathbb{R}^H$ indicate that projecting the mask onto x-axis and y-axis, respectively. $\boldsymbol{l}_x \in \{0,1\}^W$ denotes the 1-D segmentation mask on x-axis and the same applies to $\boldsymbol{l}_y$. The process of projection is illustrated in Fig. 6.1 (top row).

The projection operation can be implemented by a max operation along with each axis. Formally, we define

$$\text{Proj}_x(\boldsymbol{b}) = \max_y(\boldsymbol{b}) = \boldsymbol{l}_x, \quad \text{Proj}_y(\boldsymbol{b}) = \max_x(\boldsymbol{b}) = \boldsymbol{l}_y, \tag{6.2}$$

where $\max_y$ and $\max_x$ are the max operations along with y-axis and x-axis, respectively.

Let $\tilde{\boldsymbol{m}} \in (0,1)^{H \times W}$ be the network predictions for the instance mask, which can be viewed as the probabilities being foreground (*i.e.*, the label is 1). We apply the same projection operations of Eq. (6.2) to the mask predictions and obtain the corresponding projections $\tilde{\boldsymbol{l}}_x$ and $\tilde{\boldsymbol{l}}_y$. We then compute the loss between the projections of the ground-truth box and the predicted mask. Formally, the projection loss term is defined as:

$$\begin{aligned} L_{proj} &= L(\text{Proj}_x(\tilde{\boldsymbol{m}}), \text{Proj}_x(\boldsymbol{b})) + L(\text{Proj}_y(\tilde{\boldsymbol{m}}), \text{Proj}_y(\boldsymbol{b})) \\ &= L(\max_y(\tilde{\boldsymbol{m}}), \max_y(\boldsymbol{b})) + L(\max_x(\tilde{\boldsymbol{m}}), \max_x(\boldsymbol{b})) \\ &= L(\tilde{\boldsymbol{l}}_x, \boldsymbol{l}_x) + L(\tilde{\boldsymbol{l}}_y, \boldsymbol{l}_y), \end{aligned} \tag{6.3}$$

where $L(\cdot, \cdot)$ is the Dice loss as in CondInst[2]. Note that *all the operations in the last equation are (sub-)differentiable*. This loss function is applied to all the instances in a training image and the final loss is their average. As shown in our experiments, by using this projection loss term, we can already obtain decent instance segmentation results without using any mask annotations, which can already provide much better localization quality than a box detector.

Pairwise affinity loss term. In almost all instance segmentation frameworks such as Mask R-CNN and CondInst, they supervise the predicted masks in a per-pixel fashion. The pixelwise supervision becomes unavailable if we do not have the mask annotations. Here, we attempt to supervise the mask in a *pairwise* way, and we will show this supervision can be *partially available* even if we do not have any mask annotations.

Now, assume we have the ground-truth masks. Consider an undirected graph $G = (V, E)$ built on an image, where V is the set of the pixels in the image, and E is the set of the edges. Each pixel is connected with its $K \times K - 1$ neighbours (the dilation trick may be applied), as shown in Fig. 6.1 (bottom left). Then we define $y_e \in \{0, 1\}$ be the label for an edge e, where $y_e = 1$ means the two pixels of the edge have the same label and $y_e = 0$ means their labels are different. Let pixels (i, j) and (l, k) be the two endpoints of the edge e. The network prediction $\tilde{\boldsymbol{m}}_{i,j} \in (0, 1)$ can be viewed as the probability of pixel (i, j) being foreground. Then the probability of $y_e = 1$ is

$$P(y_e = 1) = \tilde{\boldsymbol{m}}_{i,j} \cdot \tilde{\boldsymbol{m}}_{k,l} + (1 - \tilde{\boldsymbol{m}}_{i,j}) \cdot (1 - \tilde{\boldsymbol{m}}_{k,l}), \tag{6.4}$$

and $P(y_e = 0) = 1 - P(y_e = 1)$. By convention, the probability distribution from the network prediction can be trained with the binary cross entropy (BCE) loss. Formally,

[2] One can also use the cross-entropy loss here.

the loss function is

$$L_{pairwise} = -\frac{1}{N}\sum_{e\in E_{in}} y_e \log P(y_e = 1) + (1 - y_e) \log P(y_e = 0), \tag{6.5}$$

where E_{in} is the set of the edges containing at least one pixel in the box. Using E_{in} instead of E here can prevent the loss from being dominated by a large number of the pixels outside the box. N is the number of the edges in E_{in}.

If only the pairwise loss is used to supervise the mask learning (in the fully-supervised setting), ideally, two possible solutions may be obtained. The first one is the same as the ground-truth mask $\boldsymbol{m}$, which is desirable. The second solution is the inverse $1-\boldsymbol{m}$. Fortunately, the second solution can be easily eliminated as long as we have a resolved label for any pixel. This can be achieved by the projection loss term because it ensures that the pixels outside the box is background. Note that the edges in E_{in} still involve some pixels outside the box, which are of great importance to help the model get rid of the undesirable solutions. Overall, the total loss for mask learning can be formulated as

$$L_{mask} = L_{proj} + L_{pairwise}. \tag{6.6}$$

We will show in experiments that the redesigned mask loss can have similar performance to the original pixelwise one in the fully-supervised settings.

6.2.2 Learning without Mask Annotations

So far, we have shown that we can employ Eq. (6.6) to supervise the masks. In Eq. (6.6), the first term L_{proj} is always valid no matter we have box or mask annotations. At first glance, the second term $L_{pairwise}$ still requires the mask annotations to compute the edge's label y_e. However, an important observation is that if two pixels have similar colors, they are very likely to have the same labels as well (*i.e.*, the corresponding edge's label is 1). Thus, we may determine a color similarity threshold τ such that the edge's label is 1 with a high probability if its color similarity is above τ. Formally, let us define the color similarity as

$$S_e = S(\boldsymbol{c}_{i,j}, \boldsymbol{c}_{l,k}) = \exp\left(-\frac{||\boldsymbol{c}_{i,j} - \boldsymbol{c}_{l,k}||}{\theta}\right), \tag{6.7}$$

where S_e be the color similarity of the edge e, and $\boldsymbol{c}_{i,j}$ and $\boldsymbol{c}_{l,k}$ are, respectively, the color vectors of the two pixels (i, j) and (l, k) linked by the edge. Here we use the LAB color space as it is closer to human perception. θ is a hyper-parameter, being 2 in this work.

