THE UNIVERSITY of ADELAIDE

Deep Object Segmentation and Beyond

Xinlong Wang

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

September 13, 2022

Contents

Abstract xv

Declaration of Authorship xvii

Acknowledgements xix

1 Introduction 1
- 1.1 Instance Segmentation Made Simpler . . . 1
- 1.2 High-Performance Instance Segmentation . . . 2
- 1.3 Self-Supervised Learning for Dense Predictions . . . 3
- 1.4 Self-Supervised Instance Segmentation . . . 3

2 Literature Review 5
- 2.1 Instance Segmentation . . . 5
- 2.2 Image Matting . . . 6
- 2.3 Self-Supervised Learning . . . 7
- 2.4 Unsupervised Object Localization . . . 8

3 SOLO: Segmenting Objects by Locations 13
- 3.1 Introduction . . . 13
- 3.2 Our Method: SOLO . . . 15
  - 3.2.1 Problem Formulation . . . 15
    - Semantic Category . . . 16
    - Instance Mask . . . 16
    - Forming Instance Segmentation . . . 17
  - 3.2.2 Network Architecture . . . 17
  - 3.2.3 SOLO Learning . . . 18
    - Label Assignment . . . 18
    - Loss Function . . . 18
  - 3.2.4 Inference . . . 19
- 3.3 Experiments . . . 19
  - 3.3.1 Main Results . . . 20
  - 3.3.2 How SOLO Works? . . . 20
  - 3.3.3 Ablation Experiments . . . 20
  - 3.3.4 SOLO-512 . . . 23

3.3.5 Error Analysis . . . 24
3.4 Decoupled SOLO . . . 24
3.5 Conclusion . . . 25

4 SOLOv2: Dynamic and Fast Instance Segmentation 31
4.1 Introduction . . . 31
4.2 Proposed Method: SOLOv2 . . . 33
4.2.1 Dynamic Instance Segmentation . . . 34
Mask Kernel G . . . 35
Mask Feature F . . . 35
Forming Instance Mask . . . 36
Learning and Inference . . . 36
4.2.2 Matrix NMS . . . 36
4.3 Experiments . . . 38
4.3.1 Instance Segmentation . . . 38
Main Results . . . 38
Ablation Experiments . . . 39
4.3.2 Extension: Object Detection . . . 42
4.3.3 Extension: Panoptic Segmentation . . . 42
4.3.4 Extension: Instance-level Image Matting . . . 43
Method . . . 44
Dataset . . . 45
Results . . . 45
4.4 Conclusion . . . 46

5 Dense Contrastive Learning for Self-Supervised Visual Pre-Training 51
5.1 Introduction . . . 51
5.2 Method . . . 53
5.2.1 Background . . . 53
5.2.2 DenseCL Pipeline . . . 54
5.2.3 Dense Contrastive Learning . . . 55
5.2.4 Dense Correspondence across Views . . . 55
5.3 Experiments . . . 56
5.3.1 Experimental Settings . . . 57
5.3.2 Main Results . . . 58
5.3.3 Ablation Study . . . 60
5.3.4 Discussions on DenseCL . . . 64
5.4 Conclusion . . . 65

6 FreeSOLO: Learning to Segment Objects without Annotations 69
6.1 Introduction . . . 69
6.2 Method . . . 71
6.2.1 Overview of FreeSOLO . . . 71

6.2.2 Free Mask 71
6.2.3 Self-Supervised SOLO 73
6.3 Experiments 75
6.3.1 Experimental Settings 75
6.3.2 Main Results 77
6.3.3 Ablation Study 78
6.4 Conclusion 82

7 Conclusion **85**

Bibliography **87**

List of Figures

3.1 Comparison of the pipelines of Mask R-CNN and the proposed SOLO. 13

3.2 **SOLO framework.** We reformulate the instance segmentation as two sub-tasks: category prediction and instance mask generation problems. An input image is divided into a uniform grids, *i.e.*, $S \times S$. Here we illustrate the grid with $S = 5$. If the center of an object falls into a grid cell, that grid cell is responsible for predicting the semantic category (top) and masks of instances (bottom). We do not show the feature pyramid network (FPN) here for simpler illustration. 15

3.3 **SOLO Head architecture**. At each FPN feature level, we attach two sibling sub-networks, one for instance category prediction (top) and one for instance mask segmentation (bottom). In the mask branch, we concatenate the x, y coordinates and the original features to encode spatial information. Here numbers denote spatial resolution and channels. In the figure, we assume 256 channels as an example. Arrows denote either convolution or interpolation. 'Align' means bilinear interpolation. '$\times 7$' means 7 convolutions. '$2H \times 2W$' is for predicting masks at higher resolutions. During inference, the mask branch outputs are further upsampled to the original image size. 17

3.4 **SOLO behavior.** We show the visualization of soft mask prediction before NMS. Here $S = 12$. For each column, the top one is the instance segmentation result, and the bottom one shows the mask activation maps. The sub-figure (i, j) in an activation map indicates the mask prediction results (after zooming out) generated by the corresponding mask channel. 21

3.5 **Decoupled SOLO head**. F is input feature. Dashed arrows denote convolutions. $k = i \cdot S + j$. '$\otimes$' denotes element-wise multiplication. . 25

3.6 **Visualization of instance segmentation results** using the Res-101-FPN backbone. The model is trained on the COCO `train2017` dataset, achieving a mask AP of 37.8 on the COCO `test-dev`. 25

4.1 Comparison of instance segmentation performance by SOLOv2 and other methods on the COCO `test-dev`. (a) The proposed SOLOv2 outperforms a range of state-of-the-art algorithms. All methods are evaluated using one Tesla V100 GPU. (b) SOLOv2 obtains higher-quality masks compared with Mask R-CNN. Mask R-CNN's mask head is typically restricted to 28×28 resolution, leading to inferior prediction at object boundaries. 32

4.2 **SOLOv2** compared to SOLO. I is the input feature after FCN-backbone representation extraction. Dashed arrows denote convolutions. $k = i \cdot S + j$; and '$\circledast$' denotes the dynamic convolution operation. 34

4.3 **Visualization of instance segmentation results** using SOLOv2 ResNet-101 backbone. The model is trained on the COCO `train2017` dataset, achieving a mask AP of 39.7% on the COCO `test-dev`. . . . 39

4.4 SOLOv2 for **object detection**. Speed-accuracy trade-off of bounding-box detection on the COCO `test-dev`. 43

4.5 Behavior of the Details Refinement module. Each plotted sub-figure corresponds to one of the 32 channels of the input/output feature maps. The output feature map recovers the low-level details. Best viewed on screens. 45

4.6 (a) Visualization of image matting results. We show the input image and the corresponding output alpha matte of our method. Best viewed on screens. (b) Demonstration of image editing applications (from left to right: original image; Bokeh background effect focusing on one instance; image composition). 46

5.1 Conceptual illustration of two contrastive learning paradigms for representation learning. We use a pair of query and key for simpler illustration. The backbone can be any convolutional neural network. (a): The contrastive loss is computed between the single feature vectors outputted by the global projection head, at the level of global feature; (b): The dense contrastive loss is computed between the dense feature vectors outputted by the dense projection head, at the level of local feature. For both paradigms, the two branches can be the same encoder or different ones, *e.g.*, an encoder and its momentum-updated one. . . 53

5.2 Comparisons of pre-trained models by fine-tuning on object detection and semantic segmentation datasets. 'Sup. IN' denotes the supervised pre-training on ImageNet. 'COCO' and 'ImageNet' indicate the pre-training models trained on COCO and ImageNet respectively. (a): The object detection results of a Faster R-CNN detector fine-tuned on VOC `trainval07+12` for 24k iterations and evaluated on VOC `test2007`; (b): The semantic segmentation results of an FCN model fine-tuned on VOC `train_aug2012` for 20k iterations and evaluated on `val2012`. The results are averaged over 5 independent trials. 57

5.3 Different pre-training schedules on COCO. For each pre-trained model, a Faster R-CNN detector is fine-tuned on VOC `trainval07+12` for 24k iterations and evaluated on `test2007`. The metric is the COCO-style AP. Results are averaged over 5 independent trials. 63

5.4 Visualization of dense correspondence. The correspondence is extracted between two views of the same image, using the 200-epoch ImageNet pre-trained model. DenseCL extracts more high-similarity matches compared with MoCo-v2. Best viewed on screen. 63

5.5 Comparison of dense correspondence extracted from random initialization to well trained DenseCL. The correspondence is extracted between two views of the same image, using the ImageNet pre-trained model. All the matches are visualized without thresholding. 64

6.1 **Overview of FreeSOLO**. Unlabeled images are first input to Free Mask to generate coarse object masks. The segmentation masks as well as their associated semantic embeddings are used to train a SOLO-based instance segmentation model via weak supervision. We use self-training to improve object mask segmentation. 70

6.2 **The Free Mask approach.** Given queries and keys from the backbone feature **I**, the keys are convolved by the queries to generate segmentation masks. The masks go through NMS to form the object mask outputs. 72

6.3 **Qualitative results of FreeSOLO for the task of class-agnostic instance segmentation.** The model is trained *without any kind of manual annotations* and can infer at 16 FPS on a V100 GPU. Best viewed on screen. 76

6.4 Qualitative results of the Free Mask. Free Mask extracts coarse masks of the common objects in unlabeled images. 79

6.5 Qualitative comparison of with and without $\mathcal{L}_{avg\_proj}$ when learning from coarse masks. The model trained without $\mathcal{L}_{avg\_proj}$ tends to only segment the contours when trained longer. 81

6.6 **More qualitative results of FreeSOLO for the task of class-agnostic instance segmentation.** The model is trained *without any kind of manual annotations* and can infer at 16 FPS on a V100 GPU. Best viewed on screen. 81
6.7 Qualitative comparison of FreeSOLO's predicted masks and ground truth masks. At some object boundaries, FreeSOLO can produce *even more precise* segmentation than manual annotations in some cases. . 82
6.8 Failure cases of FreeSOLO. Our method could fail to localize objects that are truncated, crowded or small. 82

List of Tables

3.1 **Instance segmentation mask AP** (%) on the COCO `test-dev`. All entries are *single-model* results. Here we adopt the "6×" schedule (72 epochs), following Chen et al. (2019b). Mask R-CNN* is our improved version with scale augmentation and longer training time. D-SOLO means Decoupled SOLO as introduced in Section 3.4. 20
3.2 The impact of **grid number and FPN**. FPN significantly improves the performance thanks to its ability to deal with varying sizes of objects. 21
3.3 **Conv vs. CoordConv.** CoordConv can considerably improve AP upon standard convolution. Two or more layers of CoordConv are not necessary. 22
3.4 **Different loss functions** may be employed in the mask branch. The Dice loss (DL) leads to best AP and is more stable to train. 23
3.5 **Different head depth.** We use depth being 7 in other experiments, as the performance becomes stable when the depth grows beyond 7. . 23
3.6 **SOLO-512**. SOLO-512 uses a model with smaller input size. All models are evaluated on `val2017`. Here the models are trained with "6×" schedule. 23
3.7 **Error analysis**. Replacing the predicted instance mask with the ground-truth ones improves the mask AP from 37.1 to 68.1, suggesting that the mask branch still has ample room to be improved. The models are based on ResNet-101-FPN. 24

4.1 **Instance segmentation** mask AP (%) on COCO `test-dev`. All entries are *single-model* results. Mask R-CNN* is our improved version with scale augmentation and longer training time (6×). 'DCN' means deformable convolutions used. 37
4.2 Instance segmentation results on the LVISv0.5 validation dataset. $*$ means re-implementation. 39
4.3 **Kernel shape.** The performance is stable when the shape goes beyond $1 \times 1 \times 256$. 40
4.4 **Explicit coordinates.** Precise coordinates input can considerably improve the results. 40
4.5 **Mask feature representation.** We compare the separate mask feature representation in parallel heads and the unified representation. . 41

4.6 **Matrix NMS.** Matrix NMS outperforms other methods in both speed and accuracy. 41
4.7 **Real-time SOLOv2**. The speed is reported on a single V100 GPU by averaging 5 runs (on COCO `test-dev`). 42
4.8 **Training schedule.** 1× means 12 epochs using single-scale training. 3× means 36 epochs with multi-scale training. 42
4.9 **Object detection** box AP (%) on the MS COCO `test-dev`. Although our bounding boxes are directly generated from the predicted masks, the accuracy outperforms most state-of-the-art methods. Speed-accuracy trade-off of typical methods is shown in Figure 4.4. 43
4.10 SOLOv2 for **panoptic segmentation** – results on MS COCO val2017. ∗ means re-implementation. 44

5.1 **Object detection fine-tuned on PASCAL VOC.** 'CC' and 'IN' indicate the pre-training models trained on COCO and ImageNet respectively. The models pre-trained on the same dataset are with the same training epochs, *i.e.*, 800 epochs for COCO and 200 epochs for ImageNet. '*' means re-implementation. The results of other methods are either from their papers or third-party implementation. All the detectors are trained on `trainval07+12` for 24k iterations and evaluated on `test2007`. The metrics include the VOC metric AP_{50} (*i.e.*, IoU threshold is 50%) and COCO-style AP and AP_{75}. The results are averaged over 5 independent trials. 59
5.2 **SOLOv2 instance segmentation and FCOS object detection fine-tuned on COCO.** 'CC' and 'IN' indicate the pre-training models trained on COCO and ImageNet respectively. All the models are trained on `train2017` with default 1× schedule and evaluated on `val2017`. The metrics include mask AP (AP^m) and bounding box AP (AP^b). . 59
5.3 **Object detection and instance segmentation fine-tuned on COCO.** 'CC' and 'IN' indicate the pre-training models trained on COCO and ImageNet respectively. All the detectors are trained on `train2017` with default 1× schedule and evaluated on `val2017`. The metrics include bounding box AP (AP^b) and mask AP (AP^m). 60
5.4 **Semi-supervised object detection and instance segmentation fine-tuned on COCO.** During the fine-tuning, only 10% training data is used. 'CC' and 'IN' indicate the pre-training models trained on COCO and ImageNet respectively. All the detectors are trained on `train2017` for 90k iterations and evaluated on `val2017`. The metrics include bounding box AP (AP^b) and mask AP (AP^m). 60

5.5 **Semantic segmentation on PASCAL VOC and Cityscapes.** 'CC' and 'IN' indicate the pre-training models trained on COCO and ImageNet respectively. The metric is the commonly used mean IoU (mIoU). Results are averaged over 5 independent trials. 61
5.6 **Ablation study of weight λ.** $\lambda = 0$ is the MoCo-v2 baseline. $\lambda = 0.5$ shows the best trade-off between detection and classification. '*' indicates training with warm-up, as discussed in Section 5.3.4. 62
5.7 **Ablation study of matching strategy.** To extract the dense correspondence according to the backbone features $\mathbf{F}_1$ and $\mathbf{F}_2$ shows the best results. 62
5.8 **Ablation study of grid size S.** The results increase as the S gets larger. We use grid size being 7 in other experiments, as the performance becomes stable when the S grows beyond 7. 63
5.9 **Ablation study of training schedule.** The results consistently improve as the training schedule gets longer. Although 1600-epoch training schedule is 0.5% AP better, we use 800-epoch schedule in other experiments for faster training. 64
5.10 **Pre-training time comparison.** The training time per epoch is reported. We measure the results on the same 8-GPU machine. The training time overhead introduced by DenseCL is less than 1%. 65

6.1 **Class-agnostic instance segmentation** on MS COCO `val2017`. Both MCG and COB require annotations more or less. 77
6.2 **Class-agnostic instance segmentation** on UVO `val` split. Results of Mask R-CNN are from the paper of UVO (Wang et al., 2021a). . . 77
6.3 **Unsupervised class-agnostic object detection** on MS COCO `val2017`. Compared results are directly from DETReg. 78
6.4 **Multi-object discovery** on PASCAL VOC `trainval07` and MS COCO `20k`. LOST* is a concurrent work. 78
6.5 **Supervised instance segmentation** with limited fully annotated images. 79
6.6 **Supervised instance segmentation** with limited segmentation masks. 79
6.7 **FreeSOLO ablation experiments.** All the experiments are with a ResNet-50 backbone. We report class-agnostic instance segmentation results (a-e) and supervised fine-tuning results (f) on the COCO `val2017` split. 80

University of Adelaide

Abstract

Deep Object Segmentation and Beyond

by Xinlong Wang

Object segmentation is a fundamental computer vision problem that aims to recognize the object of interest and group the corresponding pixels in an image. With wide applications in self-driving cars, medical imaging, augmented reality, *etc.*, object segmentation has attracted a lot of research attention. In this thesis, we propose a series of methods to solve the challenging problem with deep neural networks. We further generalize the proposed methods to solve extensive tasks such as image matting and study the interactions between object segmentation and unsupervised learning.