In order to confirm our hypothesis above, we visualize the proportion of the positive edges in all the edges with color similarity above the threshold τ on the COCO `val2017` split. Fig. 6.3 (blue curve) shows that the proportions of the positive edges in all the edges with $S_e \geq \tau$ as the threshold τ increases. As shown in figure, if the

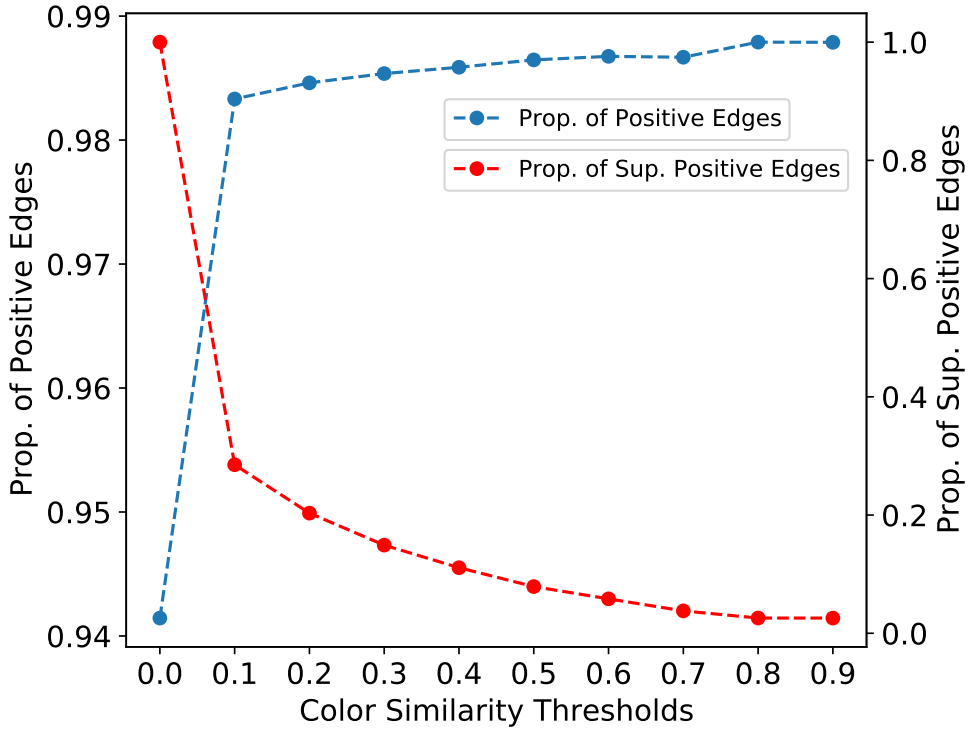

FIGURE 6.3. **The relationship between the edges' labels and the color similarity thresholds.** 'blue curve': the proportion of the positive edges in the edges with color similarity above the threshold. 'red curve': the proportion of the supervised positive edges in all the positive edges. The number of positive edges are computed with the ground-truth masks of the COCO `val2017` split.

threshold is 0.1, more than 98% of the edges are positive. The proportion can be further improved if we continue to increase τ, but an overlarge threshold would reduce the number of the supervised edges (red curve in Fig. 6.3). In experiments, we found that the threshold is not sensitive to the final performance.

Given the high proportion of positive edges, during training, we can safely assume that all the edges with $S_e \geq \tau$ are positive and then only compute the pairwise loss for them. Other edges are discarded during the loss computation. As a result, the pairwise loss becomes

$$L_{pairwise} = -\frac{1}{N} \sum_{e \in E_{in}} \mathbb{1}_{\{S_e \geq \tau\}} \log P(y_e = 1), \tag{6.8}$$

where $\mathbb{1}_{\{S_e \geq \tau\}}$ is the indicator function, being 1 if $S_e \geq \tau$ and 0 otherwise. Eq. (6.8) only involves the term in Eq. (6.5) for positive edges because we can only infer that an edge e is positive if $S_e \geq \tau$. If $S_e < \tau$, the label is agnostic. As we only have positive labels in Eq. (6.8), one may note this would result in two possible trivial solutions, *i.e.*, the masks of all the pixels being 0 or 1. However, the masks with all pixels being 0 do not meet the projection term; and the masks of all pixels being 1 almost never appear since the pairwise term encourages the pixels near the box boundaries to be negative if their colors are similar to that of the negative pixels outside the box.

mask loss	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Dice loss	35.6	56.3	37.8	16.9	38.9	51.0
proposed	35.4	55.9	37.6	17.0	38.8	50.7

TABLE 6.1. **The projection and pairwise affinity mask loss vs. the original pixelwise one** in the fully-supervised settings. As we can see here, they attain very similar mask AP on the COCO split `val2017`.

6.3 Experiments

We conduct experiments on COCO Lin et al., 2014 and Pascal VOC Everingham et al., 2010. For COCO, the models are trained with `train2017` (115K images) and the ablation experiments are evaluated on `val2017` (5K images). Unless specified, *only box annotations* are used during training. Our main results are reported on `test-dev`. For Pascal VOC, following previous works Hsu et al., 2019; Khoreva et al., 2017, we train the models on the augmented Pascal VOC 2012 dataset Hariharan et al., 2011 with 10, 582 training images, and evaluate them on Pascal VOC 2012 `val` split with 1, 449 images.

6.3.1 Implementation Details

On the COCO dataset, BoxInst has the same training and testing details as that of CondInst. The only exception is that we increase the channels of the mask heads from 8 to 16, which can result in better performance with negligible extra computational overhead, and the compared baselines are adjusted accordingly. For the pairwise loss term, we compute the pairwise relationship within 3×3 patches with the dilation rate being 2. On Pascal VOC, following Hsu et al., 2019, we use batch size 8 and the number of iterations is 20K. The learning rate is reduced by a factor of 10 at step 15K. Only left-right flipping is used as the data augmentation during training. Other settings are the same as on COCO. The inference is the same as the original CondInst on both benchmarks. The performance is evaluated with the COCO-style mask AP.

6.3.2 Projection and Pairwise Affinity Loss for Mask Learning

We first demonstrate that the redesigned mask loss can have similar performance to the original pixelwise mask loss in the fully-supervised settings. The experiments are conducted on COCO. We replace the original Dice loss for mask training in CondInst with the proposed one, and keep other settings exactly the same. As shown in Table 6.1, the proposed mask loss can have similar performance (35.4% vs. 35.6% mask AP), which suggests that using the proposed loss for mask learning is feasible.

6.3.3 Box-supervised Instance Segmentation

The key advantage of the proposed mask loss is that it can still supervise the predicted masks with only box annotations. We confirm this here and conduct experiments to

	prop.	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
fully-sup.	-	35.4	55.9	37.6	17.0	38.8	50.7
$\tau = 0$	94.1%	9.4	30.3	3.3	7.6	10.3	11.4
$\tau = 0.1$	98.3%	**30.7**	52.2	**31.1**	13.8	**33.1**	**45.7**
$\tau = 0.2$	**98.4%**	30.6	**52.6**	30.9	**13.9**	32.8	45.5

TABLE 6.2. **Varying the color similarity threshold** τ in the proposed mask loss on the COCO `val2017` split. "prop." is the proportion of the positive edges in the edges with $S_e \geq \tau$. "fully-sup.": fully-supervised results.

size	dilation	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
3	1	29.7	52.0	29.6	13.4	32.3	44.4
3	2	**30.7**	52.2	**31.1**	**13.8**	**33.1**	**45.7**
5	1	30.5	**52.3**	30.7	13.7	33.0	**45.7**
5	2	29.9	51.9	30.0	13.8	32.1	45.0

TABLE 6.3. **Varying the size and dilation of the local patches (with** $\tau = 0.1$**)** in the proposed mask loss on the COCO `val2017` split. "prop." is the proportion of the positive edges in the edges with $S_e \geq \tau$. "fully-sup.": fully-supervised results. As shown in Table 6.2, by using $\tau = 0.1$, BoxInst can achieve 30.7 mask AP with only box annotations, which is close the fully-supervised mask AP (35.4%) and significantly better localization precision than boxes (10.6% mask AP as shown in Table 6.4).