First, we propose segmenting objects by locations (SOLO), a new, embarrassingly simple approach to segment all the object instances in an image and recognize their categories. Unlike previous methods that rely on either bounding box detection or grouping post-processing, SOLO directly maps a raw input image to the desired object categories and instance masks with a fully convolutional network. We demonstrate a much simpler and flexible instance segmentation framework with strong performance.

Second, we present SOLOv2, a dynamic and fast instance segmentation solution that follows the principle of SOLO but improves it in terms of both speed and accuracy. SOLOv2 achieves state-of-the-art results with high efficiency, making it suitable for both mobile and cloud applications. We further demonstrate the generality of our method by extending it to perform panoptic segmentation and image matting.

Third, we propose dense contrastive learning (DenseCL) to learn better representation from large-scale unlabeled images for dense prediction tasks such as segmentation. The proposed DenseCL performs dense pairwise contrastive learning at the level of pixels. Our method largely closes the gap between self-supervised pre-training and downstream dense prediction tasks.

Finally, we propose a fully unsupervised learning method that learns to segment objects without any annotations. We present FreeSOLO, a self-supervised instance segmentation framework built on top of our simple-yet-effective methods SOLO(v2) for segmentation, and DenesCL for unsupervised learning. For the first time, we demonstrate unsupervised instance segmentation successfully.

The code and models are publicly available at `https://github.com/WXinlong`.

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.

Xinlong Wang

September 13, 2022

Acknowledgements

First and foremost, my great thanks go to my supervisor Chunhua Shen. I could not have undertaken this journey without him. I met Chunhua in the summer of 2016 when I was an undergraduate student who knew nothing about computer vision or deep learning. At that time, I didn't even plan to pursue a master's degree, no mention to be a PhD student. However, everything changed when he brought me to this fascinating field. I spent wonderful years working on computer vision under his advisory. Chunhua has very high-level understanding and insights about the field while he is also uncommonly detail-oriented. More importantly, he is an extremely kind and supportive supervisor that I could not have asked for more. I am forever grateful to him.

I am also very grateful to Yuning Jiang for his invaluable mentorship, and the great support in my hard times. Special thanks go to Tao Kong, Shu Liu, Gang Yu, Jiaya Jia, and Lei Li, who gave me a lot of support at various times. I also would like to thank all the co-authors during my PhD: Rufeng Zhang, Tong He, Yifan Liu, Xingyu Zhang, Jia-Wang Bian, Mingyu You, Zhi Tian, Hao Chen, Yuqing Wang, Weian Mao, Zhiding Yu, Shalini De Mello, Jan Kautz, Anima Anandkumar, and Jose M. Alvarez. I acknowledge the financial support for my research from Adelaide International Scholarship and Google PhD Fellowship.

I thank my girlfriend Wanxuan Lu for her love and support. We met each other at the beautiful Adelaide. Last but not least, I would like to thank my parents Mingliang Wang and Fengxiu He for their unconditional love. They made me who I am today and I never know how to pay them back.

Publications

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

- *SOLO: Segmenting Objects by Locations*
  Xinlong Wang, Tao Kong, Chunhua Shen, Yuning Jiang, Lei Li
  European Conference on Computer Vision (ECCV), 2020

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

- *SOLO: A Simple Framework for Instance Segmentation*
  Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, Lei Li
  IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2021

- *Dense Contrastive Learning for Self-Supervised Visual Pre-Training*
  Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, Lei Li
  IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021

- *FreeSOLO: Learning to Segment Objects without Annotations*
  Xinlong Wang, Zhiding Yu, Shalini De Mello, Jan Kautz, Anima Anandkumar, Chunhua Shen, Jose M. Alvarez
  IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2022

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

- *Instance-Aware Embedding for Point Cloud Instance Segmentation*
  Tong He, Yifan liu, Chunhua Shen, **Xinlong Wang**, Changming Sun
  European Conference on Computer Vision (ECCV), 2020

- *Diverse Knowledge Distillation for End-to-End Person Search*
  Xinyu Zhang, **Xinlong Wang**, Jia-Wang Bian, Chunhua Shen, Mingyu You
  AAAI Conference on Artificial Intelligence (AAAI), 2021

- *BoxInst: High-Performance Instance Segmentation with Box Annotations*
  Zhi Tian, Chunhua Shen, **Xinlong Wang**, Hao Chen
  IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021

- *End-to-End Video Instance Segmentation with Transformers*
  Yuqing Wang, Zhaoliang Xu, **Xinlong Wang**, Chunhua Shen, Baoshan Cheng, Hao Shen, Huaxia Xia
  IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021

- *FCPose: Fully Convolutional Multi-Person Pose Estimation with Dynamic Instance-Aware Convolutions*
  Weian Mao, Zhi Tian, **Xinlong Wang**, Chunhua Shen
  IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021

Chapter 1

Introduction

Object segmentation requires the computer to localize the objects in an image at the pixel level, *i.e.*, to recognize the objects of interest (*e.g.*, cars, pedestrians, and traffic signs) and output segmentation masks covering the corresponding objects. Object instance segmentation is one of the most fundamental problems in computer vision, with wide applications in self-driving cars, medical imaging, augmented reality, *etc.*.

Human vision can effortless localize the target objects precisely in a complex scene. However, it is still very challenging for computers, even after sixty years of development in the field of computer vision. The main obstacles are two-fold. 1) The problem of object instance segmentation is hard to formulate. The objects can be at any position of the image, with a wide variety of categories, poses, scales, and occlusions. Instance segmentation requires the correct separation of all objects in an image while also semantically segmenting each instance at the pixel level (Hariharan et al., 2014), making it one of the most challenging problems in computer vision. 2) The human annotation for instance segmentation is tremendously time-consuming and expensive. Training an instance segmentation model requires pixel-wise segmentation masks for every single object (Lin et al., 2014; Cordts et al., 2016), which results in a much higher labeling cost than other vision tasks such as image classification and object detection. The extremely expensive annotation largely impedes its development and wide applications.

To address these problems, we propose a series of novel methods in this thesis, which formulate the problem with a simple, fast and accurate framework, and enable to learn segmentation models with limited or even no human supervision.

1.1 Instance Segmentation Made Simpler

Existing instance segmentation methods can be categorized into two groups, *i.e.*, top-down (Li et al., 2017; He et al., 2017b; Chen et al., 2019a; Chen et al., 2019b) and bottom-up (Newell, Huang, and Deng, 2017; De Brabandere, Neven, and Van Gool, 2017; Liu et al., 2017; Gao et al., 2019) paradigms. The former approach, namely "detect-then-segment", first detects bounding boxes and then segments the instance

mask in each bounding box. The latter approach learns an affinity relation, assigning an embedding vector to each pixel, by pushing away pixels belonging to different instances and pulling close pixels in the same instance. A grouping post-processing is then needed to separate instances. Both these two paradigms are step-wise and indirect, which either heavily rely on accurate bounding box detection or depend on per-pixel embedding learning and the grouping processing.

In this thesis, we propose a simple framework for instance segmentation that is end-to-end trainable and can directly map a raw input image to the desired instance masks with constant inference time, eliminating the need for the grouping post-processing as in bottom-up methods or the bounding-box detection and RoI operations in top-down approaches. Specifically, inspired by semantic segmentation (Long, Shelhamer, and Darrell, 2015a) that segments semantics by distinguishing different semantic categories, we propose to distinguish object instances in the image by introducing the notion of "*instance categories*", *i.e.*, the quantized center locations and object sizes, which enables **s**egmenting **o**bjects by **lo**cations, thus the name of our method, **SOLO.** SOLO formulates the task of instance segmentation as two sub-tasks of pixel-level classification, solvable using standard FCNs, thus dramatically simplifying the formulation of instance segmentation. For the first time, we demonstrate a very simple instance segmentation approach achieving *on par* results with the dominant "detect-then-segment" method on the challenging COCO dataset (Lin et al., 2014) with diverse scenes and semantic classes. Details will be introduced in Chapter 3.

1.2 High-Performance Instance Segmentation

SOLO has significantly simplified the challenging instance segmentation task and made it as simple as other well-formulated dense prediction tasks such as depth estimation and semantic segmentation. In SOLOv2, we further propose various improvements to enhance SOLO in terms of both speed and accuracy.

We observe that three main bottlenecks limit the performance of SOLO: 1) inefficient mask representation and learning; 2) not high enough resolution for finer mask predictions; 3) slow mask non-maximum suppression (NMS). In SOLOv2, we eliminate the above bottlenecks all at once. We first introduce a dynamic scheme, which enables dynamically segmenting objects by locations. Specifically, the mask learning process can be divided into two parts: convolution kernel learning and feature learning. We further construct a unified and high-resolution mask feature representation for instance-aware segmentation. As such, we are able to predict high-resolution object masks, as well as learn the mask kernels and mask features separately and efficiently. At last, we propose an efficient and effective matrix NMS algorithm to remove redundant mask predictions, replacing the original greedy NMS. Matrix NMS can process 500 masks in less than 1 ms in simple python implementation by performing NMS with parallel matrix operations in one shot. With the proposed improvements, our

approach achieves state-of-the-art results for instance segmentation in terms of both speed and accuracy, while being considerably simpler than the existing methods.

By adding the semantic segmentation branch, we easily extend our method to solve panoptic segmentation and achieve state-of-the-art results. We also extend our framework to perform image matting at the instance level. Thanks to the ability to generate high-quality object masks, our method is able to solve instance-level image matting in one shot with minimal modifications. Details are in Chapter 4.

1.3 Self-Supervised Learning for Dense Predictions

The simple architecture and decent speed/accuracy trade-off of SOLO and SOLOv2 largely lower the bar of using and inferencing instance segmentation models. However, the bar of training instance segmentation is still high. Training an instance segmentation model is very costly as it requires both the instance-level and pixel-level annotations. The notoriously time-consuming and expensive annotation is the main obstacle that impedes instance segmentation's broad applications.

In this thesis, we propose a simple framework for self-supervised visual learning, for learning general visual representations from unlabeled images and reducing labeling costs for instance segmentation and other dense prediction tasks. To date, most existing self-supervised learning methods are designed and optimized for image classification (Chen et al., 2020b; He et al., 2020; Grill et al., 2020). These pre-trained models can be sub-optimal for dense prediction tasks due to the discrepancy between image-level prediction and pixel-level prediction. To fill this gap, we design an effective, dense self-supervised learning method that directly works at the level of pixels (or local features) by taking into account the correspondence between local features. Specifically, we present dense contrastive learning (DenseCL), which implements self-supervised learning by optimizing a pairwise contrastive (dis)similarity loss at the pixel level between two views of input images. Our method demonstrates consistently superior performance when transferring to downstream dense prediction tasks including object detection, semantic segmentation, and instance segmentation; and outperforms the state-of-the-art methods by a large margin. Details will be introduced in Chapter 5.

1.4 Self-Supervised Instance Segmentation

SOLO and SOLOv2 enable learning efficient and accurate instance segmentation with sufficient human supervision. DenseCL enables us to learn useful visual representations from unlabeled images, which can benefit downstream object recognition and segmentation tasks. In this thesis, we move one step forward by proposing a fully unsupervised learning method that learns class-agnostic instance segmentation without any annotations.

We present FreeSOLO, a self-supervised instance segmentation framework built on top of the simple yet strong instance segmentation framework of SOLOv2, and the self-supervised dense feature learning method of DenseCL. Our method also presents a novel localization-aware pre-training framework, where objects can be discovered from complicated scenes in an unsupervised manner. Specifically, FreeSOLO contains two major pillars: Free Mask and Self-supervised SOLO. The former extracts coarse object masks in an unsupervised manner. The latter takes the coarse masks and trains the instance segmentation model with several novel designs to overcome label noise in the coarse masks. For the first time, we demonstrate unsupervised class-agnostic instance segmentation successfully. FreeSOLO further demonstrates superiority as a strong pre-training method. We will elaborate on it in Chapter 6.

To summarize, in this thesis, we propose several novel methods, *i.e.*, SOLO, SOLOv2, DenseCL, and FreeSOLO, to solve the challenging instance segmentation problem and the extensive tasks such as image matting, as well as enable to leverage unlabeled data and learn segmentation models with limited or even no manual annotations.

Chapter 2

Literature Review

Here we review some works that are closest to ours.

2.1 Instance Segmentation

Given input images, instance segmentation models are trained to predict the annotated mask and category for each object of interest. To solve this problem, most of the existing instance segmentation methods can be categorized into two groups, *i.e.*, top-down and bottom-up paradigms.

Top-down instance segmentation. The methods that segment object instances in a priori bounding box fall into the typical top-down paradigm. Fully Convolutional Instance-aware Semantic Segmentation (FCIS) (Li et al., 2017) assembles the position-sensitive score maps within the region-of-interests (ROIs) generated by a region proposal network (RPN) to predict instance masks. Mask R-CNN (He et al., 2017b) extends the Faster R-CNN detector (Ren et al., 2015b) by adding a branch for segmenting the object instances within the detected bounding boxes. Based on Mask R-CNN, PANet (Liu et al., 2018b) further enhances the feature representation to improve the accuracy, Mask Scoring R-CNN (Huang et al., 2019) adds a mask-IoU branch to predict the quality of the predicted mask for improving the performance. HTC (Chen et al., 2019a) interweaves box and mask branches for a joint multi-stage processing. TensorMask (Chen et al., 2019b) adopts the dense sliding window paradigm to segment the instance in the local window for each pixel with a predefined number of windows and scales. YOLACT (Bolya et al., 2019) learns a group of coefficients which are normalized to $(-1, 1)$ for each anchor box. During the inference, it first performs a bounding box detection and then uses the predicted boxes to crop the assembled masks. In contrast to the top-down methods above, our SOLO framework is totally box-free thus not being restricted by (anchor) box locations and scales, and naturally benefits from the inherent advantages of FCNs.

Bottom-up instance segmentation. This category of the approaches generate instance masks by grouping the pixels into an arbitrary number of object instances presented in an image. In Newell, Huang, and Deng (2017), pixels are grouped into

instances using the learned associative embedding. A discriminative loss function (De Brabandere, Neven, and Van Gool, 2017) learns pixel-level instance embedding efficiently, by pushing away pixels belonging to different instances and pulling close pixels in the same instance. SGN (Liu et al., 2017) decomposes the instance segmentation problem into a sequence of sub-grouping problems. SSAP (Gao et al., 2019) learns a pixel-pair affinity pyramid, the probability that two pixels belong to the same instance, and sequentially generates instances by a cascaded graph partition. Typically, bottom-up methods lag behind in accuracy compared to top-down methods, especially on the dataset with diverse scenes. Instead of exploiting pixel pairwise relations and pixel grouping, our methods directly learn with the instance mask annotations solely during training, and predicts instance masks end-to-end without grouping post-processing.

Direct instance segmentation. To our knowledge, no prior methods directly train with mask annotations solely, and predict instance masks and semantic categories in one shot without the need of grouping post-processing. Several recently proposed methods may be viewed as the 'semi-direct' paradigm. AdaptIS (Sofiiuk, Barinova, and Konushin, 2019) first predicts point proposals, and then sequentially generates the mask for the object located at the detected point proposal. PolarMask (Xie et al., 2020a) proposes to use the polar representation to encode masks and transforms per-pixel mask prediction to distance regression. They both do not need bounding boxes for training but are either being step-wise or founded on compromise, *e.g.*, coarse parametric representation of masks. Our SOLO framework takes an image as input, directly outputs instance masks and corresponding class probabilities, in a fully convolutional, box-free and grouping-free paradigm.

2.2 Image Matting

Image matting is a fundamental problem in computer vision and graphics and has attracted much research attention (Levin, Lischinski, and Weiss, 2007; Xu et al., 2017; Lu et al., 2019). Given an image, image matting demands for accurate foreground estimation, which is typically formulated as the alpha map prediction, *i.e.*, to output the soft transition between foreground and background. Traditional matting methods can be categorized into two types: the sampling-based methods (Chuang et al., 2001; He et al., 2011) and propagation-based methods (Levin, Lischinski, and Weiss, 2007; Chen, Li, and Tang, 2013; He, Sun, and Tang, 2010). Sampling-based methods infer the alpha values by sampling pixels and solving for alpha using color models. In contrast, propagation-based methods propagate the alpha values of the known foreground and background to the unknown regions. Recently, deep learning methods (Xu et al., 2017; Lu et al., 2019; Shen et al., 2016) have demonstrated superior results for matting. Xu *et al.* (Xu et al., 2017) proposed a fully convolutional network (FCN) to predict the alpha matte of the unknown region in a supervised learning manner, with a trimap being part of the input to the network. Subsequent approaches (Zhang et al.,

2019; Sengupta et al., 2020) explore the solution of learning a mapping directly from the input RGB image to the output alpha matte without explicitly inputting trimaps (trimap-free). Although not requiring the user to provide a trimap explicitly, these methods predict trimaps first by the network and use the predicted trimap for conducting matting by the second part of the network. Thus essentially those methods can be viewed as two-stage approaches to trimap-free matting.

Different from above methods, we perform binary mask segmentation and soft matting *simultaneously*, *i.e.*, soft instance segmentation, within a one-stage FCN framework. Furthermore, inheriting the capability of instance segmentation, we are able to produce a soft matte individually for each foreground instance, which is not possible for existing matting methods.

2.3 Self-Supervised Learning

Self-supervised pre-training. Generally speaking, the success of self-supervised learning (Wu et al., 2018; He et al., 2020; Xie et al., 2020b; Zhao et al., 2020; Han, Xie, and Zisserman, 2020; Grill et al., 2020) can be attributed to two important aspects namely *contrastive learning*, and *pretext tasks*. The objective functions used to train visual representations in many methods are either reconstruction-based loss functions (Doersch, Gupta, and Efros, 2015; Pathak et al., 2016; Goodfellow et al., 2014), or contrastive loss that measures the co-occurrence of multiple views (Tian, Krishnan, and Isola, 2019). Contrastive learning, holds the key to most state-of-the-art methods (Wu et al., 2018; He et al., 2020; Chen et al., 2020b; Xie et al., 2020b), in which the positive pair is usually formed with two augmented views of the same image (or other visual patterns), while negative ones are formed with different images.

A wide range of pretext tasks have been explored to learn a good representation. These examples include colorization (Zhang, Isola, and Efros, 2016), context autoencoders (Doersch, Gupta, and Efros, 2015), inpainting (Pathak et al., 2016), spatial jigsaw puzzles (Noroozi and Favaro, 2016) and discriminate orientation (Gidaris, Singh, and Komodakis, 2018). These methods achieved very limited success in computer vision. The breakthrough approach is SimCLR (Chen et al., 2020b), which follows an instance discrimination pretext task, similar to Wu et al., 2018, where the features of each instance are pulled away from those of all other instances in the training set. Invariances are encoded from low-level image transformations such as cropping, scaling, and color jittering. Contrastive learning and pretext tasks are often combined to form a representation learning framework. DenseCL belongs to the self-supervised pre-training paradigm, and we naturally make the framework friendly for dense prediction tasks such as semantic segmentation and object detection.

Pre-training for dense prediction tasks. Pre-training has enabled surprising results on many dense prediction tasks, including object detection (Ren et al., 2015a;

Redmon and Farhadi, 2018a) and semantic segmentation (Long, Shelhamer, and Darrell, 2015b). These models are usually fine-tuned from ImageNet pre-trained model, which is designed for image-level recognition tasks. Some previous studies have shown the gap between ImageNet pre-training and dense prediction tasks in the context of network architecture (Li et al., 2018; Kong et al., 2016; Tan, Pang, and Le, 2020; Sun et al., 2019). YOLO9000 (Redmon and Farhadi, 2017) proposes to joint train the object detector on both classification and detection data. He, Girshick, and Dollár (2019) demonstrate that even we pre-train on extremely larger classification dataset (*e.g.*, Instagram (Mahajan et al., 2018), which is 3000× larger than ImageNet), the transfer improvements on object detection are relatively small. Recent works (Li et al., 2019a; Zhou et al., 2020) show that pre-trained models utilizing object detection data and annotations (*e.g.*, MS COCO (Lin et al., 2014)) could achieve on par performance on object detection and semantic segmentation compared with ImageNet pre-trained model. While the supervised pre-training for dense prediction tasks has been explored before DenseCL, there are few works on designing an unsupervised paradigm for dense prediction tasks. Concurrent and independent works (Pinheiro et al., 2020a; Chaitanya et al., 2020a) also find that contrastive learning at the level of local features matters. One of the main differences is that they construct the positive pairs according to the geometric transformation, while we also propose a flexible and novel approach to construct positive pairs according to the implicit correspondence across views. Our method is totally decoupled from the data pre-processing, thus enabling fast and flexible training while being agnostic about what kind of augmentation is used and how the images are sampled.

2.4 Unsupervised Object Localization

Unsupervised object discovery A wide range of approaches have been proposed for unsupervised object discovery, including statistical topic discovery models (Sivic et al., 2005; Russell et al., 2006), link analysis technique (Kim and Torralba, 2009), clustering by composition (Faktor and Irani, 2014), and part-based matching (Cho et al., 2015). Some recent works (Vo et al., 2019; Vo, Pérez, and Ponce, 2020) formulate object discovery as an optimization problem. Large-Scale Unsupervised Object Discovery (Vo et al., 2021) further proposes to formulate unsupervised object discovery as a ranking problem. Yet, the existing methods have achieved limited success in challenging and complicated scenes. Furthermore, most of these methods can only find coarse bounding boxes of objects. By contrast, our method discovers and localizes objects in the wild with pixel-wise segmentation masks. With bounding boxes obtained from predicted masks, FreeSOLO outperforms the state-of-the-art unsupervised object discovery methods by a large margin.

Unsupervised segmentation To remove the dependency on manual supervision, some object co-segmentation works (Joulin, Bach, and Ponce, 2010; Hsu, Lin, and Chuang, 2018; Chen et al., 2021) make a strong assumption about the image collection,

i.e., to segment common repeated objects in a collection of images. Besides, there are a few works (Ji, Vedaldi, and Henriques, 2019; Hwang et al., 2019; Gansbeke et al., 2021) that explore unsupervised semantic segmentation. Some (Ji, Vedaldi, and Henriques, 2019) only deal with simple scenarios, and some (Hwang et al., 2019; Gansbeke et al., 2021) still require a salient object estimator or boundary annotations. In addition, the key difference lies in the task. Instead of semantic segmentation, our method solves the harder problem of instance segmentation, *i.e.*, to segment each object individually.

Statement of Authorship

Title of Paper	SOLO: Segmenting Objects by Locations
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)	Xinlong Wang		
Contribution to the Paper	Proposed the ideas, conducted experiments, and wrote the manuscript.		
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	27/05/2022

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	Tao Kong		
Contribution to the Paper	Discussion, writing revision and conducting some experiments.		
Signature		Date	28/05/2022

Name of Co-Author	Chunhua Shen		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	28/05/2022

Name of Co-Author	Yuning Jiang		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	6/1/2022

Name of Co-Author	Lei Li		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	5/28/2022

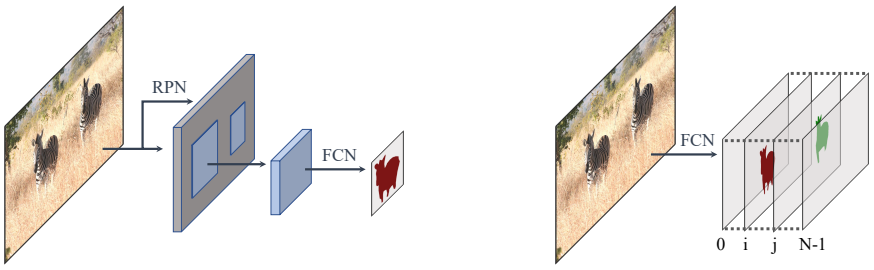
Chapter 3

SOLO: Segmenting Objects by Locations

3.1 Introduction

Instance segmentation is challenging because it requires the correct separation of all objects in an image while also semantically segmenting each instance at the pixel level. Objects in an image belong to a fixed set of semantic categories, but the number of instances varies. As a result, semantic segmentation can be easily formulated as a dense per-pixel classification problem, while it is challenging to predict instance labels directly following the same paradigm.

To overcome this obstacle, recent instance segmentation methods can be categorized into two groups, *i.e.*, top-down and bottom-up paradigms. The former approach, namely 'detect-then-segment', first detects bounding boxes and then segments the instance mask in each bounding box. The latter approach learns an affinity relation, assigning an embedding vector to each pixel, by pushing away pixels belonging to different instances and pulling close pixels in the same instance. A grouping post-processing is then needed to separate instances. Both these two paradigms are step-wise and *indirect*, which either heavily rely on accurate bounding box detection or depend on per-pixel embedding learning and the grouping processing.



(a) Mask R-CNN (b) SOLO

FIGURE 3.1. Comparison of the pipelines of Mask R-CNN and the proposed SOLO.

In contrast, we aim to directly segment instance masks, under the supervision of full instance mask annotations instead of masks in boxes or additional pixel pairwise relations. We start by rethinking a question: *What are the fundamental differences between object instances in an image?* Take the challenging MS COCO dataset (Lin et al., 2014) for example. There are in total 36,780 objects in the validation subset, 98.3% of object pairs have center distance greater than 30 pixels. As for the rest 1.7% of object pairs, 40.5% of them have size ratio greater than 1.5×. To conclude, in most cases two instances in an image either have different center locations or have different object sizes. This observation makes one wonder whether we could directly distinguish instances by the center locations and object sizes?

In the closely related field, semantic segmentation, now the dominate paradigm leverages a fully convolutional network (FCN) to output dense predictions with N channels. Each output channel is responsible for one of the semantic categories (including background). Semantic segmentation aims to distinguish different semantic categories. Analogously, in this work, we propose to distinguish object instances in the image by introducing the notion of "*instance categories*", *i.e.*, the quantized center locations and object sizes, which enables to **s**egment **o**bjects by **lo**cations, thus the name of our method, **SOLO.**

Locations An image can be divided into a grid of $S \times S$ cells, thus leading to S^2 center location classes. According to the coordinates of the object center, an object instance is assigned to one of the grid cells, as its center location category. Note that grids are used conceptually to assign location category for each pixel. Each output channel is responsible for one of the center location categories, and the corresponding channel map should predict the instance mask of the object belonging to that location. Thus, structural geometric information is naturally preserved in the spatial matrix with dimensions of height by width. Unlike DeepMask (Pinheiro, Collobert, and Dollár, 2015) and TensorMask (Chen et al., 2019b), which run in a dense sliding-window manner and segment an object in a fixed local patch, our method naturally outputs accurate masks for all scales of instances without the limitation of (anchor) box locations and scales.

In essence, an instance location category approximates the location of the object center of an instance. Thus, by classification of each pixel into its instance location category, it is equivalent to predict the object center of each pixel in the latent space. The importance here of converting the location prediction task into classification is that, with classification it is much more straightforward and easier to model varying number of instances using a fixed number of channels, at the same time not relying on post-processing like grouping or learning embeddings.

Sizes To distinguish instances with different object sizes, we employ the feature pyramid network (FPN) (Lin et al., 2017a), so as to assign objects of different sizes to different levels of feature maps. Thus, all the object instances are separated regularly,

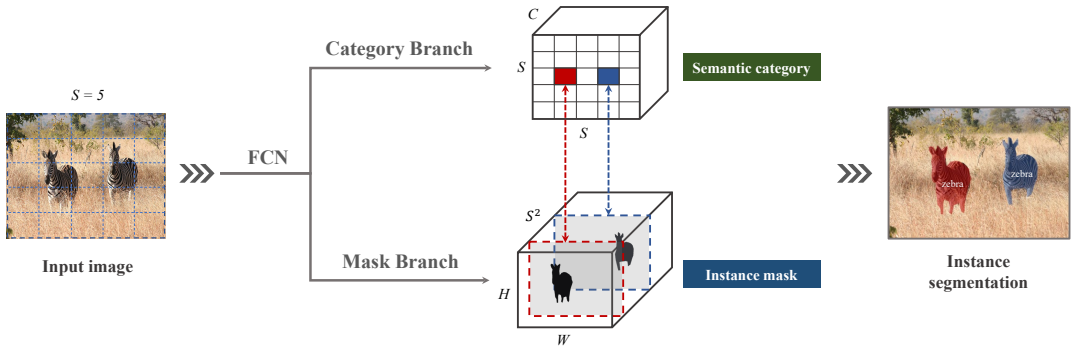


FIGURE 3.2. **SOLO framework.** We reformulate the instance segmentation as two sub-tasks: category prediction and instance mask generation problems. An input image is divided into a uniform grids, *i.e.*, $S \times S$. Here we illustrate the grid with $S = 5$. If the center of an object falls into a grid cell, that grid cell is responsible for predicting the semantic category (top) and masks of instances (bottom). We do not show the feature pyramid network (FPN) here for simpler illustration.

enabling to classify objects by "instance categories". Note that FPN was designed for the purposes of detecting objects of different sizes in an image.

In the sequel, we empirically show that FPN is one of the core components for our method and has a profound impact on the segmentation performance, especially objects of varying sizes being presented.

With the proposed SOLO framework, we are able to optimize the network in an end-to-end fashion for the instance segmentation task using mask annotations solely, and perform pixel-level instance segmentation out of the restrictions of local box detection and pixel grouping. For the first time, we demonstrate a very simple instance segmentation approach achieving *on par* results to the dominant "detect-then-segment" method on the challenging COCO dataset (Lin et al., 2014) with diverse scenes and semantic classes. Additionally, we showcase the generality of our framework via the task of instance contour detection, by viewing the instance edge contours as a one-hot binary mask, with almost no modification SOLO can generate reasonable instance contours. The proposed SOLO only needs to solve two pixel-level classification tasks, thus it may be possible to borrow some of the recent advances in semantic segmentation for improving SOLO. *The embarrassing simplicity and strong performance of the proposed SOLO method may predict its application to a wide range of instance-level recognition tasks.*

3.2 Our Method: SOLO

3.2.1 Problem Formulation

The central idea of SOLO framework is to reformulate the instance segmentation as two simultaneous category-aware prediction problems. Concretely, our system divides

the input image into a uniform grids, *i.e.*, $S \times S$. If the center of an object falls into a grid cell, that grid cell is responsible for 1) predicting the semantic category as well as 2) segmenting that object instance.

Semantic Category

For each grid, our SOLO predicts the C-dimensional output to indicate the semantic class probabilities, where C is the number of classes. These probabilities are conditioned on the grid cell. If we divide the input image into $S \times S$ grids, the output space will be $S \times S \times C$, as shown in Figure 3.2 (top). This design is based on the assumption that each cell of the $S \times S$ grid must belong to one individual instance, thus only belonging to one semantic category. During inference, the C-dimensional output indicates the class probability for each object instance.

Instance Mask

In parallel with the semantic category prediction, each positive grid cell will also generate the corresponding instance mask. For an input image I, if we divide it into $S \times S$ grids, there will be at most S^2 predicted masks in total. We explicitly encode these masks at the third dimension (channel) of a 3D output tensor. Specifically, the instance mask output will have $H_I \times W_I \times S^2$ dimension. The k^{th} channel will be responsible to segment instance at grid (i, j), where $k = i \cdot S + j$ (with i and j zero-based)[1]. To this end, a one-to-one correspondence is established between the semantic category and class-agnostic mask (Figure 3.2).

A direct approach to predict the instance mask is to adopt the fully convolutional networks, like FCNs in semantic segmentation (Long, Shelhamer, and Darrell, 2015a). However the conventional convolutional operations are *spatially invariant* to some degree. Spatial invariance is desirable for some tasks such as image classification as it introduces robustness. However, here we need a model that is *spatially variant*, or in more precise words, position sensitive, since our segmentation masks are conditioned on the grid cells and must be separated by different feature channels.

Our solution is very simple: at the beginning of the network, we directly feed normalized pixel coordinates to the networks, inspired by 'CoordConv' operator (Liu et al., 2018a). Specifically, we create a tensor of same spatial size as input that contains pixel coordinates, which are normalized to $[-1, 1]$. This tensor is then concatenated to the input features and passed to the following layers. By simply giving the convolution access to its own input coordinates, we add the spatial functionality to the conventional FCN model. It should be noted that CoordConv is not the only choice. For example the semi-convolutional operators (Novotný et al., 2018) may be competent, but we employ CoordConv for its simplicity and being easy to implement. If the original feature tensor is of size $H \times W \times D$, the size of new tensor becomes $H \times W \times (D + 2)$,

[1] We also show an equivalent and more efficient implementation in Section 3.4.