L_{proj}	$L_{pairwise}$	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
box mask		10.6	32.2	4.6	5.7	11.3	15.6
✓		21.2	45.2	17.7	10.0	21.4	32.5
✓	✓	**30.7**	**52.2**	**31.1**	**13.8**	**33.1**	**45.7**

TABLE 6.4. **The mask AP on COCO** `val2017` by applying the different loss terms. "box mask": using the masks generated by boxes.

investigate the hyper-parameters in the proposed mask loss.

Varying the threshold of color similarity. As mentioned before, we use a color similarity threshold τ to determine the edges that will be used to compute the pairwise loss. Here, we conduct experiments by varying τ. When $\tau = 0$, all of the edges defined by the size of neighborhood are used to compute the loss. As shown in Table 6.2, in this case, 94.1% of the edges are truly positive, and ∼6% of the edges are negative. Since we consider positive all the edges with $S_e \geq \tau$, as shown in Eq. (6.8), the loss computation would introduce ∼6% noisy labels and all of the truly negative edges are wrongly labelled positive. Thus, unsurprisingly, this experiment yields a trivial solution with poor performance (9.4% mask AP) that almost all pixels in the box are predicted as foreground. If we increase τ to 0.1, the proportion of the truly positive edges are improved to 98.3%, and only less than ∼2% of the edges are wrongly labelled. As a result, the model can yield high-quality instance masks, achieving 30.7% mask AP (vs. fully-supervised counterpart 35.4%). This result is even better than that of some fully-supervised methods such as YOLACT and PolarMask. Some qualitative results are shown in Fig. 6.2. If we further increase the threshold τ to 0.2, the performance

method	backbone	aug.	sched.	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
fully supervised methods:									
Mask R-CNN	ResNet-50-FPN	✓	3×	37.5	59.3	40.2	21.1	39.6	48.3
CondInst	ResNet-50-FPN	✓	3×	37.8	59.1	40.5	21.0	40.3	48.7
Mask R-CNN	ResNet-101-FPN	✓	3×	38.8	60.9	41.9	21.8	41.4	50.5
YOLACT-700	ResNet-101-FPN	✓	4.5×	31.2	50.6	32.8	12.1	33.3	47.1
PolarMask	ResNet-101-FPN	✓	2×	32.1	53.7	33.1	14.7	33.8	45.3
CondInst	ResNet-101-FPN	✓	3×	39.1	60.9	42.0	21.5	41.7	50.9
box-supervised methods:									
BBTP† (prev. best)	ResNet-101-FPN		1×	21.1	45.5	17.2	11.2	22.0	29.8
BoxInst †	ResNet-101-FPN		1×	31.6	54.0	31.9	13.9	34.2	**48.2**
BoxInst	ResNet-50-FPN	✓	3×	32.1	55.1	32.4	15.6	34.3	43.5
BoxInst	ResNet-101-FPN	✓	1×	32.5	55.3	33.0	15.6	35.1	44.1
BoxInst	ResNet-101-FPN	✓	3×	33.2	56.5	33.6	16.2	35.3	45.1
BoxInst	ResNet-101-BiFPN	✓	3×	**33.9**	**57.7**	**34.5**	**16.5**	**36.1**	46.6

TABLE 6.5. **Comparisons with state-of-the-art methods** on the COCO `test-dev` split. "†" means that the results are on the COCO `val2017` split. BBTP only reported the results on the `val2017` split. Our BoxInst outperforms the previous best reported mask AP by over absolute 10% mask AP. Ours even outperforms two recent fully supervised methods, YOLACT and PolarMask, and is close to state-of-the-art fully-supervised results. '1×' means 90K iterations.

slightly drops to 30.6% mask AP. This might be because the number of the supervised positive edges decreases as we increase the threshold, as shown in Fig. 6.3.

Varying the neighborhood of the pixels. We conduct experiments with the different neighbours for each pixel. The size (*i.e.*, K) defines how many surrounding pixels of each pixel are used to compute the pairwise loss with the pixel. Additionally, we may use the dilation trick to enlarge the scope (as in dilated convolutions). As shown in Table 6.3, by increasing the size from 3×3 to 5×5, the performance is boosted from 29.7% to 30.5%. This suggests that a relatively long-distance pairwise relationship is important to the final performance. However, using 5×5 makes the training relatively slower and costs more memory footprint. Thus, we apply the dilation rate 2 to the 3×3 patches. This can capture the long-distance relationship without increasing the computational overheads, thus achieving a similar performance (30.7%). The performance cannot be further improved by applying the dilation trick to the 5×5 patches because the assumption, two pixels with similar colors probably have the same label, might not hold if the two pixels are far from each other.

The contribution of each loss term. Table 6.4 shows the contribution of each loss term. Even if only the first projection term is used, we can also achieve decent performance (21.2% mask AP), which can already provide much higher localization precision than boxes (10.6% mask AP). By further using the proposed pairwise term, high-quality instance masks can be obtained and the performance is much improved to 30.7%.

method	backbone	AP	AP_{50}	AP_{75}
GrabCut Rother, Kolmogorov, and Blake, 2004	ResNet-101	17.8	37.8	15.5
SDI Khoreva et al., 2017	VGG-16	-	44.8	16.3
BBTP Hsu et al., 2019	ResNet-101	23.1	54.1	17.1
BBTP w/ CRF	ResNet-101	27.5	59.1	21.9
BBTP*	ResNet-101	20.5	51.1	14.3
BBTP* w/ CRF	ResNet-101	25.0	56.9	18.9
BoxInst	ResNet-50	32.2	58.1	31.0
BoxInst	ResNet-101	**34.4**	**60.1**	**34.6**

TABLE 6.6. **Results on Pascal VOC** `val2012`. Here, BBTP* denotes the results after we fix the issue 2020 in its Matlab evaluation code. Clearly, BoxInst achieves significantly improved mask AP, outperforming previous best by about 10%. Here, the GrabCut obtains the instance masks by taking as input the boxes generated by BoxInst. Thus, the only difference between the GrabCut and BoxInst is the way to obtain the masks.

6.3.4 Comparisons with State-of-the-art

We compare BoxInst with state-of-the-art fully/box supervised instance segmentation methods on the COCO dataset. As shown in Table 6.5, with the same backbone and training settings, BoxInst significantly surpasses the previous best reported result Hsu et al., 2019 by absolute 10.5% mask AP (*e.g.*, from 21.1% to 31.6%). *BoxInst, without using any mask annotations, performs even better than some recent fully-supervised methods such as PolarMask Xie et al., 2020 and YOLACT Bolya et al., 2019b*, with the same backbones and similar training and testing settings (32.5% with R-101 1× vs. PolarMask 32.1% R-101 2× and YOLACT-700 31.2% R-101 4.5×). BoxInst also demonstrates competitive performance with top-performing fully-supervised instance segmentation methods. For example, with the same backbone ResNet-50-FPN and 3× training schedule, BoxInst achieves 32.1% mask AP (vs. 37.8% of the fully-supervised CondInst). Notably, Some qualitative results are shown in Fig. 6.2. The excellent performance shows that BoxInst dramatically narrows the performance gap between the fully supervised and box-supervised instance segmentation, and for the first time, the great potential of box-supervised instance segmentation is revealed.