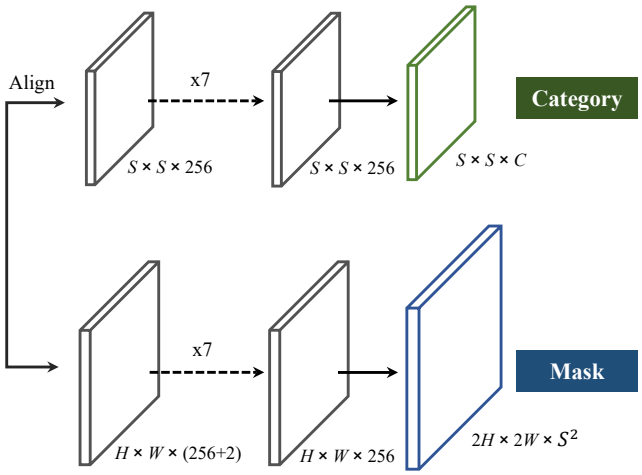


FIGURE 3.3. **SOLO Head architecture**. At each FPN feature level, we attach two sibling sub-networks, one for instance category prediction (top) and one for instance mask segmentation (bottom). In the mask branch, we concatenate the x, y coordinates and the original features to encode spatial information. Here numbers denote spatial resolution and channels. In the figure, we assume 256 channels as an example. Arrows denote either convolution or interpolation. 'Align' means bilinear interpolation. '$\times 7$' means 7 convolutions. '$2H \times 2W$' is for predicting masks at higher resolutions. During inference, the mask branch outputs are further upsampled to the original image size.

in which the last two channels are x-y pixel coordinates. For more information on CoordConv, we refer readers to Liu et al. (2018a).

Forming Instance Segmentation

In SOLO, the category prediction and the corresponding mask are naturally associated by their reference grid cell, *i.e.*, $k = i \cdot S + j$. Based on this, we can directly form the final instance segmentation result for each grid. The raw instance segmentation results are generated by gathering all grid results. Finally, non-maximum-suppression (NMS) is used across the channels (masks) to obtain the final instance segmentation results. No other post processing operations are needed.

3.2.2 Network Architecture

SOLO attaches to a convolutional backbone. We use FPN (Lin et al., 2017a), which generates a pyramid of feature maps with different sizes with a fixed number of channels (usually 256-d) for each level. These maps are used as input for each prediction head: semantic category and instance mask. Weights for the head are shared across different levels. Grid number may varies at different pyramids. Only the last conv is not shared in this scenario.

To demonstrate the generality and effectiveness of our approach, we instantiate SOLO with multiple architectures. The differences include: (a) the *backbone* architecture

used for feature extraction, (b) the network *head* for computing the instance segmentation results, and (c) training *loss function* used to optimize the model. Most of the experiments are based on the *head* architecture as shown in Figure 3.3. We also utilize different variants to further study the generality. We note that our instance segmentation heads have a straightforward structure. More complex designs have the potential to improve performance but are not the focus of this work.

3.2.3 SOLO Learning

Label Assignment

For category prediction branch, the network needs to give the object category probability for each of $S \times S$ grid. Specifically, grid (i, j) is considered as a positive sample if it falls into the *center region* of any ground truth mask, Otherwise it is a negative sample. Center sampling is effective in recent works of object detection (Tian et al., 2019; Kong et al., 2020), and here we also utilize a similar technique for mask category classification. Given the mass center (c_x, c_y), width w, and height h of the ground truth mask, the center region is controlled by constant scale factors ϵ: $(c_x, c_y, \epsilon w, \epsilon h)$. We set $\epsilon = 0.2$ and there are on average 3 positive samples for each ground truth mask.

Besides the label for instance category, we also have a binary segmentation mask for each positive sample. Since there are S^2 grids, we also have S^2 output masks for each image. For each positive samples, the corresponding target binary mask will be annotated. One may be concerned that the order of masks will impact the mask prediction branch, however, we show that the most simple row-major order works well for our method.

Loss Function

We define our training loss function as follows:

$$L = L_{cate} + \lambda L_{mask}, \tag{3.1}$$

where L_{cate} is the conventional Focal Loss (Lin et al., 2017b) for semantic category classification. L_{mask} is the loss for mask prediction:

$$L_{mask} = \frac{1}{N_{pos}} \sum_{k} \mathbb{1}_{\{\mathbf{p}^*_{i,j} > 0\}} d_{mask}(\mathbf{m}_k, \mathbf{m}^*_k), \tag{3.2}$$

Here indices $i = \lfloor k/S \rfloor, j = k \bmod S$, if we index the grid cells (instance category labels) from left to right and top to down. N_{pos} denotes the number of positive samples, $\mathbf{p}^*$ and $\mathbf{m}^*$ represent category and mask target respectively. $\mathbb{1}$ is the indicator function, being 1 if $\mathbf{p}^*_{i,j} > 0$ and 0 otherwise.

We have compared different implementations of $d_{mask}(\cdot,\cdot)$: Binary Cross Entropy (BCE), Focal Loss (Lin et al., 2017b) and Dice Loss (Milletari, Navab, and Ahmadi, 2016). Finally, we employ Dice Loss for its effectiveness and stability in training. λ in Equation (3.1) is set to 3. The Dice Loss is defined as

$$L_{Dice} = 1 - D(\mathbf{p}, \mathbf{q}), \tag{3.3}$$

where D is the dice coefficient which is defined as

$$D(\mathbf{p}, \mathbf{q}) = \frac{2\sum_{x,y}(\mathbf{p}_{x,y} \cdot \mathbf{q}_{x,y})}{\sum_{x,y}\mathbf{p}_{x,y}^2 + \sum_{x,y}\mathbf{q}_{x,y}^2}. \tag{3.4}$$

Here $\mathbf{p}_{x,y}$ and $\mathbf{q}_{x,y}$ refer to the value of pixel located at (x, y) in predicted soft mask $\mathbf{p}$ and ground truth mask $\mathbf{q}$.

3.2.4 Inference

The inference of SOLO is very straightforward. Given an input image, we forward it through the backbone network and FPN, and obtain the category score $\mathbf{p}_{i,j}$ at grid (i, j) and the corresponding masks $\mathbf{m}_k$, where $k = i \cdot S + j$. We first use a confidence threshold of 0.1 to filter out predictions with low confidence. Then we select the top 500 scoring masks and feed them into the NMS operation. We use a threshold of 0.5 to convert predicted soft masks to binary masks.

Maskness. We calculate maskness for each predicted mask, which represents the quality and confidence of mask prediction $\text{maskness} = \frac{1}{\mathrm{N_f}}\sum_{\mathrm{i}}^{\mathrm{N_f}} \mathbf{p_i}$. Here N_f the number of foreground pixels of the predicted soft mask $\mathbf{p}$, *i.e.*, the pixels that have values greater than threshold 0.5. The classification score for each prediction is multiplied by the maskness as the final confidence score.

3.3 Experiments

We present experimental results on the MS COCO instance segmentation benchmark (Lin et al., 2014), and report ablation studies by evaluating on the 5k `val2017` split. For our main results, we report COCO mask AP on the `test-dev` split, which has no public labels and is evaluated on the evaluation server.

Training details. SOLO is trained with stochastic gradient descent (SGD). We use synchronized SGD over 8 GPUs with a total of 16 images per mini-batch. Unless otherwise specified, all models are trained for 36 epochs with an initial learning rate of 0.01, which is then divided by 10 at 27th and again at 33th epoch. Weight decay of 0.0001 and momentum of 0.9 are used. All models are initialized from ImageNet pre-trained weights. We use scale jitter where the shorter image side is randomly sampled from 640 to 800 pixels, following (Chen et al., 2019b).

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
two-stage:							
MNC (Dai, He, and Sun, 2016)	Res-101-C4	24.6	44.3	24.8	4.7	25.9	43.6
FCIS (Li et al., 2017)	Res-101-C5	29.2	49.5	–	7.1	31.3	50.0
Mask R-CNN (He et al., 2017b)	Res-101-FPN	35.7	58.0	37.8	15.5	38.1	52.4
MaskLab+ (Chen et al., 2018a)	Res-101-C4	37.3	59.8	39.6	16.9	39.9	53.5
Mask R-CNN*	Res-101-FPN	37.8	59.8	40.7	20.5	40.4	49.3
one-stage:							
TensorMask (Chen et al., 2019b)	Res-50-FPN	35.4	57.2	37.3	16.3	36.8	49.3
TensorMask (Chen et al., 2019b)	Res-101-FPN	37.1	59.3	39.4	17.4	39.1	51.6
YOLACT (Bolya et al., 2019)	Res-101-FPN	31.2	50.6	32.8	12.1	33.3	47.1
PolarMask (Xie et al., 2020a)	Res-101-FPN	30.4	51.9	31.0	13.4	32.4	42.8
ours:							
SOLO	Res-50-FPN	36.8	58.6	39.0	15.9	39.5	52.1
SOLO	Res-101-FPN	37.8	59.5	40.4	16.4	40.6	54.2
D-SOLO	Res-101-FPN	38.4	59.6	41.1	16.8	41.5	54.6
D-SOLO	Res-DCN-101-FPN	40.5	62.4	43.7	17.7	43.6	59.3

TABLE 3.1. **Instance segmentation mask AP** (%) on the COCO `test-dev`. All entries are *single-model* results. Here we adopt the "6×" schedule (72 epochs), following Chen et al. (2019b). Mask R-CNN* is our improved version with scale augmentation and longer training time. D-SOLO means Decoupled SOLO as introduced in Section 3.4.

3.3.1 Main Results

We compare SOLO to the state-of-the-art methods in instance segmentation on MS COCO `test-dev` in Table 3.1. SOLO with ResNet-101 achieves a mask AP of 37.8%, the state of the art among existing *two-stage* instance segmentation methods such as Mask R-CNN. SOLO outperforms all previous *one-stage* methods, including TensorMask (Chen et al., 2019b). Flops of SOLO with ResNet-101 averaged over COCO `val2017` split is 422G. Some SOLO outputs are visualized in Figure 3.6.

3.3.2 How SOLO Works?

We show the network outputs generated by $S = 12$ grids (Figure 3.4). The sub-figure (i, j) indicates the soft mask prediction results generated by the corresponding mask channel. Here we can see that different instances activates at different mask prediction channels. By explicitly segmenting instances at different positions, SOLO converts the instance segmentation problem into a position-aware classification task. Only one instance will be activated at each grid, and one instance may be predicted by multiple adjacent mask channels. During inference, we use NMS to suppress these redundant masks.

3.3.3 Ablation Experiments

Grid number. We compare the impacts of grid number on the performance with single output feature map as shown in Table 3.2. The feature is generated by merging C3, C4, and C5 outputs in ResNet (stride: 8). To our surprise, $S = 12$ can already

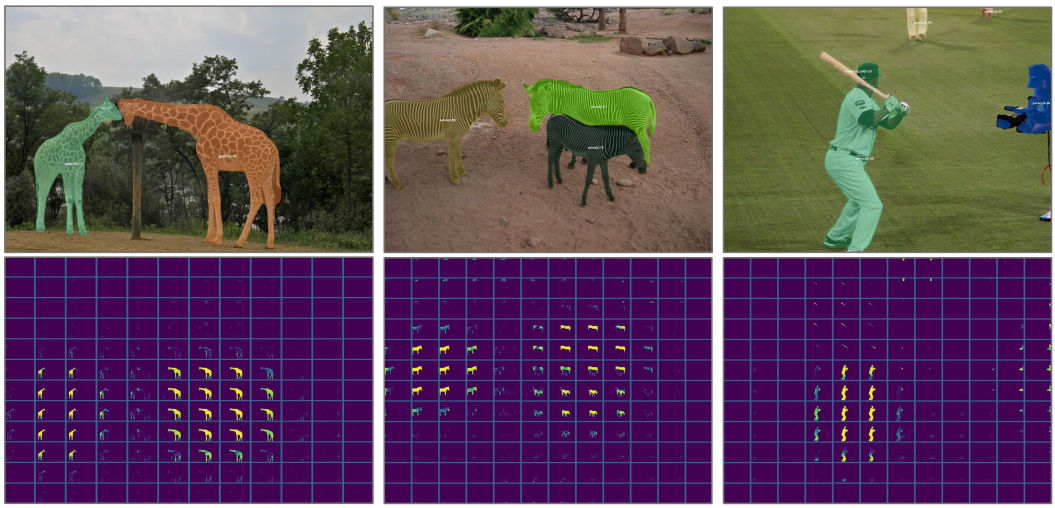


FIGURE 3.4. **SOLO behavior.** We show the visualization of soft mask prediction before NMS. Here $S = 12$. For each column, the top one is the instance segmentation result, and the bottom one shows the mask activation maps. The sub-figure (i, j) in an activation map indicates the mask prediction results (after zooming out) generated by the corresponding mask channel.

grid number	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
12	27.2	44.9	27.6	8.7	27.6	44.5
24	29.0	47.3	29.9	10.0	30.1	45.8
36	28.6	46.3	29.7	9.5	29.5	45.2
Pyramid	35.8	57.1	37.8	15.0	38.7	53.6

TABLE 3.2. The impact of **grid number and FPN**. FPN significantly improves the performance thanks to its ability to deal with varying sizes of objects.

achieve 27.2% AP on the challenging MS COCO dataset. SOLO achieves 29% AP when improving the grid number to 24. This results indicate that our single-scale SOLO can be applicable to some scenarios where object scales do not vary much.

Multi-level Prediction. From Table 3.2 we can see that our single-scale SOLO could already get 29.0 AP on MS COCO dataset. In this ablation, we show that the performance could be further improved via multi-level prediction using FPN (Lin et al., 2017a). We use five pyramids to segment objects of different scales (details in supplementary). Scales of ground-truth masks are explicitly used to assign them to the levels of the pyramid. From P2 to P6, the corresponding grid numbers are $[40, 36, 24, 16, 12]$ respectively. Based on our multi-level prediction, we further achieve 35.8 AP. As expected, the performance over all the metrics has been largely improved.

CoordConv. Another important component that facilitates our SOLO paradigm is the *spatially variant* convolution (CoordConv (Liu et al., 2018a)). As shown in Table 3.3, the standard convolution can already have spatial variant property to some extent, which is in accordance with the observation in Liu et al. (2018a). As also

revealed in Islam*, Jia*, and Bruce (2020), CNNs can implicitly learn the absolute position information from the commonly used zero-padding operation. However, the implicitly learned position information is coarse and inaccurate. When making the convolution access to its own input coordinates through concatenating extra coordinate channels, our method enjoys 3.6 absolute AP gains. Two or more CoordConvs do not bring noticeable improvement. It suggests that a single CoordConv already enables the predictions to be well spatially variant/position sensitive.

#CoordConv	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
0	32.2	52.6	33.7	11.5	34.3	51.6
1	35.8	57.1	37.8	15.0	38.7	53.6
2	35.7	57.0	37.7	14.9	38.7	53.3
3	35.8	57.4	37.7	15.7	39.0	53.0

TABLE 3.3. **Conv vs. CoordConv.** CoordConv can considerably improve AP upon standard convolution. Two or more layers of CoordConv are not necessary.

Loss function. Table 3.4 compares different loss functions for our mask optimization branch. The methods include conventional Binary Cross Entropy (BCE), Focal Loss (FL), and Dice Loss (DL). To obtain improved performance, for Binary Cross Entropy we set a mask loss weight of 10 and a pixel weight of 2 for positive samples. The mask loss weight of Focal Loss is set to 20. As shown, the Focal Loss works much better than ordinary Binary Cross Entropy loss. It is because that the majority of pixels of an instance mask are in background, and the Focal Loss is designed to mitigate the sample imbalance problem by decreasing the loss of well-classified samples. However, the Dice Loss achieves the best results without the need of manually adjusting the loss hyper-parameters. Dice Loss views the pixels as a whole object and could establish the right balance between foreground and background pixels automatically. Note that with carefully tuning the balance hyper-parameters and introducing other training tricks, the results of Binary Cross Entropy and Focal Loss may be considerably improved. However the point here is that with the Dice Loss, training typically becomes much more stable and more likely to attain good results without using much heuristics. To make a fair comparison, we also show the results of Mask R-CNN with Dice loss in the supplementary, which performs worse (-0.9AP) than original BCE loss.

Alignment in the category branch. In the category prediction branch, we must match the convolutional features with spatial size $H \times W$ to $S \times S$. Here, we compare three common implementations: interpolation, adaptive-pool, and region-grid-interpolation. (a) Interpolation: directly bilinear interpolating to the target grid size; (b) Adaptive-pool: applying a 2D adaptive max-pool over $H \times W$ to $S \times S$; (c) Region-grid-interpolation: for each grid cell, we use bilinear interpolation conditioned on dense sample points, and aggregate the results with average. From our observation, there is no noticeable performance gap between these variants ($\pm$ 0.1AP), indicating that the alignment process does not have a significant impact on the final accuracy.

mask loss	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
BCE	30.0	50.4	31.0	10.1	32.5	47.7
FL	31.6	51.1	33.3	9.9	34.9	49.8
DL	35.8	57.1	37.8	15.0	38.7	53.6

TABLE 3.4. **Different loss functions** may be employed in the mask branch. The Dice loss (DL) leads to best AP and is more stable to train.

Different head depth. In SOLO, instance segmentation is formulated as a pixel-to-pixel task and we exploit the spatial layout of masks by using an FCN. In Table 3.5, we compare different head depth used in our work. Changing the head depth from 4 to 7 gives 1.2 AP gains. The results show that when the depth grows beyond 7, the performance becomes stable. In this work, we use depth being 7 in other experiments.

head depth	4	5	6	7	8
AP	34.6	35.2	35.5	35.8	35.8

TABLE 3.5. **Different head depth.** We use depth being 7 in other experiments, as the performance becomes stable when the depth grows beyond 7.

Previous works (*e.g.*, Mask R-CNN) usually adopt four conv layers for mask prediction. In SOLO, the mask is conditioned on the spatial position and we simply attach the coordinate to the beginning of the head. The mask head must have enough representation power to learn such transformation. For the semantic category branch, the computational overhead is negligible since $S^2 \ll H \times W$.

3.3.4 SOLO-512

Speed-wise, the Res-101-FPN SOLO runs at 10.4 FPS on a V100 GPU (all post-processing included), vs. TensorMask's 2.6 FPS and Mask R-CNN's 11.1 FPS. We also train a smaller version of SOLO designed to speed up the inference. We use a model with smaller input resolution (shorter image size of 512 instead of 800). Other training and testing parameters are the same between SOLO-512 and SOLO.

	backbone	AP	AP_{50}	AP_{75}	fps
SOLO	ResNet-50-FPN	36.0	57.5	38.0	12.1
SOLO	ResNet-101-FPN	37.1	58.7	39.4	10.4
SOLO-512	ResNet-50-FPN	34.2	55.9	36.0	22.5
SOLO-512	ResNet-101-FPN	35.0	57.1	37.0	19.2

TABLE 3.6. **SOLO-512**. SOLO-512 uses a model with smaller input size. All models are evaluated on `val2017`. Here the models are trained with "6×" schedule.

With 34.2 mask AP, SOLO-512 achieves a model inference speed of 22.5 FPS, showing that SOLO has potentiality for real-time instance segmentation applications. The speed is reported on a single V100 GPU by averaging 5 runs.

3.3.5 Error Analysis

To quantitatively understand SOLO for mask prediction, we perform an error analysis by replacing the predicted masks with ground-truth values. For each predicted binary mask, we compute IoUs with ground-truth masks, and replace it with the most overlapping ground-truth mask. As reported in Table 3.7, if we replace the predicted masks with ground-truth masks, the AP increases to 68.1%. This experiment suggests that there are still ample room for improving the mask branch. We expect techniques developed (a) in semantic segmentation, and (b) for dealing occluded/tiny objects could be applied to boost the performance.

	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
baseline	37.1	58.7	39.4	16.0	41.1	54.2
w/gt mask	68.1	68.3	68.2	46.1	75.0	78.5

TABLE 3.7. **Error analysis**. Replacing the predicted instance mask with the ground-truth ones improves the mask AP from 37.1 to 68.1, suggesting that the mask branch still has ample room to be improved. The models are based on ResNet-101-FPN.

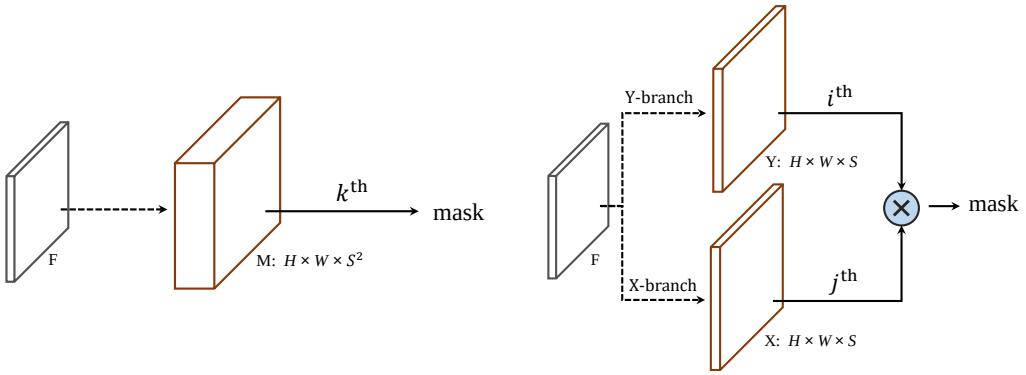
3.4 Decoupled SOLO

Given an predefined grid number, *e.g.*, $S = 20$, our SOLO head outputs $S^2 = 400$ channel maps. However, the prediction is somewhat redundant as in most cases the objects are located sparsely in the image. In this section, we further introduce an equivalent and significantly more efficient variant of the vanilla SOLO, termed **Decoupled SOLO**, shown in Figure 3.5.

In Decoupled SOLO, the original output tensor $M \in \mathbb{R}^{H \times W \times S^2}$ is replaced with two output tensors $X \in \mathbb{R}^{H \times W \times S}$ and $Y \in \mathbb{R}^{H \times W \times S}$, corresponding two axes respectively. Thus, the output space is decreased from $H \times W \times S^2$ to $H \times W \times 2S$. For an object located at grid location (i, j), the mask prediction of that object is defined as the element-wise multiplication of two channel maps:

$$\mathbf{m}_k = \mathbf{x}_j \otimes \mathbf{y}_i, \tag{3.5}$$

where $\mathbf{x}_j$ and $\mathbf{y}_i$ are the j^{th} and i^{th} channel map of X and Y after `sigmoid` operation. The motivation behind this is that the probability of a pixel belonging to location category (i, j) is the joint probability of belonging to i^{th} row and j^{th} column, as the horizontal and vertical location categories are independent.



(a) Vanilla head (b) Decoupled head

FIGURE 3.5. **Decoupled SOLO head**. F is input feature. Dashed arrows denote convolutions. $k = i \cdot S + j$. '$\otimes$' denotes element-wise multiplication.

We conduct experiments using the the same hyper-parameters as vanilla SOLO. As shown in Table 3.1, Decoupled SOLO even achieves slightly better performance (0.6 AP gains) than vanilla SOLO. With DCN-101 (Dai et al., 2017a) backbone, we further achieve 40.5 AP, which is considerably better than current dominant approaches. It indicates that the Decoupled SOLO serves as an efficient and equivalent variant in accuracy of SOLO. Note that, as the output space is largely reduced, the Decoupled SOLO needs considerably less GPU memory during training and testing.

FIGURE 3.6. **Visualization of instance segmentation results** using the Res-101-FPN backbone. The model is trained on the COCO `train2017` dataset, achieving a mask AP of 37.8 on the COCO `test-dev`.

3.5 Conclusion

In this work we have developed a direct instance segmentation framework, termed SOLO. Our SOLO is end-to-end trainable and can directly map a raw input image to the desired instance masks with constant inference time, eliminating the need for the grouping post-processing as in bottom-up methods or the bounding-box detection and

RoI operations in top-down approaches. Given the simplicity, flexibility, and strong performance of SOLO, we hope that our SOLO can serve as a cornerstone for many instance-level recognition tasks.

Statement of Authorship

Title of Paper	SOLOv2: Dynamic and Fast Instance Segmentation
Publication Status	☑ Published ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and Unsubmitted work written in manuscript style
Publication Details	Published in NeurIPS 2020

Principal Author

Name of Principal Author (Candidate)	Xinlong Wang		
Contribution to the Paper	Proposed the ideas, conducted experiments, and wrote the manuscript.		
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	27/05/2022

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	Rufeng Zhang		
Contribution to the Paper	Discussion, writing revision and conducting some experiments.		
Signature		Date	May 28, 2022

Name of Co-Author	Tao Kong		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	28/05/2022

Name of Co-Author	Lei Li		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	5/28/2022

Name of Co-Author	Chunhua Shen		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	28/05/2022

Statement of Authorship

Title of Paper	SOLO: A Simple Framework for Instance Segmentation
Publication Status	☑ Published ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and Unsubmitted work written in manuscript style
Publication Details	Published in TPAMI 2021

Principal Author

Name of Principal Author (Candidate)	Xinlong Wang
Contribution to the Paper	Proposed the ideas, conducted experiments, and wrote the manuscript.
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: 27/05/2022

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	Rufeng Zhang
Contribution to the Paper	Discussion, writing revision and conducting some experiments.
Signature	Date: May 28, 2022

Name of Co-Author	Chunhua Shen
Contribution to the Paper	Discussion and writing revision.
Signature	Date: 28/05/2022

Name of Co-Author	Tao Kong
Contribution to the Paper	Discussion and writing revision.
Signature	Date: 28/05/2022

Name of Co-Author	Lei Li			
Contribution to the Paper	Discussion and writing revision.			
Signature			Date	5/28/2022

Chapter 4

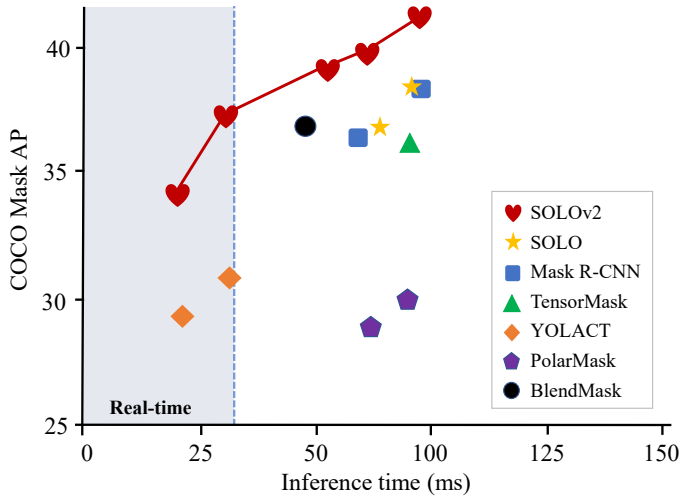
SOLOv2: Dynamic and Fast Instance Segmentation

4.1 Introduction

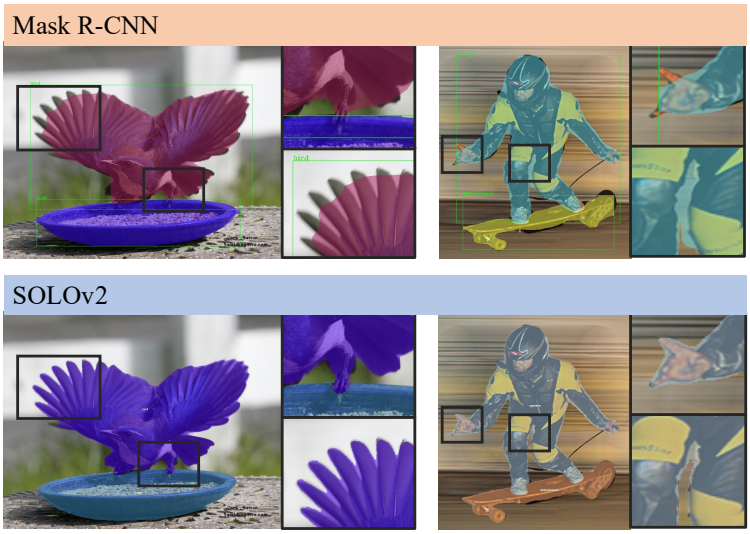
For representing the object locations, bounding box stands out for its simplicity. Localizing objects using bounding boxes has been extensively explored, including the problem formulation, network architecture, post-processing and all those focusing on optimizing and processing the bounding boxes. The tailored solutions largely boost the performance and efficiency, thus enabling wide downstream applications recently. However, bounding boxes are coarse and unnatural. Human vision can effortlessly localize objects by their irregular boundaries. Instance segmentation, *i.e.*, localizing objects using masks, pushes object localization to the limit at pixel level and opens up opportunities to more instance-level perception and applications. To date, most existing methods deal with instance segmentation in the view of bounding boxes, *i.e.*, segmenting objects in (anchor) bounding boxes. How to develop direct, efficient and strong instance segmentation including the supporting facilities, *e.g.*, post-processing, is largely unexplored compared to bounding box detection and instance segmentation methods built on top of it.

SOLO has taken the first step to build an embarrassingly simple and direct instance segmentation system. SOLO formulates the task of instance segmentation as two sub-tasks of pixel-level classification, solvable using standard FCNs, thus dramatically simplifying the formulation of instance segmentation. It takes an image as input, directly outputs instance masks and corresponding class probabilities, in a fully convolutional, box-free and grouping-free paradigm. However, three main bottlenecks limit the performance of SOLO: a) inefficient mask representation and learning; b) not high enough resolution for finer mask predictions; c) slow mask NMS. In this work, we eliminate the above bottlenecks all at once.

We first introduce a dynamic scheme, which enables dynamically segmenting objects by locations. Specifically, the mask learning process can be divided into two parts: convolution kernel learning and feature learning (Figure 4.2(b)). When classifying the



(a) Accuracy vs. Speed



(b) Segmentation Detail Comparison

FIGURE 4.1. Comparison of instance segmentation performance by SOLOv2 and other methods on the COCO `test-dev`. (a) The proposed SOLOv2 outperforms a range of state-of-the-art algorithms. All methods are evaluated using one Tesla V100 GPU. (b) SOLOv2 obtains higher-quality masks compared with Mask R-CNN. Mask R-CNN's mask head is typically restricted to 28×28 resolution, leading to inferior prediction at object boundaries.

pixels into different location categories, the mask kernels are predicted dynamically by the network and conditioned on the input. We further construct a unified and high-resolution mask feature representation for instance-aware segmentation. As such, we are able to predict high-resolution object masks, as well as learning the mask kernels and mask features separately and efficiently.

We further propose an efficient and effective matrix NMS algorithm. As a post-processing step for suppressing the duplicate predictions, non-maximum suppression

(NMS) serves as an integral part in state-of-the-art object detection systems. Take the widely adopted multi-class NMS for example. For each class, the predictions are sorted in descending order by confidence. Then for each prediction, it removes all other highly overlapped predictions. Such sequential and recursive operations result in non-negligible latency. For mask NMS, this drawback is further magnified. Compared to bounding box, it consumes more time to compute the IoU of each mask pair, thus leading to huge overhead. We address this problem by introducing Matrix NMS, which performs NMS with parallel matrix operations in one shot. Our Matrix NMS outperforms the existing NMS and its varieties in both accuracy and speed. As a result, *Matrix NMS processes 500 masks in less than 1 ms in simple python implementation*, and outperforms the recently proposed Fast NMS (Bolya et al., 2019) by 0.4% AP.

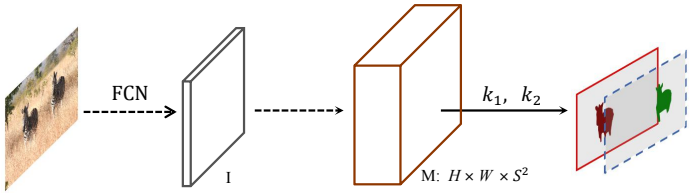
With these improvements, SOLOv2 outperforms SOLO by 1.9% AP while being 33% faster. The Res-50-FPN SOLOv2 achieves 38.8% mask AP at 18 FPS on the challenging MS COCO dataset, evaluated on a single V100 GPU card. A light-weight version of SOLOv2 executes at 31.3 FPS and yields 37.1% mask AP. Interestingly, although the concept of bounding box is thoroughly eliminated in our method, our bounding box byproduct, *i.e.*, by directly converting the predicted mask to its bounding box, yields 44.9% AP for object detection, which even *surpasses many state-of-the-art, highly-engineered object detection methods*.

We believe that, with our simple, fast and sufficiently strong solution, instance segmentation can be a popular alternative to the widely used object bounding box detection, and SOLOv2 may play an important role and predict its wide applications.

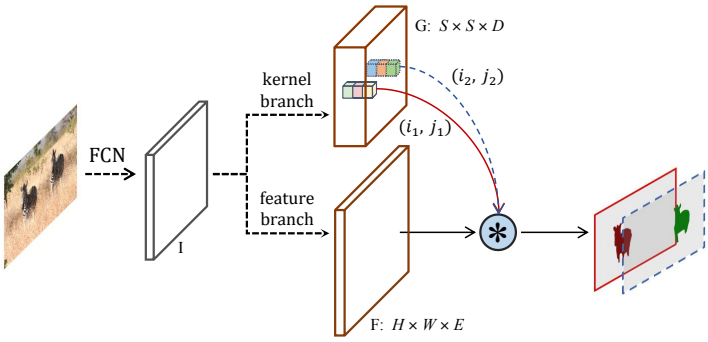
4.2 Proposed Method: SOLOv2

An instance segmentation system should separate different instances at pixel level. To distinguish instances, we follow the basic concept of 'segmenting objects by locations' introduced in Chapter 3. The input image is conceptually divided into $S \times S$ grids. If the center of an object falls into a grid cell, then the grid cell corresponds to a binary mask for that object. As such, the system outputs S^2 masks in total, denoted as $M \in \mathbb{R}^{H \times W \times S^2}$. The k^{th} channel is responsible for segmenting instance at position (i, j), where $k = i \cdot S + j$ (see Figure 4.2(a)).