6.3.5 Experiments on Pascal VOC

We also conduct experiments on Pascal VOC. As shown in Table 6.6, BoxInst achieves state-of-the-art instance segmentation with only box annotations. With the same backbone and training settings, BoxInst outperforms BBTP both in AP_{50} and AP_{75} by a large margin. Notably, the AP_{75} is improved by more than relative 200% (14.3% vs. 34.6% mask AP), which suggests BoxInst can produce the masks of much higher quality. BoxInst is even much better than the BBTP with CRF. Additionally, BoxInst also performs much better than SDI Khoreva et al., 2017. We also compare BoxInst with the traditional unsupervised segmentation method GrabCut Rother, Kolmogorov, and Blake, 2004. In the experiment, GrabCut takes as input the bounding-boxes predicted by the ResNet-101 based FCOS in BoxInst. Thus the only difference between BoxInst and GrabCut is the way of obtaining instance masks. As shown in Table 6.6, BoxInst

L_{proj}	$L_{pairwise}$	all 80 classes			60 unseen classes		
		AP	AP_{50}	AP_{75}	AP	AP_{50}	AP_{75}
		24.7	44.6	24.2	19.9	38.3	18.5
✓		31.8	52.5	33.2	29.7	49.3	31.0
✓	✓	**32.5**	**53.0**	**34.0**	**30.9**	**50.1**	**32.4**
box supervised		30.7	52.2	31.1	29.6	49.7	30.4

TABLE 6.7. **BoxInst for semi-supervised instance segmentation**. These models are trained with the 20 classes mask annotations and the other 60 classes (*i.e.*, unseen classes) are only with box annotations.

L_{proj}	$L_{pairwise}$	all 80 classes			20 unseen classes		
		AP	AP_{50}	AP_{75}	AP	AP_{50}	AP_{75}
		32.1	51.6	33.9	25.5	45.5	25.1
✓		33.1	53.8	34.3	31.6	57.4	30.0
✓	✓	**33.8**	**54.3**	**35.7**	**35.9**	**60.9**	**36.3**
box supervised		30.7	52.2	31.1	29.6	49.7	30.4

TABLE 6.8. **BoxInst for semi-supervised instance segmentation**. The models are trained with the 60 classes mask annotations and other 20 classes (*i.e.*, unseen classes) are only with box annotations.

is far better than GrabCut (17.8% vs. 34.4% mask AP). Moreover, BoxInst is fully convolutional and can benefit from the highly-efficient GPUs, thus inferring tens of times faster than GrabCut.

6.3.6 Extensions: Semi-supervised Instance Segmentation

In this section, we show that our method can also help the model generalize to unseen categories in the semi-supervised setting where only partial classes have the mask annotations. Following previous works Zhou et al., 2020; Hu et al., 2018; Kuo et al., 2019 in this setting, we conduct the experiments on the COCO dataset and split the 80 classes in COCO into two groups – 20 classes present in Pascal VOC and 60 classes not in Pascal VOC. Then the models are trained with the mask annotations of one group of classes, and another group of classes only have the box annotations. The generalization ability is evaluated with the mask AP averaged over the group of classes without mask annotations (*i.e.*, unseen classes).

We first train the model with the 20 classes mask annotations. As shown in Table 6.7 (1st row), if our proposed loss terms are not used, where the mask loss is only computed for the instances with mask annotations and other instances are discarded during the mask learning, the model can only achieve 25.5% mask AP on the unseen categories. This low performance suggests that the model is difficult to generalize to unseen classes. If we use the L_{proj} term for the 60 classes without the mask annotations during training, as shown in the table (2nd row), the performance can be dramatically improved to 29.7%. If we further apply the pairwise term $L_{pairwise}$, the performance can be boosted to 30.9%. Moreover, compared to the setting only using the box

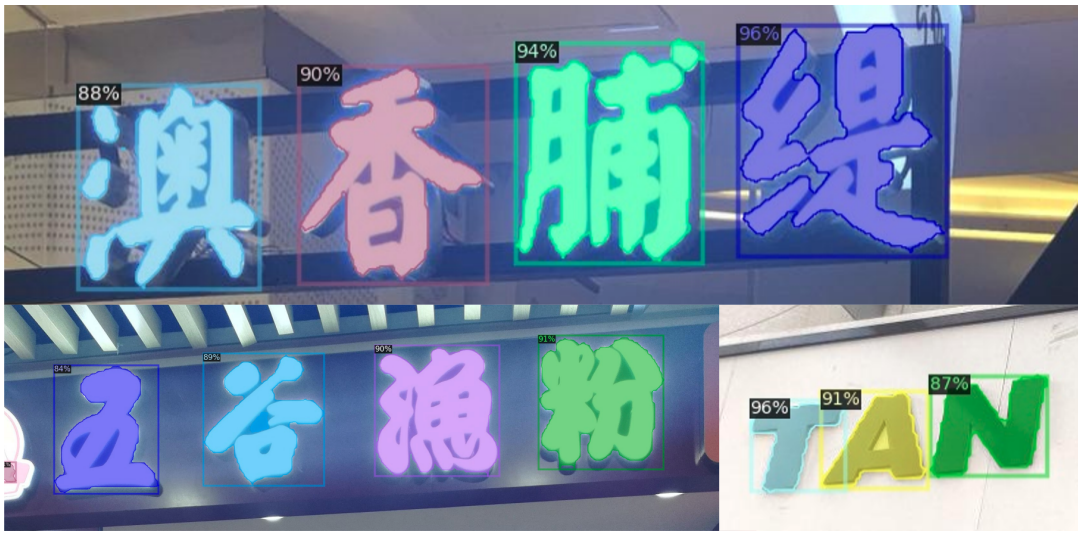

FIGURE 6.4. Character masks predicted by BoxInst. No mask annotations are used for training.

annotations (last row in Table 6.7), the performance on the unseen classes is also improved from 29.6% to 30.9%, which suggests that the totally box-supervised model can also benefit from the partial mask annotations. Additionally, the experimental results with the 60 classes masks are shown in Table 6.8, and the same conclusions can be drawn.

6.3.7 Extensions: Box-supervised Character Segmentation

In order to demonstrate the generality of BoxInst, we conduct experiments to obtain the character masks with character box annotations. Our experiments are conducted on the ICDAR 2019 ReCTS dataset Zhang et al., 2019, which contains 20K training images and 5K testing images and these images are annotated with text-line and character-level boxes. We train our model with the character boxes. All the training settings are the same as that of COCO. Since we do not have mask annotations for the testing set, it is impossible to report the mask AP. We instead show some qualitative results in Fig. 6.4, demonstrating that BoxInst can obtain high-quality character masks. It is well known that the text masks provide useful cues for detecting and recognising text of arbitrary shapes Lyu et al., 2018; Liu et al., 2020d. We believe that the ability of BoxInst generating character masks automatically may inspire new applications on this task.

6.4 Conclusions

In this work, we have proposed BoxInst that can achieve high-quality instance segmentation with only box annotations. The core idea of BoxInst is to replace the original pixelwise mask loss with the proposed projection and pairwise affinity mask

loss. With the proposed mask loss, we show excellent instance segmentation performance without using any mask annotations on COCO and Pascal VOC, significantly improving the state-of-the-art.