Such paradigm could generate the instance segmentation results in an elegant way. However, there are three main bottlenecks that limit its performance: a) inefficient mask representation and learning. It takes a lot of memory and computation to predict the output tensor M, which has S^2 channels. Besides, as the S is different for different FPN level, the last layer of each level is learned separately and not shared, which results in an inefficient training. b) inaccurate mask predictions. Finer predictions require high-resolution masks to deal with the details at object boundaries. But large



(a) SOLO



(b) SOLOv2

FIGURE 4.2. **SOLOv2** compared to SOLO. I is the input feature after FCN-backbone representation extraction. Dashed arrows denote convolutions. $k = i \cdot S + j$; and '$\circledast$' denotes the dynamic convolution operation.

resolutions will considerably increase the computational cost. c) slow mask NMS. Compared with box NMS, mask NMS takes more time and leads to a larger overhead.

In this section, we show that these challenges can be effectively solved by our proposed dynamic mask representation and Matrix NMS, and we introduce them in the sequel.

4.2.1 Dynamic Instance Segmentation

We first briefly revisit the mask generation in SOLO. To generate the instance mask of S^2 channels corresponding to $S \times S$ grids, the last layer takes one level of pyramid features $F \in \mathbb{R}^{H \times W \times E}$ as input and at last applies a convolution layer with S^2 output channels. The operation can be written as:

$$M_{i,j} = G_{i,j} \circledast F, \tag{4.1}$$

where $G_{i,j} \in \mathbb{R}^{1 \times 1 \times E}$ is the convolution kernel, and $M_{i,j} \in \mathbb{R}^{H \times W}$ is the final mask containing only one instance whose center is at location (i, j).

In other words, we need two input F and G to generate the final mask M. Previous work explicitly output the whole M for training and inference. Note that tensor M is very large, and to directly predict M is memory and computational inefficient. In most cases the objects are located sparsely in the image. M is redundant as only a small part of S^2 kernels actually functions during a single inference.

From another perspective, if we separately learn F and G, the final M could be directly generated using the both components. In this way, we can simply pick the valid ones from predicted S^2 kernels and perform the convolution dynamically. The number of model parameters also decreases. What is more, as the predicted kernel is generated dynamically conditioned on the input, it benefits from the flexibility and adaptive nature. Additionally, each of S^2 kernels is conditioned on the location. It is in accordance with the core idea of segmenting objects by locations and goes a step further by predicting the segmenters by locations.

Mask Kernel G

Given the backbone and FPN, we predict the mask kernel G at each pyramid level. We first resize the input feature $F_I \in \mathbb{R}^{H_I \times W_I \times C}$ into shape of $S \times S \times C$. Then 4×convs and a final $3 \times 3 \times D$ conv are employed to generate the kernel G. We add the spatial functionality to F_I by giving the first convolution access to the normalized coordinates following CoordConv (Liu et al., 2018a), *i.e.*, concatenating two additional input channels which contains pixel coordinates normalized to $[-1, 1]$. Weights for the head are shared across different feature map levels.

For each grid, the kernel branch predicts the D-dimensional output to indicate predicted convolution kernel weights, where D is the number of parameters. For generating the weights of a 1×1 convolution with E input channels, D equals E. As for 3×3 convolution, D equals $9E$. These generated weights are conditioned on the locations, *i.e.*, the grid cells. If we divide the input image into $S \times S$ grids, the output space will be $S \times S \times D$, There is no activation function on the output.

Mask Feature F

Since the mask feature and mask kernel are decoupled and separately predicted, there are two ways to construct the mask feature. We can put it into the head, along with the kernel branch. It means that we predict the mask features for each FPN level. Or, to predict a unified mask feature representation for all FPN levels. We have compared the two implementations in Section 4.3.1 by experiments. Finally, we employ the latter one for its effectiveness and efficiency.

For learning a unified and high-resolution mask feature representation, we apply feature pyramid fusion inspired by the semantic segmentation in Kirillov et al. (2019a). After repeated stages of 3×3 conv, group norm (Wu and He, 2018), ReLU and 2× bilinear upsampling, the FPN features P2 to P5 are merged into a single output at $1/4$

scale. The last layer after the element-wise summation consists of 1×1 convolution, group norm and ReLU. It should be noted that we feed normalized pixel coordinates to the deepest FPN level (at $1/32$ scale), before the convolutions and bilinear upsamplings. The provided accurate position information is important for enabling position sensitivity and predicting instance-aware features.

Forming Instance Mask

For each grid cell at (i, j), we first obtain the mask kernel $G_{i,j,:} \in \mathbb{R}^D$. Then $G_{i,j,:}$ is convolved with F to get the instance mask. In total, there will be at most S^2 masks for each prediction level. Finally, we use the proposed Matrix NMS to get the final instance segmentation results.

Learning and Inference

The training loss function is defined as follows:

$$L = L_{cate} + \lambda L_{mask}, \tag{4.2}$$

where L_{cate} is the conventional Focal Loss (Lin et al., 2017b) for semantic category classification, L_{mask} is the Dice Loss for mask prediction. For more details, we refer readers to Chapter 3.

During the inference, we forward input image through the backbone network and FPN, and obtain the category score $\mathbf{p}_{i,j}$ at grid (i, j). We first use a confidence threshold of 0.1 to filter out predictions with low confidence. The corresponding predicted mask kernels are then used to perform convolution on the mask feature. After the `sigmoid` operation, we use a threshold of 0.5 to convert predicted soft masks to binary masks. The last step is the Matrix NMS.

4.2.2 Matrix NMS

Motivation. Our Matrix NMS is motivated by Soft-NMS (Bodla et al., 2017). Soft-NMS decays the other detection scores as a monotonic decreasing function $f(\texttt{iou})$ of their overlaps. By decaying the scores according to IoUs recursively, higher IoU detections will be eliminated with a minimum score threshold. However, such process is sequential like traditional Greedy NMS and could not be implemented in parallel.

Matrix NMS views this process from another perspective by considering how a predicted mask m_j being suppressed. For m_j, its decay factor is affected by: (a) The penalty of each prediction m_i on m_j $(s_i > s_j)$, where s_i and s_j are the confidence scores; and (b) the probability of m_i being suppressed. For (a), the penalty of each prediction m_i on m_j could be easily computed by $f(\texttt{iou}_{i,j})$. For (b), the probability of m_i being suppressed is not so elegant to be computed. However, the probability usually has positive correlation with the IoUs. So here we directly approximate the

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
box-based:							
Mask R-CNN (He et al., 2017b)	Res-101-FPN	35.7	58.0	37.8	15.5	38.1	52.4
Mask R-CNN*	Res-101-FPN	37.8	59.8	40.7	**20.5**	40.4	49.3
MaskLab+ (Chen et al., 2018a)	Res-101-C4	37.3	59.8	39.6	16.9	39.9	53.5
TensorMask (Chen et al., 2019b)	Res-101-FPN	37.1	59.3	39.4	17.4	39.1	51.6
YOLACT (Bolya et al., 2019)	Res-101-FPN	31.2	50.6	32.8	12.1	33.3	47.1
MEInst (Zhang et al., 2020)	Res-101-FPN	33.9	56.2	35.4	19.8	36.1	42.3
CenterMask (Wang et al., 2020c)	Hourglass-104	34.5	56.1	36.3	16.3	37.4	48.4
BlendMask (Chen et al., 2020a)	Res-101-FPN	38.4	60.7	41.3	18.2	41.5	53.3
box-free:							
PolarMask (Xie et al., 2020a)	Res-101-FPN	32.1	53.7	33.1	14.7	33.8	45.3
SOLO (Wang et al., 2020a)	Res-101-FPN	37.8	59.5	40.4	16.4	40.6	54.2
SOLOv2	Res-50-FPN	38.8	59.9	41.7	16.5	41.7	56.2
SOLOv2	Res-101-FPN	39.7	60.7	42.9	17.3	42.9	57.4
SOLOv2	Res-DCN-101-FPN	**41.7**	**63.2**	**45.1**	18.0	**45.0**	**61.6**

TABLE 4.1. **Instance segmentation** mask AP (%) on COCO `test-dev`. All entries are *single-model* results. Mask R-CNN* is our improved version with scale augmentation and longer training time (6×). 'DCN' means deformable convolutions used.

probability by the most overlapped prediction on m_i as

$$f(\texttt{iou}_{\cdot,i}) = \min_{\forall s_k > s_i} f(\texttt{iou}_{k,i}). \tag{4.3}$$

To this end, the final decay factor becomes

$$decay_j = \min_{\forall s_i > s_j} \frac{f(\texttt{iou}_{i,j})}{f(\texttt{iou}_{\cdot,i})}, \tag{4.4}$$

and the updated score is computed by $s_j = s_j \cdot decay_j$. We consider two most simple decremented functions, denoted as `linear` $f(\texttt{iou}_{i,j}) = 1 - \texttt{iou}_{i,j}$, and `Gaussian` $f(\texttt{iou}_{i,j}) = \exp\left(-\frac{\texttt{iou}_{i,j}^2}{\sigma}\right)$.

Implementation. All the operations in Matrix NMS could be implemented in one shot without recurrence. We first compute a $N \times N$ pairwise IoU matrix for the top N predictions sorted descending by score. For binary masks, the IoU matrix could be efficiently implemented by matrix operations. Then we get the most overlapping IoUs by column-wise max on the IoU matrix. Next, the decay factors of all higher scoring predictions are computed, and the decay factor for each prediction is selected as the most effect one by column-wise min (Eqn. (4.4)). Finally, the scores are updated by the decay factors. For usage, we just need thresholding and selecting top-k scoring masks as the final predictions.

In our code base, Matrix NMS is 9×faster than traditional NMS and being more accurate (Table 4.6). We show that Matrix NMS serves as a superior alternative of traditional NMS in both accuracy and speed, and can be easily integrated into the state-of-the-art detection/segmentation systems.

4.3 Experiments

To evaluate the proposed method SOLOv2, we conduct experiments on three basic tasks, instance segmentation, object detection, and panoptic segmentation on MS COCO (Lin et al., 2014). We also present experimental results on the recently proposed LVIS dataset (Gupta, Dollar, and Girshick, 2019), which has more than 1K categories and thus is considerably more challenging.

4.3.1 Instance Segmentation

For instance segmentation, we report lesion and sensitivity studies by evaluating on the COCO 5K `val2017` split. We also report COCO mask AP on the `test-dev` split, which is evaluated on the evaluation server. SOLOv2 is trained with stochastic gradient descent (SGD). We use synchronized SGD over 8 GPUs with a total of 16 images per mini-batch. Unless otherwise specified, all models are trained for 36 epochs (*i.e.*, 3×) with an initial learning rate of 0.01, which is then divided by 10 at 27th and again at 33th epoch. We use scale jitter where the shorter image side is randomly sampled from 640 to 800 pixels.

Main Results

We compare SOLOv2 to the state-of-the-art methods in instance segmentation on MS COCO `test-dev` in Table 4.1. SOLOv2 with ResNet-101 achieves a mask AP of 39.7%, which is much better than other state-of-the-art instance segmentation methods. Our method shows its superiority especially on large objects (*e.g.*, +5.0 AP_L than Mask R-CNN).

We also provide the speed-accuracy trade-off on COCO to compare with some dominant instance segmenters (Figure 4.1 (a)). We show our models with ResNet-50, ResNet-101, ResNet-DCN-101 and two light-weight versions described in Section 4.3.1. The proposed SOLOv2 outperforms a range of state-of-the-art algorithms, both in accuracy and speed. The running time is tested on our local machine, with a single V100 GPU. We download code and pre-trained models to test inference time for each model on the same machine. Further, as described in Figure 4.1 (b), SOLOv2 predicts much finer masks than Mask R-CNN which performs on the local region.

Beside the MS COCO dataset, we also demonstrate the effectiveness of SOLOv2 on LVIS dataset. Table 4.2 reports the performances on the rare (1∼10 images), common (11∼100), and frequent (> 100) subsets, as well as the overall AP. Both the reported Mask R-CNN and SOLOv2 use data resampling training strategy, following Gupta, Dollar, and Girshick (2019). Our SOLOv2 outperforms the baseline method by about 1% AP. For large-size objects (AP_L), our SOLOv2 achieves 6.7% AP improvement, which is consistent with the results on the COCO dataset.

	backbone	AP_r	AP_c	AP_f	AP_S	AP_M	AP_L	AP
Mask-RCNN	Res-50-FPN	14.5	24.3	28.4	-	-	-	24.4
Mask-RCNN*-3×	Res-50-FPN	12.1	25.8	28.1	**18.7**	31.2	38.2	24.6
SOLOv2	Res-50-FPN	13.4	26.6	28.9	15.9	34.6	44.9	25.5
SOLOv2	Res-101-FPN	**16.3**	**27.6**	**30.1**	16.8	**35.8**	**47.0**	**26.8**

TABLE 4.2. Instance segmentation results on the LVISv0.5 validation dataset. $*$ means re-implementation.

FIGURE 4.3. **Visualization of instance segmentation results** using SOLOv2 ResNet-101 backbone. The model is trained on the COCO `train2017` dataset, achieving a mask AP of 39.7% on the COCO `test-dev`.

We further show visualization of the final instance segmentation results in Figure 4.3. Different objects are in different colors. Our method shows promising results in diverse scenes. It is worth pointing out that the details at the boundaries are segmented well, especially for large objects.

Ablation Experiments

We investigate and compare the following five aspects in our methods.

Kernel shape. We consider the kernel shape from two aspects: number of input channels and kernel size. The comparisons are shown in Table 4.3. 1×1 conv shows equivalent performance to 3×3 conv. Changing the number of input channels from 128 to 256 attains 0.4% AP gains. When it grows beyond 256, the performance becomes stable. In this work, we set the number of input channels to be 256 in all other experiments.

Kernel shape	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
$3 \times 3 \times 64$	37.4	58.0	39.9	15.6	40.8	56.9
$1 \times 1 \times 64$	37.4	58.1	40.1	15.5	41.1	56.3
$1 \times 1 \times 128$	37.4	58.1	40.2	15.8	41.1	56.6
$1 \times 1 \times 256$	37.8	58.5	40.4	15.6	41.3	56.8
$1 \times 1 \times 512$	37.7	58.3	40.4	15.4	41.5	56.6

TABLE 4.3. **Kernel shape.** The performance is stable when the shape goes beyond $1 \times 1 \times 256$.

Effectiveness of coordinates. Since our method segments objects by locations, or specifically, learns the object segmenters by locations, the position information is very important. For example, if the mask kernel branch is unaware of the positions, the objects with the same appearance may have the same predicted kernel, leading to the same output mask. On the other hand, if the mask feature branch is unaware of the position information, it would not know how to assign the pixels to different feature channels in the order that matches the mask kernel. As shown in Table 4.4, the model achieves 36.3% AP without explicit coordinates input. The results are reasonably good because that CNNs can implicitly learn the absolute position information from the commonly used zero-padding operation, as revealed in Islam*, Jia*, and Bruce (2020). The pyramid zero-paddings in our mask feature branch should have contributed considerably. However, the implicitly learned position information is coarse and inaccurate. When making the convolution access to its own input coordinates through concatenating extra coordinate channels, our method enjoys 1.5% absolute AP gains.

Kernel	Feature	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
		36.3	57.4	38.6	15.6	39.8	54.7
✓		36.3	57.3	38.5	15.1	40.0	54.1
	✓	37.1	58.0	39.4	15.2	40.5	55.9
✓	✓	37.8	58.5	40.4	15.6	41.3	56.8

TABLE 4.4. **Explicit coordinates.** Precise coordinates input can considerably improve the results.

Unified mask feature representation. For mask feature learning, we have two options: to learn the feature in the head separately for each FPN level or to construct a unified representation. For the former one, we implement as SOLO and use seven 3×3 convolutions to predict the mask features. For the latter one, we fuse the

FPN's features in a simple way and obtain the unified mask representations. The detailed implementation is in supplementary material. We compare these two modes in Table 4.5. As shown, the unified representation achieves better results, especially for the medium and large objects. This is easy to understand: In separate way, the large-size objects are assigned to high-level feature maps of low spatial resolutions, leading to coarse boundary prediction.

Mask Feature	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Separate	37.3	58.2	40.0	15.7	40.8	55.5
Unified	37.8	58.5	40.4	15.6	41.3	56.8

TABLE 4.5. **Mask feature representation.** We compare the separate mask feature representation in parallel heads and the unified representation.

Matrix NMS. Our Matrix NMS can be implemented totally in parallel. Table 4.6 presents the speed and accuracy comparison of Hard-NMS, Soft-NMS, Fast NMS and our Matrix NMS. Since all methods need to compute the IoU matrix, we pre-compute the IoU matrix in advance for fair comparison. The speed reported here is that of the NMS process alone, excluding computing IoU matrices. Hard-NMS and Soft-NMS are widely used in current object detection and segmentation models. Unfortunately, both methods are recursive and spend much time budget (*e.g.*, 22 ms). Our Matrix NMS only needs <1 ms and is almost cost free! Here we also show the performance of Fast NMS, which utilizes matrix operations but with performance penalty. To conclude, our Matrix NMS shows its advantages on both speed and accuracy.