Chapter 7

Conclusions

In this thesis, we have proposed a series of novel methods for instance-level visual recognition with fully convolutional networks. These proposed methods are simple yet effective, as evidenced by the excellent performance on some challenging benchmarks.

First, we present the anchor-free detector FCOS. Without using anchor boxes, FCOS works by predicting the distances from the locations of the feature maps to the four boundaries of the boxes. We empirically show that the much simpler solution can work reliably in many challenging cases. We also conducted many analyses about the anchor boxes in this thesis. These analyses surprisingly show that the long-standing anchor boxes might not be as important as we thought. Our anchor-free detector can have almost the same recall rates and even better performance. Moreover, due to the elimination of the anchor boxes, the detector becomes very simple and flexible, and it can be easily extended to downstream computer vision tasks such as keypoint detection and instance segmentation. To our knowledge, FCOS is the first anchor-free detector that demonstrates improved object detection performance on challenging benchmarks such as MS-COCO.

Inspired by FCOS, we propose DirectPose to extend FCOS to the keypoint detection task. Unlike previous keypoint detection methods, which rely on ROI operations or grouping processing, DirectPose directly regresses all the instance-agnostic keypoints with the fully convolutional networks, and thus it is able to directly map a raw input image to the final keypoint results, making the task much simpler. It can also unify object detection and keypoint detection in a single simple and elegant framework. However, the naive DirectPose framework has difficulty in localizing precise keypoints since it relies on a single feature vector to encode all the keypoints of an instance as well as the convolutional features and the predictions are misaligned. In DirectPose, we address the issues with the proposed KPAlign module. The KPAlign module is able to align the convolutional features and the target keypoints and avoids that a single feature vector encodes all the keypoints of an instance, thus improving the final performance by a large margin.

Moreover, we also propose CondInst for instance segmentation. Previous top-performing instance segmentation methods rely on ROI operations. The ROI operations are the

core operation making the model attend to instances. However, the ROI operations come with some drawbacks. In CondInst, for the first time, CondInst proposes to employ the mask heads based on dynamic filters (or conditional convolutions) to attend to the instances. This is made possible because the weights of the dynamic filters are dynamically-generated in inference and conditioned on the instance to be predicted. As a result, the mask head based on the dynamic filter can distinguish the instance mask from the masks of other instances. In that way, CondInst eliminates the ROI operations in previous methods and solves instance segmentation in the fully convolutional fashion. Moreover, since the mask head of CondInst is dynamic and is asked to predict the mask only for one instance, the mask head can be very compact (*i.e.*, only hundreds of weights). Thus, the mask head only takes negligible inference time per instance. This makes CondInst infer faster and have almost constant inference time regardless of the number of the instances in the image. CondInst also yields the instance masks of much higher quality. For example, we improve the mask's resolution of Mask R-CNN by tens of times.

Finally, to avoid the cost of the mask annotations in instance segmentation, built on CondInst, we propose BoxInst to solve instance segmentation with only box annotations. The core idea of BoxInst is to redesign the mask loss in CondInst as the original pixelwise mask loss cannot supervise the mask learning if the mask annotations are not available. The redesigned mask loss first ensures that the tightest box covering the predicted mask of an instance matches the ground-truth box of the instance. Second, the redesigned mask loss encourages that the predicted mask of an instance has the same pairwise label similarity as the ground-truth mask of the instance. The pairwise label similarity can be partially derived by exploiting the prior that the proximal pixels with similar colors in an image are very likely to have the same category label. Thus, the resigned mask loss can still be partially supervised even if the mask annotations are not available. Our experiments show that BoxInst can yield high-quality instance masks with only box-level supervision, largely closing the gap between fully-supervised and box-supervised instance segmentation. It has been long believed that obtaining the instance masks is a very challenging task. Our strong performance without using any mask annotations shows that it may not necessarily be the case.

We believe that the proposed methods in this thesis can substantially change the status quo of the instance-level visual recognition, and we have seen many works built upon our methods so far. We hope that our proposed methods can lay a solid foundation for many tasks and applications that require instance-level recognition and bring some new insights to the entire computer vision community.

Bibliography

(2019). https://github.com/yqyao/FCOS_PLUS.

(2020). https://github.com/chengchunhsu/WSIS_BBTP/issues/11.

A. Paszke et al. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Proc. Advances in Neural Inf. Process. Syst.* Pp. 8024–8035.

Arun, Aditya, C. V. Jawahar, and Pawan Kumar (2020). "Weakly Supervised Instance Segmentation by Learning Annotation Consistent Instances". In: *Proc. Eur. Conf. Comp. Vis.*

Bearman, Amy, Olga Russakovsky, Vittorio Ferrari, and Li Fei-Fei (2016). "What's the point: Semantic segmentation with point supervision". In: *Proc. Eur. Conf. Comp. Vis.* Pp. 549–565.

Bian, Jia-Wang, Huangying Zhan, Naiyan Wang, Tat-Jun Chin, Chunhua Shen, and Ian Reid (2020). "Unsupervised Depth Learning in Challenging Indoor Video: Weak Rectification to Rescue". In: *arXiv preprint arXiv:2006.02708*.

Bian, Jiawang, Zhichao Li, Naiyan Wang, Huangying Zhan, Chunhua Shen, Ming-Ming Cheng, and Ian Reid (2019). "Unsupervised scale-consistent depth and ego-motion learning from monocular video". In: *Advances in neural information processing systems*, pp. 35–45.

Bolya, Daniel, Chong Zhou, Fanyi Xiao, and Yong Jae Lee (2019a). "YOLACT: real-time instance segmentation". In: *Proc. IEEE Int. Conf. Comp. Vis.* Pp. 9157–9166.

Bolya, Daniel, Chong Zhou, Fanyi Xiao, and Yong Jae Lee (2019b). "YOLACT: Real-time Instance Segmentation". In: *Proc. IEEE Int. Conf. Comp. Vis.*

Boominathan, Lokesh, Srinivas SS Kruthiventi, and R Venkatesh Babu (2016). "Crowdnet: A deep convolutional network for dense crowd counting". In: *Proc. ACM Int. Conf. Multimedia*. ACM, pp. 640–644.

Cao, Zhe, Tomas Simon, Shih-En Wei, and Yaser Sheikh (2017). "Realtime multi-person 2d pose estimation using part affinity fields". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7291–7299.

Chen, Hao, Kunyang Sun, Zhi Tian, Chunhua Shen, Yongming Huang, and Youliang Yan (2020). "BlendMask: Top-down meets bottom-up for instance segmentation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Pp. 8573–8581.

Chen, Kai, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. (2019a). "Hybrid task cascade for instance segmentation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Pp. 4974–4983.

Chen, Liang-Chieh, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan Yuille (2017a). "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 40.4, pp. 834–848.

Chen, Xinlei, Ross Girshick, Kaiming He, and Piotr Dollár (2019b). "Tensormask: A foundation for dense object segmentation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Pp. 2061–2069.

Chen, Yilun, Zhicheng Wang, Yuxiang Peng, Zhiqiang Zhang, Gang Yu, and Jian Sun (2018). "Cascaded pyramid network for multi-person pose estimation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7103–7112.