Method	Iter?	Time (ms)	AP
Hard-NMS	✓	9	36.3
Soft-NMS	✓	22	36.5
Fast NMS	✗	<1	36.2
Matrix NMS	✗	<1	36.6

TABLE 4.6. **Matrix NMS.** Matrix NMS outperforms other methods in both speed and accuracy.

Real-time setting. We design two light-weight models for different purposes. 1) `Speed priority`, the number of convolution layers in the prediction head is reduced to two and the input shorter side is 448. 2) `Accuracy priority`, the number of convolution layers in the prediction head is reduced to three and the input shorter side is 512. Moreover, deformable convolution (Dai et al., 2017b) is used in the backbone and the last layer of prediction head. We train both models with the 3× schedule, with shorter side randomly sampled from [352, 512]. Results are shown in Table 4.7. SOLOv2 can not only push state-of-the-art, but has also been ready for real-time applications.

Model	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	fps
SOLOv2-448	34.0	54.0	36.1	10.3	36.3	54.4	46.5
SOLOv2-512	37.1	57.7	39.7	12.9	40.0	57.4	31.3

TABLE 4.7. **Real-time SOLOv2**. The speed is reported on a single V100 GPU by averaging 5 runs (on COCO `test-dev`).

Training schedule. We report the results of different training schedules in Table 4.8, including 12 epochs using single-scale training (1×) and 36 epochs with multi-scale training (3×).

Schedule	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
1×	34.8	54.8	36.8	13.1	38.0	53.8
3×	37.8	58.5	40.4	15.6	41.3	56.8

TABLE 4.8. **Training schedule.** 1× means 12 epochs using single-scale training. 3× means 36 epochs with multi-scale training.

4.3.2 Extension: Object Detection

Although our instance segmentation solution removes the dependence of bounding box prediction, we are able to produce the 4D object bounding box from each instance mask. In Table S5 in the supplementary, we compare the generated box detection performance with other object detection methods on MS COCO. All models are trained on the `train2017` subset and tested on `test-dev`.

As shown in Table 4.9, our detection results outperform most methods, especially for objects of large sizes, demonstrating the effectiveness of SOLOv2 in bounding-box object detection. We also plot the speed/accuracy trade-off curve for different methods in Figure 4.4. We show our models with ResNet-101 and two light-weight versions described above. The plot reveals that the bounding box performance of SOLOv2 beats most recent object detection methods in both accuracy and speed. Here we emphasize that our results are directly generated from the off-the-shelf instance mask, without any box based training or engineering.

An observation from Figure 4.4 is as follows. If one does not care much about the annotation cost difference between mask annotation and bounding box annotation, it appears to us that there is no reason to use box detectors for downstream applications, considering the fact that our SOLOv2 beats most modern detectors in both accuracy and speed.

4.3.3 Extension: Panoptic Segmentation

We also demonstrate the effectiveness of SOLOv2 on the problem of panoptic segmentation (Kirillov et al., 2019b). The proposed SOLOv2 can be easily extended to panoptic segmentation by adding the semantic segmentation branch, analogue to

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
YOLOv3 (Redmon and Farhadi, 2018b)	DarkNet53	33.0	57.9	34.4	18.3	35.4	41.9
SSD513 (Liu et al., 2016)	ResNet-101	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 (Liu et al., 2016)	ResNet-101	33.2	53.3	35.2	13.0	35.4	51.1
RefineDet (Zhang et al., 2018)	ResNet-101	36.4	57.5	39.5	16.6	39.9	51.4
Faster R-CNN (Lin et al., 2017a)	Res-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
RetinaNet (Lin et al., 2017b)	Res-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
FoveaBox (Kong et al., 2020)	Res-101-FPN	40.6	60.1	43.5	23.3	45.2	54.5
RPDet (Yang et al., 2019)	Res-101-FPN	41.0	62.9	44.3	23.6	44.1	51.7
FCOS (Tian et al., 2019)	Res-101-FPN	41.5	60.7	45.0	24.4	44.8	51.6
SOLOv2	Res-50-FPN	40.4	59.8	42.8	20.5	44.2	53.9
SOLOv2	Res-101-FPN	42.6	61.2	45.6	22.3	46.7	56.3
SOLOv2	Res-DCN-101-FPN	44.9	63.8	48.2	23.1	48.9	61.2

TABLE 4.9. **Object detection** box AP (%) on the MS COCO `test-dev`. Although our bounding boxes are directly generated from the predicted masks, the accuracy outperforms most state-of-the-art methods. Speed-accuracy trade-off of typical methods is shown in Figure 4.4.

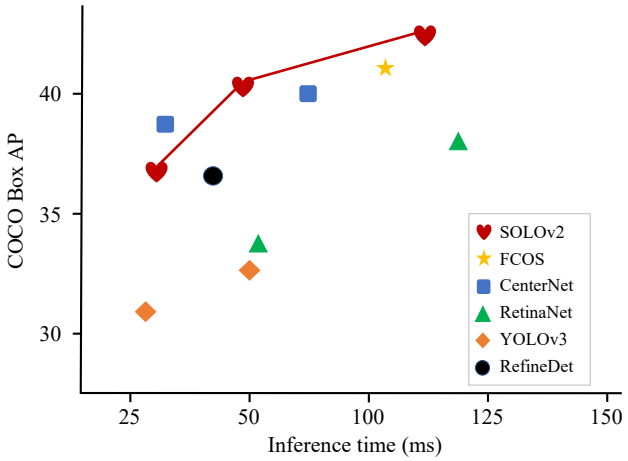


FIGURE 4.4. SOLOv2 for **object detection**. Speed-accuracy trade-off of bounding-box detection on the COCO `test-dev`.

the mask feature branch. We use annotations of COCO 2018 panoptic segmentation task. All models are trained on `train2017` subset and tested on `val2017`. We use the same strategy as in Panoptic-FPN (Kirillov et al., 2019a) to combine instance and semantic results. As shown in Table 4.10, our method achieves state-of-the-art results and outperforms other recent box-free methods by a large margin. All methods listed use the same backbone (ResNet50-FPN) except SSAP (ResNet101) and Pano-DeepLab (Xception-71). Note that UPSNet has used deformable convolutions (Dai et al., 2017b) for better performance.

4.3.4 Extension: Instance-level Image Matting

Image matting is a fundamental problem in computer vision and graphics and has attracted much research attention (Levin, Lischinski, and Weiss, 2007; Xu et al., 2017; Lu et al., 2019). Given an image, image matting demands for accurate foreground

	PQ	PQ^{Th}	PQ^{St}
box-based:			
AUNet (Li et al., 2019b)	39.6	49.1	25.2
UPSNet (Xiong et al., 2019)	**42.5**	48.5	**33.4**
Panoptic-FPN (Kirillov et al., 2019a)	39.0	45.9	28.7
Panoptic-FPN*-1×	38.7	45.9	27.8
Panoptic-FPN*-3×	40.8	48.3	29.4
box-free:			
AdaptIS (Sofiiuk, Barinova, and Konushin, 2019)	35.9	40.3	29.3
SSAP (Gao et al., 2019)	36.5	–	–
Pano-DeepLab (Cheng et al., 2020)	39.7	43.9	**33.2**
SOLOv2	**42.1**	**49.6**	30.7

TABLE 4.10. SOLOv2 for **panoptic segmentation** – results on MS COCO val2017. $*$ means re-implementation.

estimation, which is typically formulated as the alpha map prediction, *i.e.*, to output the soft transition between foreground and background. Most matting methods require an extra trimap input, which indicates the regions of absolute foreground, absolute background and unknown. Few methods (Chen et al., 2018b; Zhang et al., 2019; Liu et al., 2020a) explore the trimap-free solution. There is an obvious ambiguity: without trimap it is very challenging to tell which object is the target foreground object given multiple objects in an image. However, it can be solved from another perspective: *to perform image matting at the instance level.* Thus, we will be able to deal with arbitrary foreground objects and enable much more flexible, automated image editing.

To further demonstrate the flexibility of the proposed framework, we extend the proposed instance segmentation framework to perform image matting at instance-level. Thanks to the ability of generating high-quality object masks, our method is able to solve instance-level image matting problem with minimal modifications.

In this setting, instead of the binary pixel-wise mask, for each instance, we want to attain the soft transitions between objects and the background, *i.e.*, soft matte, which is critically important for photo-realistic image manipulation.

Method

The modifications lie in the mask feature branch and the loss function. In mask feature branch, we use the raw input to enhance the feature representation in image details. For the loss function, we add a mean average error term computed between the predicted soft matte and the ground-truth matte. The model is initialized by the weights pre-trained on COCO instance segmentation. During inference, we obtain the soft mattes after sigmoid operation and no thresholding is needed.

(a) Before the Details Refinement module (b) After the Details Refinement module

FIGURE 4.5. Behavior of the Details Refinement module. Each plotted sub-figure corresponds to one of the 32 channels of the input/output feature maps. The output feature map recovers the low-level details. Best viewed on screens.

In parallel with the prediction head, the mask feature branch takes the feature maps as input and fuses them into a $1/4$ scale feature map. Specifically, we modify the mask feature branch by introducing the Details Refinement module. It takes $1/4$ scale fused features and $1/1$ scale raw image as input and outputs the $1/1$ scale low-level features. Specifically, after 1×1 conv the $4\times$ bilinear upsampling, the input features are concatenated with the raw RGB input. The output features after three 3×3 convs are expanded to 256 channels through a 1×1 conv. The hidden feature maps all have a small number of channels, *e.g.*, 32. The output and the upsampled original features are fused together through an element-wise summation as the final mask features, which will be convolved by the predicted convolution kernels to generate individual soft mattes for all the objects.

Dataset

To our knowledge, no existing public datasets provide separate alpha matte annotations for each instance. Thus, we construct a training set consists of human matting data with 880 alpha mattes and 520 images with person category from the LVIS training set (Gupta, Dollar, and Girshick, 2019).

Results

We apply the well-trained model to the images collected from the Internet. As shown in Figure 4.6(a), our model is able to generate high-quality alpha matte. The predicted alpha matte enables us to perform some image editing applications for instance-level image editing and compositing. The accurate predictions at object boundaries make the composite image look very natural. In order to analyze the effect of details refinement, we visualize the feature maps at its input and output. As shown in Figure 4.5(a), the input feature maps show coarse activation at boundaries and details. The details are recovered after the detail refinement.

Note that the model trained with human matting data has reasonable matting results for objects of other categories, *e.g.*, the dog in Figure 4.6(a). This capability comes from two aspects. 1) As mentioned in Section 4.3.4, the model is initialized

(a) Image Matting Results

(b) Image Editing Applications

FIGURE 4.6. (a) Visualization of image matting results. We show the input image and the corresponding output alpha matte of our method. Best viewed on screens. (b) Demonstration of image editing applications (from left to right: original image; Bokeh background effect focusing on one instance; image composition).

by the weights pre-trained on COCO instance segmentation. During the training, the weights of the object category branch are frozen for maintaining the strong object recognition ability. 2) The matting capability is transferred from human to other categories. Because in our framework, the matte prediction is class-agnostic. Basically, the mask branch predicts the soft masks for all the potential objects. The object classes are determined by the category branch. Thus, when the model learns with human matting data, the enhanced matte features (Figure 4.5(b)) also benefit objects of other categories.

4.4 Conclusion

In this work, we proposed SOLOv2, a dynamic and fast instance segmentation solution with strong performance. The method includes three key techniques. a) We proposed to learn dynamic convolutional kernels for the mask prediction, conditioned on the location, which leads to a much more compact yet more powerful head design, and achieving better results; b) We re-designed the object mask generation in a simple and unified way, which yields more accurate boundaries; b) Moreover, unlike box NMS as in object detection, for direct instance segmentation a bottleneck in inference efficiency is the NMS of masks. We developed a simple and much faster NMS strategy, termed Matrix NMS, for NMS processing of masks, without sacrificing mask AP.

Our experiments on the MS COCO and LVIS datasets demonstrate the superior performance in terms of both accuracy and speed of the proposed SOLOv2. Being versatile for instance-level recognition tasks, we show that without any modification to the framework, SOLOv2 performs competitively for panoptic segmentation. Thanks to its simplicity (being proposal free, anchor free, FCN-like), strong performance in both accuracy and speed, and potentially being capable of solving many instance-level tasks, SOLOv2 can be a strong baseline approach to instance recognition and beyond.

Statement of Authorship

Title of Paper	Dense Contrastive Learning for Self-Supervised Visual Pre-Training
Publication Status	☑ Published ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and Unsubmitted work written in manuscript style
Publication Details	Published in CVPR 2021

Principal Author

Name of Principal Author (Candidate)	Xinlong Wang		
Contribution to the Paper	Proposed the ideas, conducted experiments, and wrote the manuscript.		
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	27/05/2022

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	Rufeng Zhang		
Contribution to the Paper	Discussion, writing revision and conducting some experiments.		
Signature		Date	May 28, 2022

Name of Co-Author	Chunhua Shen		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	28/05/2022

Name of Co-Author	Tao Kong		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	28/05/2022

Name of Co-Author	Lei Li		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	5/28/2022

Chapter 5

Dense Contrastive Learning for Self-Supervised Visual Pre-Training

5.1 Introduction

Pre-training has become a well-established paradigm in many computer vision tasks. In a typical pre-training paradigm, models are first pre-trained on large-scale datasets and then fine-tuned on target tasks with less training data, *e.g.*, instance segmentation and semantic segmentation. Specifically, the supervised ImageNet pre-training has been dominant for years, where the models are pre-trained to solve image classification and transferred to downstream tasks. However, there is a gap between image classification pre-training and target dense prediction tasks, such as object instance segmentation (Everingham et al., 2010; Lin et al., 2014) and semantic segmentation (Cordts et al., 2016). The former focuses on assigning a category to an input image, while the latter needs to perform dense classification or regression over the whole image. For example, semantic segmentation aims to assign a category for each pixel, and object detection aims to predict the categories and bounding boxes for all object instances of interest. A straightforward solution would be to pre-train on dense prediction tasks directly. However, these tasks' annotation is notoriously time-consuming compared to the image-level labeling, making it hard to collect data at a massive scale to pre-train a universal feature representation.

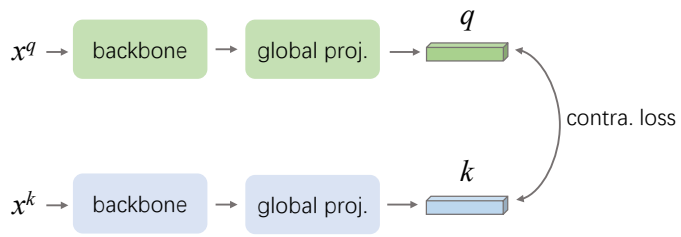
Recently, unsupervised visual pre-training has attracted much research attention, which aims to learn a proper visual representation from a large set of unlabeled images. A few methods (He et al., 2020; Chen et al., 2020b; Chen et al., 2020c; Grill et al., 2020) show the effectiveness in downstream tasks, which achieve comparable or better results compared to supervised ImageNet pre-training. However, the gap between image classification pre-training and target dense prediction tasks still exists. First, almost all recent self-supervised learning methods formulate the learning

as image-level prediction using global features. They all can be thought of as classifying each image into its own version, *i.e.*, instance discrimination (Wu et al., 2018). Moreover, existing approaches are usually evaluated and optimized on the image classification benchmark. Nevertheless, better image classification does not guarantee more accurate object detection, as shown in He, Girshick, and Dollár (2019). Thus, self-supervised learning that is customized for dense prediction tasks is on demand. As for unsupervised pre-training, dense annotation is no longer needed. A clear approach would be pre-training as a dense prediction task *directly*, thus removing the gap between pre-training and target dense prediction tasks.