Chen, Yu, Chunhua Shen, Xiu-Shen Wei, Lingqiao Liu, and Jian Yang (2017b). "Adversarial PoseNet: A Structure-aware Convolutional Network for Human Pose Estimation". In: *Proc. IEEE Int. Conf. Comp. Vis.*

Cheng, Bowen, Bin Xiao, Jingdong Wang, Honghui Shi, Thomas S Huang, and Lei Zhang (2020a). "Higherhrnet: Scale-aware representation learning for bottom-up human pose estimation". In: pp. 5386–5395.

Cheng, Tianheng, Xinggang Wang, Lichao Huang, and Wenyu Liu (2020b). "Boundary-preserving Mask R-CNN". In: *Proc. Eur. Conf. Comp. Vis.*

Chu, Xuangeng, Anlin Zheng, Xiangyu Zhang, and Jian Sun (2020). "Detection in Crowded Scenes: One Proposal, Multiple Predictions". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Dai, Jifeng, Kaiming He, Yi Li, Shaoqing Ren, and Jian Sun (2016). "Instance-sensitive fully convolutional networks". In: *Proc. Eur. Conf. Comp. Vis.* Springer, pp. 534–549.

Dai, Jifeng, Kaiming He, and Jian Sun (2015). "Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation". In: *Proc. IEEE Int. Conf. Comp. Vis.* Pp. 1635–1643.

Dai, Jifeng, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei (2017). "Deformable convolutional networks". In: *Proc. IEEE Int. Conf. Comp. Vis.* Pp. 764–773.

Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei (2009). "ImageNet: A large-scale hierarchical image database". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* IEEE, pp. 248–255.

Dollár, Piotr, Ron Appel, Serge Belongie, and Pietro Perona (2014). "Fast Feature Pyramids for Object Detection". In: *IEEE Trans. Pattern Anal. Mach. Intell.*

Dollar, Piotr, Christian Wojek, Bernt Schiele, and Pietro Perona (2011). "Pedestrian detection: An evaluation of the state of the art". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 34.4, pp. 743–761.

Duan, Kaiwen, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian (2019). "CenterNet: Keypoint triplets for object detection". In: *Proc. IEEE Int. Conf. Comp. Vis.* Pp. 6569–6578.

Duan, Kaiwen, Lingxi Xie, Honggang Qi, Song Bai, Qingming Huang, and Qi Tian (2020). "Corner Proposal Network for Anchor-free, Two-stage Object Detection". In: *Proc. Eur. Conf. Comp. Vis.*

Everingham, Mark, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman (2010). "The Pascal visual object classes (VOC) challenge". In: *Int. J. Comput. Vision* 88.2, pp. 303–338.

Fang, Hao-Shu, Shuqin Xie, Yu-Wing Tai, and Cewu Lu (2017). "Rmpe: Regional multi-person pose estimation". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2334–2343.

Fathi, Alireza, Zbigniew Wojna, Vivek Rathod, Peng Wang, Hyun Oh Song, Sergio Guadarrama, and Kevin P Murphy (2017). "Semantic instance segmentation via deep metric learning". In: *arXiv: Comp. Res. Repository*.

Fu, Cheng-Yang, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander Berg (2017). "DSSD: Deconvolutional single shot detector". In: *arXiv preprint arXiv:1701.06659*.

Girshick, Ross (2015). "Fast R-CNN". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Pp. 1440–1448.

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

Girshick, Ross, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He (2018). *Detectron*. https://github.com/facebookresearch/detectron.

Gkioxari, Georgia, Bharath Hariharan, Ross Girshick, and Jitendra Malik (2014). "Using k-poselets for detecting people and localizing their keypoints". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3582–3589.

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

Guo, Dongyan, Jun Wang, Ying Cui, Zhenhua Wang, and Shengyong Chen (2020). "SiamCAR: Siamese Fully Convolutional Classification and Regression for Visual Tracking". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Hariharan, Bharath, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik (2011). "Semantic contours from inverse detectors". In: *Proc. IEEE Int. Conf. Comp. Vis.* Pp. 991–998.

He, Kaiming, Georgia Gkioxari, Piotr Dollár, and Ross Girshick (2017). "Mask r-cnn". In: *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2015). "Spatial pyramid pooling in deep convolutional networks for visual recognition". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 37.9, pp. 1904–1916.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep residual learning for image recognition". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Pp. 770–778.

He, Tong, Chunhua Shen, Zhi Tian, Dong Gong, Changming Sun, and Youliang Yan (2019a). "Knowledge Adaptation for Efficient Semantic Segmentation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

He, Tong, Chunhua Shen, Zhi Tian, Dong Gong, Changming Sun, and Youliang Yan (2019b). "Knowledge adaptation for efficient semantic segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 578–587.

He, Tong, Zhi Tian, Weilin Huang, Chunhua Shen, Yu Qiao, and Changming Sun (2018). "An end-to-end textspotter with explicit alignment and attention". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Pp. 5020–5029.

Hsu, Cheng-Chun, Kuang-Jui Hsu, Chung-Chi Tsai, Yen-Yu Lin, and Yung-Yu Chuang (2019). "Weakly supervised instance segmentation using the bounding box tightness prior". In: *Proc. Advances in Neural Inf. Process. Syst.* Note that the mask AP results on COCO are in the supplementary, available at https://tinyurl.com/yyjovxn6.

Hu, Ronghang, Piotr Dollár, Kaiming He, Trevor Darrell, and Ross Girshick (2018). "Learning to segment every thing". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Pp. 4233–4241.

Huang, Lichao, Yi Yang, Yafeng Deng, and Yinan Yu (2015). "Densebox: Unifying landmark localization with end to end object detection". In: *arXiv: Comp. Res. Repository* abs/1509.04874.

Huang, Shaoli, Mingming Gong, and Dacheng Tao (2017). "A coarse-fine network for keypoint localization". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3028–3037.

Huang, Zhaojin, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang (2019). "Mask scoring R-CNN". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Pp. 6409–6418.

Insafutdinov, Eldar, Leonid Pishchulin, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele (2016). "Deepercut: A deeper, stronger, and faster multi-person pose estimation model". In: *European Conference on Computer Vision*. Springer, pp. 34–50.

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

Jia, Xu, Bert De Brabandere, Tinne Tuytelaars, and Luc Van Gool (2016). "Dynamic filter networks". In: *Proc. Advances in Neural Inf. Process. Syst.* Pp. 667–675.

Khoreva, Anna, Rodrigo Benenson, Jan Hosang, Matthias Hein, and Bernt Schiele (2017). "Simple does it: Weakly supervised instance and semantic segmentation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Pp. 876–885.

Krähenbühl, Philipp and Vladlen Koltun (2011). "Efficient inference in fully connected CRFs with Gaussian edge potentials". In: *Proc. Advances in Neural Inf. Process. Syst.* Pp. 109–117.

Kulharia, Viveka, Siddhartha Chandra, Amit Agrawal, Philip Torr, and Ambrish Tyagi (2020). "Box2Seg: Attention Weighted Loss and Discriminative Feature Learning for Weakly Supervised Segmentation". In: *Proc. Eur. Conf. Comp. Vis.*

Kuo, Weicheng, Anelia Angelova, Jitendra Malik, and Tsung-Yi Lin (2019). "ShapeMask: Learning to segment novel objects by refining shape priors". In: *Proc. IEEE Int. Conf. Comp. Vis.* Pp. 9207–9216.

Law, Hei and Jia Deng (2018). "Cornernet: Detecting objects as paired keypoints". In: *Proc. Eur. Conf. Comp. Vis.* Pp. 734–750.

Lee, Youngwan and Jongyoul Park (2020). "CenterMask: Real-Time Anchor-Free Instance Segmentation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Lin, Tsung-Yi, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie (2017a). "Feature pyramid networks for object detection". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Pp. 2117–2125.

Lin, Tsung-Yi, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár (2017b). "Focal loss for dense object detection". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Pp. 2980–2988.

Lin, Tsung-Yi, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and Lawrence Zitnick (2014). "Microsoft COCO: Common objects in context". In: *Proc. Eur. Conf. Comp. Vis.* Pp. 740–755.

Liu, Fayao, Chunhua Shen, Guosheng Lin, and Ian Reid (2016a). "Learning Depth from Single Monocular Images Using Deep Convolutional Neural Fields". In: *IEEE Trans. Pattern Anal. Mach. Intell.*

Liu, Shu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia (2018). "Path aggregation network for instance segmentation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Pp. 8759–8768.

Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg (2016b). "SSD: Single shot multibox detector". In: *Proc. Eur. Conf. Comp. Vis.* Springer, pp. 21–37.

Liu, Yifan, Changyong Shu, Jingdong Wang, and Chunhua Shen (2020a). "Structured knowledge distillation for dense prediction". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Liu, Yifan, Changyong Shun, Jingdong Wang, and Chunhua Shen (2020b). "Structured Knowledge Distillation for Dense Prediction". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. eprint: 1903.04197. URL: https://ieeexplore.ieee.org/document/9115859.

Liu, Yuliang, Hao Chen, Chunhua Shen, Tong He, Lianwen Jin, and Liangwei Wang (2020c). "ABCNet: Real-time Scene Text Spotting with Adaptive Bezier-Curve Network". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Liu, Yuliang, Hao Chen, Chunhua Shen, Tong He, Lianwen Jin, and Liangwei Wang (2020d). "ABCNet: Real-Time Scene Text Spotting With Adaptive Bezier-Curve Network". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Long, Jonathan, Evan Shelhamer, and Trevor Darrell (2015). "Fully convolutional networks for semantic segmentation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Pp. 3431–3440.

Luo, Wenjie, Yujia Li, Raquel Urtasun, and Richard Zemel (2016). "Understanding the effective receptive field in deep convolutional neural networks". In: *Advances in neural information processing systems*, pp. 4898–4906.

Lyu, Pengyuan, Minghui Liao, Cong Yao, Wenhao Wu, and Xiang Bai (2018). "Mask TextSpotter: An End-to-End Trainable Neural Network for Spotting Text with Arbitrary Shapes". In: *Proc. Eur. Conf. Comp. Vis.*

M. Abadi et al. (2016). "TensorFlow: A system for large-scale machine learning". In: *USENIX Symp. Operating Systems Design & Implementation (OSDI)*, pp. 265–283.

Milan, Anton, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler (2016). "MOT16: A benchmark for multi-object tracking". In: *arXiv preprint arXiv:1603.00831*.

Milletari, Fausto, Nassir Navab, and Seyed-Ahmad Ahmadi (2016). "V-net: Fully convolutional neural networks for volumetric medical image segmentation". In: *Proc. Int. Conf. 3D Vision (3DV)*. IEEE, pp. 565–571.

Neven, Davy, Bert De Brabandere, Marc Proesmans, and Luc Van Gool (2019). "Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Pp. 8837–8845.

Newell, Alejandro, Zhiao Huang, and Jia Deng (2017). "Associative embedding: End-to-end learning for joint detection and grouping". In: *Proc. Advances in Neural Inf. Process. Syst.* Pp. 2277–2287.

Newell, Alejandro, Kaiyu Yang, and Jia Deng (2016). "Stacked hourglass networks for human pose estimation". In: *Proc. Eur. Conf. Comp. Vis.* Springer, pp. 483–499.

Nie, Xuecheng, Jiashi Feng, Jianfeng Zhang, and Shuicheng Yan (2019). "Single-Stage Multi-Person Pose Machines". In: *The IEEE International Conference on Computer Vision (ICCV)*.

Novotny, David, Samuel Albanie, Diane Larlus, and Andrea Vedaldi (2018). "Semi-convolutional Operators for Instance Segmentation". In: *Proceedings of the European Conference on Computer Vision (ECCV)*.

Papandreou, George, Liang-Chieh Chen, Kevin Murphy, and Alan Yuille (2015). "Weakly- and semi-supervised learning of a deep convolutional network for semantic image segmentation". In: *Proc. IEEE Int. Conf. Comp. Vis.* Pp. 1742–1750.

Papandreou, George, Tyler Zhu, Liang-Chieh Chen, Spyros Gidaris, Jonathan Tompson, and Kevin Murphy (2018). "Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 269–286.

Papandreou, George, Tyler Zhu, Nori Kanazawa, Alexander Toshev, Jonathan Tompson, Chris Bregler, and Kevin Murphy (2017). "Towards accurate multi-person pose

estimation in the wild". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4903–4911.

Perez, Ethan, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville (2018). "Film: Visual reasoning with a general conditioning layer". In: *Proc. AAAI Conf. Artificial Intell.*

Pishchulin, Leonid, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter V Gehler, and Bernt Schiele (2016). "Deepcut: Joint subset partition and labeling for multi person pose estimation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4929–4937.

Pishchulin, Leonid, Arjun Jain, Mykhaylo Andriluka, Thorsten Thormählen, and Bernt Schiele (2012). "Articulated people detection and pose estimation: Reshaping the future". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 3178–3185.

Plummer, Bryan A, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik (2015). "Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models". In: *Proceedings of the IEEE international conference on computer vision*, pp. 2641–2649.

Pont-Tuset, Jordi, Pablo Arbelaez, Jonathan Barron, Ferran Marques, and Jitendra Malik (2016). "Multiscale combinatorial grouping for image segmentation and object proposal generation". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 39.1, pp. 128–140.

Rajchl, Martin, Matthew Lee, Ozan Oktay, Konstantinos Kamnitsas, Jonathan Passerat-Palmbach, Wenjia Bai, Mellisa Damodaram, Mary Rutherford, Joseph Hajnal, Bernhard Kainz, et al. (2016). "Deepcut: Object segmentation from bounding box annotations using convolutional neural networks". In: *IEEE Trans. Medical Imaging* 36.2, pp. 674–683.

Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi (2016). "You only look once: Unified, real-time object detection". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Pp. 779–788.

Redmon, Joseph and Ali Farhadi (2017). "YOLO9000: better, faster, stronger". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Pp. 7263–7271.

Redmon, Joseph and Ali Farhadi (2018). "Yolov3: An incremental improvement". In: *arXiv: Comp. Res. Repository* abs/1804.02767.

Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun (2015). "Faster R-CNN: Towards real-time object detection with region proposal networks". In: *Proc. Advances in Neural Inf. Process. Syst.* Pp. 91–99.

Rezatofighi, Hamid, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese (2019). "Generalized intersection over union: A metric and a loss for bounding box regression". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Pp. 658–666.

Rother, Carsten, Vladimir Kolmogorov, and Andrew Blake (2004). "GrabCut: interactive foreground extraction using iterated graph cuts". In: *ACM Trans. Graphics* 23.3, pp. 309–314.

Shao, Shuai, Zijian Zhao, Boxun Li, Tete Xiao, Gang Yu, Xiangyu Zhang, and Jian Sun (2018). "Crowdhuman: A benchmark for detecting human in a crowd". In: *arXiv: Comp. Res. Repository* abs/1805.00123.

Shen, Chunhua, Peng Wang, Sakrapee Paisitkriangkrai, and Anton van den Hengel (2013). "Training Effective Node Classifiers for Cascade Classification". In: *Int. J. Comput. Vision* 103.3, pp. 326–347.

Sofiiuk, Konstantin, Olga Barinova, and Anton Konushin (2019). "Adaptis: Adaptive instance selection network". In: *Proc. IEEE Int. Conf. Comp. Vis.* Pp. 7355–7363.

Song, Chunfeng, Yan Huang, Wanli Ouyang, and Liang Wang (2019). "Box-driven class-wise region masking and filling rate guided loss for weakly supervised semantic segmentation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Pp. 3136–3145.

Sun, Ke, Bin Xiao, Dong Liu, and Jingdong Wang (2019). "Deep High-Resolution Representation Learning for Human Pose Estimation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5693–5703.

Sun, Xiao, Bin Xiao, Fangyin Wei, Shuang Liang, and Yichen Wei (2018). "Integral human pose regression". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 529–545.

Tan, Mingxing, Ruoming Pang, and Quoc Le (2020). "EfficientDet: Scalable and efficient object detection". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Tian, Zhi, Hao Chen, and Chunhua Shen (2019). "DirectPose: Direct End-to-End Multi-Person Pose Estimation". In: *arXiv: Comp. Res. Repository* abs/1911.07451.

Tian, Zhi, Tong He, Chunhua Shen, and Youliang Yan (2019a). "Decoders Matter for Semantic Segmentation: Data-Dependent Decoding Enables Flexible Feature Aggregation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Pp. 3126–3135.

Tian, Zhi, Chunhua Shen, and Hao Chen (2020). "Conditional Convolutions for Instance Segmentation". In: *Proc. Eur. Conf. Comp. Vis.*

Tian, Zhi, Chunhua Shen, Hao Chen, and Tong He (2019b). "FCOS: Fully Convolutional One-Stage Object Detection". In: *Proc. IEEE Int. Conf. Comp. Vis.*

Viola, Paul and Michael Jones (2001). "Robust Real-time Object Detection". In: *Int. J. Comput. Vision.*

Wang, Guangting, Chong Luo, Xiaoyan Sun, Zhiwei Xiong, and Wenjun Zeng (2020a). "Tracking by Instance Detection: A Meta-Learning Approach". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Wang, Xinlong, Tao Kong, Chunhua Shen, Yuning Jiang, and Lei Li (2020b). "SOLO: Segmenting Objects by Locations". In: *Proc. Eur. Conf. Comp. Vis.*

Wang, Xinlong, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen (2020c). "SOLOv2: Dynamic and Fast Instance Segmentation". In: *Proc. Advances in Neural Information Processing Systems (NeurIPS).*

Wang, Xinlong, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen (2020d). "SOLOv2: Dynamic and Fast Instance Segmentation". In: *Proc. Advances in Neural Inf. Process. Syst.*

Wang, Zhongdao, Liang Zheng, Yixuan Liu, and Shengjin Wang (2019). "Towards Real-Time Multi-Object Tracking". In: *arXiv preprint arXiv:1909.12605*.

Wei, Shih-En, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh (2016). "Convolutional pose machines". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4724–4732.

Wu, Yuxin and Kaiming He (2018). "Group normalization". In: *Proc. Eur. Conf. Comp. Vis.* Pp. 3–19.

Wu, Yuxin, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick (2019). *Detectron2*. https://github.com/facebookresearch/detectron2.

Xiao, Bin, Haiping Wu, and Yichen Wei (2018). "Simple baselines for human pose estimation and tracking". In: *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 466–481.

Xie, Enze, Peize Sun, Xiaoge Song, Wenhai Wang, Xuebo Liu, Ding Liang, Chunhua Shen, and Ping Luo (2020). "Polarmask: Single shot instance segmentation with polar representation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Pp. 12193–12202.

Yang, Brandon, Gabriel Bender, Quoc V Le, and Jiquan Ngiam (2019a). "CondConv: Conditionally Parameterized Convolutions for Efficient Inference". In: *Proc. Advances in Neural Inf. Process. Syst.* Pp. 1305–1316.

Yang, Ze, Shaohui Liu, Han Hu, Liwei Wang, and Stephen Lin (2019b). "Reppoints: Point set representation for object detection". In: *Proc. IEEE Int. Conf. Comp. Vis.* Pp. 9657–9666.

Yin, Wei, Yifan Liu, Chunhua Shen, and Youliang Yan (2019). "Enforcing geometric constraints of virtual normal for depth prediction". In: *Proc. IEEE Int. Conf. Comp. Vis.*

Yin, Wei, Xinlong Wang, Chunhua Shen, Yifan Liu, Zhi Tian, Songcen Xu, Changming Sun, and Dou Renyin (2020). "DiverseDepth: Affine-invariant Depth Prediction Using Diverse Data". In: *arXiv preprint arXiv:2002.00569*.

Yu, Fisher, Dequan Wang, Evan Shelhamer, and Trevor Darrell (2018). "Deep layer aggregation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Pp. 2403–2412.

Yu, Jiahui, Yuning Jiang, Zhangyang Wang, Zhimin Cao, and Thomas Huang (2016). "Unitbox: An advanced object detection network". In: *Proc. ACM Int. Conf. Multimedia*. ACM, pp. 516–520.

Zhang, Rufeng, Zhi Tian, Chunhua Shen, Mingyu You, and Youliang Yan (2020). "Mask Encoding for Single Shot Instance Segmentation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Zhang, Rui, Yongsheng Zhou, Qianyi Jiang, Qi Song, Nan Li, Kai Zhou, Lei Wang, Dong Wang, Minghui Liao, Mingkun Yang, et al. (2019). "ICDAR 2019 robust reading challenge on reading chinese text on signboard". In: *Proc. Int. Conf. Document Analysis Recogn.* Pp. 1577–1581.

Zhou, Xingyi, Dequan Wang, and Philipp Krähenbühl (2019). "Objects as Points". In: *arXiv: Comp. Res. Repository.* Vol. abs/1904.07850.

Zhou, Xinyu, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang (2017). "EAST: an efficient and accurate scene text detector". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Pp. 5551–5560.

Zhou, Yanzhao, Xin Wang, Jianbin Jiao, Trevor Darrell, and Fisher Yu (2020). "Learning Saliency Propagation for Semi-supervised Instance Segmentation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Zhu, Chenchen, Yihui He, and Marios Savvides (2019). "Feature Selective Anchor-Free Module for Single-Shot Object Detection". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Zhu, Xizhou, Han Hu, Stephen Lin, and Jifeng Dai (2019). "Deformable convnets v2: More deformable, better results". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Pp. 9308–9316.