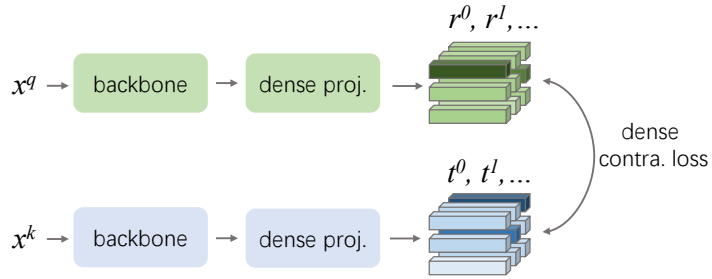
Inspired by the supervised dense prediction tasks, *e.g.*, SOLO for instance segmentation introduced in Chapter 3 and Chapter 4, which perform dense per-pixel classification, we propose dense contrastive learning (DenseCL) for self-supervised visual pre-training. DenseCL views the self-supervised learning task as a dense pairwise contrastive learning rather than the global image classification. First, we introduce a dense projection head that takes the features from backbone networks as input and generates dense feature vectors. Our method naturally preserves the spatial information and constructs a dense output format, compared to the existing global projection head that applies a global pooling to the backbone features and outputs a single, global feature vector for each image. Second, we define the positive sample of each local feature vector by extracting the correspondence across views. To construct an unsupervised objective function, we further design a dense contrastive loss, which extends the conventional InfoNCE loss (Oord, Li, and Vinyals, 2018) to a dense paradigm. With the above approaches, we perform contrastive learning densely using a fully convolutional network (FCN) (Long, Shelhamer, and Darrell, 2015b), similar to target dense prediction tasks.

Our main contributions are thus summarized as follows.

- We propose a new contrastive learning paradigm, *i.e.*, dense contrastive learning, which performs dense pairwise contrastive learning at the level of pixels (or local features).
- With the proposed dense contrastive learning, we design a simple and effective self-supervised learning method tailored for dense prediction tasks, termed DenseCL, which fills the gap between self-supervised pre-training and dense prediction tasks.
- DenseCL significantly outperforms the state-of-the-art MoCo-v2 (Chen et al., 2020c) when transferring the pre-trained model to downstream dense prediction tasks, including object detection (+2.0% AP), instance segmentation (+0.9% AP) and semantic segmentation (+3.0% mIoU), and far surpasses the supervised ImageNet pre-training.



(a) Global Contrastive Learning



(b) Dense Contrastive Learning

FIGURE 5.1. Conceptual illustration of two contrastive learning paradigms for representation learning. We use a pair of query and key for simpler illustration. The backbone can be any convolutional neural network. (a): The contrastive loss is computed between the single feature vectors outputted by the global projection head, at the level of global feature; (b): The dense contrastive loss is computed between the dense feature vectors outputted by the dense projection head, at the level of local feature. For both paradigms, the two branches can be the same encoder or different ones, *e.g.*, an encoder and its momentum-updated one.

5.2 Method

5.2.1 Background

For self-supervised representation learning, the breakthrough approaches are MoCo-v1/v2 (He et al., 2020; Chen et al., 2020c) and SimCLR (Chen et al., 2020b), which both employ contrastive unsupervised learning to learn good representations from unlabeled data. We briefly introduce the state-of-the-art self-supervised learning framework by abstracting a common paradigm.

Pipeline. Given an unlabeled dataset, an instance discrimination (Wu et al., 2018) pretext task is followed where the features of each image in the training set are pulled away from those of other images. For each image, random 'views' are generated by random data augmentation. Each view is fed into an encoder for extracting features that encode and represent the whole view. There are two core components in an encoder, *i.e.*, the backbone network and the projection head. The projection head attaches to the backbone network. The backbone is the model to be transferred after pre-training, while the projection head will be thrown away once the pre-training is

completed. For a pair of views, they can be encoded by the same encoder (Chen et al., 2020b), or separately by an encoder and its momentum-updated one (He et al., 2020). The encoder is trained by optimizing a pairwise contrastive (dis)similarity loss, as revisited below. The overall pipeline is illustrated in Figure 5.1(a).

Loss function. Following the principle of MoCo (He et al., 2020), the contrastive learning can be considered as a dictionary look-up task. For each encoded query q, there is a set of encoded keys $\{k_0, k_1, ...\}$, among which a single positive key k_+ matches query q. The encoded query and keys are generated from different views. For an encoded query q, its positive key k_+ encode different views of the same image, while the negative keys encode the views of different images. A contrastive loss function InfoNCE (Oord, Li, and Vinyals, 2018) is employed to pull q close to k_+ while pushing it away from other negative keys:

$$\mathcal{L}_q = -\log \frac{\exp(q \cdot k_+/\tau)}{\exp(q \cdot k_+/\tau) + \sum_{k_-} \exp(q \cdot k_-/\tau)}, \tag{5.1}$$

where τ denotes a temperature hyper-parameter as in Wu et al. (2018).

5.2.2 DenseCL Pipeline

We propose a new self-supervised learning framework tailored for dense prediction tasks, termed DenseCL. DenseCL extends and generalizes the existing framework to a dense paradigm. Compared to the existing paradigm revisited in 5.2.1, the core differences lie in the encoder and loss function. Given an input view, the dense feature maps are extracted by the backbone network, *e.g.*, ResNet (He et al., 2016a) or any other convolutional neural network, and forwarded to the following projection head. The projection head consists of two sub-heads in parallel, which are global projection head and dense projection head respectively. The global projection head can be instantiated as any of the existing projection heads such as the ones in He et al. (2020), Chen et al. (2020b), and Chen et al. (2020c), which takes the dense feature maps as input and generates a global feature vector for each view. For example, the projection head in Chen et al. (2020c) consists of a global pooling layer and an MLP which contains two fully connected layers with a ReLU layer between them. In contrast, the dense projection head takes the same input but outputs dense feature vectors.

Specifically, the global pooling layer is removed and the MLP is replaced by the identical 1×1 convolution layers (Long, Shelhamer, and Darrell, 2015b). In fact, the dense projection head has the same number of parameters as the global projection head. The backbone and two parallel projection heads are end-to-end trained by optimizing a joint pairwise contrastive (dis)similarity loss at the levels of both global features and local features.

5.2.3 Dense Contrastive Learning

We perform dense contrastive learning by extending the original contrastive loss function to a dense paradigm. We define a set of encoded keys $\{t_0, t_1, ...\}$ for each encoded query r. However, here each query no longer represents the whole view, but encodes a local part of a view. Specifically, it corresponds to one of the $S_h \times S_w$ feature vectors generated by the dense projection head, where S_h and S_w denote the spatial size of the generated dense feature maps. Note that S_h and S_w can be different, but we use $S_h = S_w = S$ for simpler illustration. Each negative key t_- is the pooled feature vector of a view from a different image. The positive key t_+ is assigned according to the extracted correspondence across views, which is one of the S^2 feature vectors from another view of the same image. For now, let us assume that we can easily find the positive key t_+. A discussion is deferred to the next section. The dense contrastive loss is defined as:

$$\mathcal{L}_r = \frac{1}{S^2} \sum_s -\log \frac{\exp(r^s \cdot t_+^s / \tau)}{\exp(r^s \cdot t_+^s / \tau) + \sum_{t_-^s} \exp(r^s \cdot t_-^s / \tau)}, \tag{5.2}$$

where r^s denotes the s^{th} out of S^2 encoded queries.

Overall, the total loss for our DenseCL can be formulated as:

$$\mathcal{L} = (1 - \lambda)\mathcal{L}_q + \lambda \mathcal{L}_r, \tag{5.3}$$

where λ acts as the weight to balance the two terms. λ is set to 0.5 which is validated by experiments in Section 5.3.3.

5.2.4 Dense Correspondence across Views

We extract the dense correspondence between the two views of the same input image. For each view, the backbone network extracts feature maps $\mathbf{F} \in \mathbb{R}^{H \times W \times K}$, from which the dense projection head generates dense feature vectors $\mathbf{\Theta} \in \mathbb{R}^{S_h \times S_w \times E}$. Note that S_h and S_w can be different, but we use $S_h = S_w = S$ for simpler illustration. The correspondence is built between the dense feature vectors from the two views, *i.e.*, $\mathbf{\Theta}_1$ and $\mathbf{\Theta}_2$. We match $\mathbf{\Theta}_1$ and $\mathbf{\Theta}_2$ using the backbone feature maps $\mathbf{F}_1$ and $\mathbf{F}_2$. The $\mathbf{F}_1$ and $\mathbf{F}_2$ are first downsampled to have the spatial shape of $S \times S$ by an adaptive average pooling, and then used to calculate the cosine similarity matrix $\mathbf{\Delta} \in \mathbb{R}^{S^2 \times S^2}$. The matching rule is that each feature vector in a view is matched to the most similar feature vector in another view. Specifically, for all the S^2 feature vectors of $\mathbf{\Theta}_1$, the correspondence with $\mathbf{\Theta}_2$ is obtained by applying an `argmax` operation to the similarity matrix $\mathbf{\Delta}$ along the last dimension. The matching process can be formulated as:

$$c_i = \arg\max_j sim(\boldsymbol{f}_i, \boldsymbol{f}'_j), \tag{5.4}$$

where $\boldsymbol{f}_i$ is the i^{th} feature vector of backbone feature maps $\mathbf{F}_1$, and $\boldsymbol{f}'_j$ is the j^{th} of $\mathbf{F}_2$. $sim(\boldsymbol{u}, \boldsymbol{v})$ denotes the cosine similarity, calculated by the dot product between ℓ_2 normalized $\boldsymbol{u}$ and $\boldsymbol{v}$, *i.e.*, $sim(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^\top \boldsymbol{v} / \|\boldsymbol{u}\| \|\boldsymbol{v}\|$. The obtained c_i denotes the i^{th} out of S^2 matching from $\boldsymbol{\Theta}_1$ to $\boldsymbol{\Theta}_2$, which means that i^{th} feature vector of $\boldsymbol{\Theta}_1$ matches ${c_i}^{\text{th}}$ of $\boldsymbol{\Theta}_2$. The whole matching process could be efficiently implemented by matrix operations, thus introducing negligible latency overhead.

For the simplest case where $S = 1$, the matching degenerates into the one in global contrastive learning as the single correspondence naturally exists between two global feature vectors, which is the case introduced in Section 5.2.1.

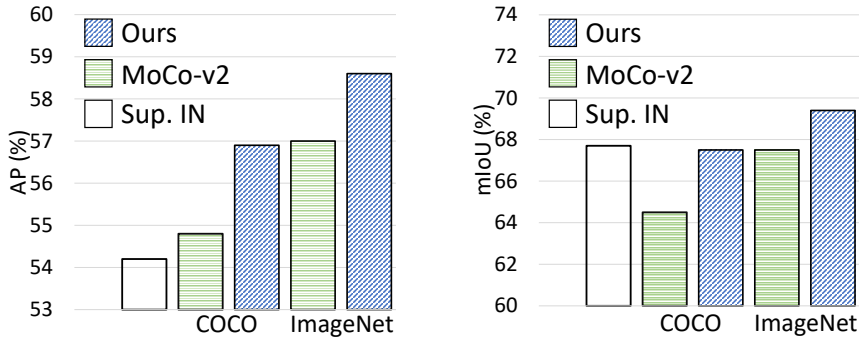
According to the extracted dense correspondence, one can easily find the positive key t_+ for each query r during the dense contrastive learning introduced in Section 5.2.3.

Note that without the global contrastive learning term (*i.e.*, $\lambda = 1$), there is a chicken-and-egg issue that good features will not be learned if incorrect correspondence is extracted, and the correct correspondence will not be available if the features are not sufficiently good. In our default setting where $\lambda = 0.5$, no unstable training is observed. Besides setting $\lambda \in (0, 1)$ during the whole training, we introduce two more solutions which can also tackle this problem, detailed in Section 5.3.4.

5.3 Experiments

We adopt MoCo-v2 (Chen et al., 2020c) as our baseline method, as which shows the state-of-the-art results and outperforms other methods by a large margin on downstream object detection task, as shown in Table 5.1. It indicates that it should serve as a very strong baseline on which we can demonstrate the effectiveness of our approach.

Technical details. We adapt most of the settings from Chen et al. (2020c). A ResNet (He et al., 2016a) is adopted as the backbone. The following global projection head and dense projection head both have a fixed-dimensional output. The former outputs a single 128-D feature vector for each input and the latter outputs dense 128-D feature vectors. Specifically, the dense projection head consists of adaptive average pooling (optional), 1×1 convolution, ReLU, and 1×1 convolution. Following Chen et al. (2020b) and Chen et al. (2020c), the hidden layer's dimension is 2048, and the final output dimension is 128. Each ℓ_2 normalized feature vector represents a query or key. For both the global and dense contrastive learning, the dictionary size is set to 65536. The momentum is set to 0.999. Shuffling BN (He et al., 2020) is used during the training. The temperature τ in Equation (5.1) and Equation (5.2) is set to 0.2. The data augmentation pipeline consists of 224×224-pixel ramdom resized cropping, random color jittering, random gray-scale conversion, gaussian blurring and random horizontal flip.



(a) Object Detection (b) Semantic Segmentation

FIGURE 5.2. Comparisons of pre-trained models by fine-tuning on object detection and semantic segmentation datasets. 'Sup. IN' denotes the supervised pre-training on ImageNet. 'COCO' and 'ImageNet' indicate the pre-training models trained on COCO and ImageNet respectively. (a): The object detection results of a Faster R-CNN detector fine-tuned on VOC `trainval07+12` for 24k iterations and evaluated on VOC `test2007`; (b): The semantic segmentation results of an FCN model fine-tuned on VOC `train_aug2012` for 20k iterations and evaluated on `val2012`. The results are averaged over 5 independent trials.

5.3.1 Experimental Settings

Datasets. The pre-training experiments are conducted on two large-scale datasets: MS COCO (Lin et al., 2014) and ImageNet (Deng et al., 2009). Only the training sets are used during the pre-training, which are ~118k and ~1.28 million images respectively. COCO and ImageNet represent two kinds of image data. The former is more natural and real-world, containing diverse scenes in the wild. It is a widely used and challenging dataset for object-level and pixel-level recognition tasks, such as object detection and instance segmentation. While the latter is heavily curated, carefully constructed for image-level recognition. A clear and quantitative comparison is the number of objects of interest. For example, COCO has a total of 123k images and 896k labeled objects, an average of 7.3 objects per image, which is far more than the ImageNet DET dataset's 1.1 objects per image.

Pre-training setup. For ImageNet pre-training, we closely follow MoCo-v2 (Chen et al., 2020c) and use the same training hyper-parameters. For COCO pre-training including both baseline and ours, we use an initial learning rate of 0.3 instead of the original 0.03, as the former shows better performance in MoCo-v2 baseline when pre-training on COCO. We adopt SGD as the optimizer and we set its weight decay and momentum to 0.0001 and 0.9. Each pre-training model is optimized on 8 GPUs with a cosine learning rate decay schedule and a mini-batch size of 256. We train for 800 epochs for COCO, which is a total ~368k iterations. For ImageNet, we train for 200 epochs, a total of 1 million iterations.

Evaluation protocol. We evaluate the pre-trained models by fine-tuning on the target dense prediction tasks end-to-end. Challenging and popular datasets are adopted

to fine-tune mainstream algorithms for different target tasks, *i.e.*VOC object detection, COCO object detection, COCO instance segmentation, VOC semantic segmentation, and Cityscapes semantic segmentation. When evaluating on object detection, we follow the common protocol that fine-tuning a Faster R-CNN detector (C4-backbone) on the VOC `trainval07+12` set with standard 2x schedule in Wu et al. (2019) and testing on the VOC `test2007` set.

In addition, we evaluate object detection and instance segmentation by fine-tuning a Mask R-CNN detector (FPN-backbone) with on COCO `train2017` split (~118k images) with the standard 1× schedule and evaluating on COCO 5k `val2017` split. we follow the settings in Tian et al. (2020). Synchronized batch normalization is used in backbone, FPN (Lin et al., 2017a) and prediction heads during the training.

For semantic segmentation, an FCN model (Long, Shelhamer, and Darrell, 2015b) is fine-tuned on VOC `train_aug2012` set (10582 images) for 20k iterations and evaluated on `val2012` set. We also evaluate semantic segmentation on Cityscapes dataset by training an FCN model on `train_fine` set (2975 images) for 40k iterations and test on `val` set. We follow the settings in `mmsegmentation` (OpenMMLab, 2020), except that the first 7×7 convolution is kept to be consistent with the pre-trained models. Batch size is set to 16. Synchronized batch normalization is used. Crop size is 512 for VOC (Everingham et al., 2010) and 769 for Cityscapes (Cordts et al., 2016).

5.3.2 Main Results

PASCAL VOC object detection. In Table 5.1, we report the object detection result on PASCAL VOC and compare it with other state-of-the-art methods. When pre-trained on COCO, our DenseCL outperforms the MoCo-v2 baseline by 2% AP. When pre-trained on ImageNet, the MoCo-v2 baseline has already surpassed other state-of-the-art self-supervised learning methods. And DenseCL still yields 1.7% AP improvements, strongly demonstrating the effectiveness of our method. The gains are consistent over all three metrics. It should be noted that we achieve much larger improvements on more stringent AP_{75} compared to those on AP_{50}, which indicates DenseCL largely helps improve the localization accuracy. Compared to the supervised ImageNet pre-training, we achieve the significant 4.5% AP gains.

COCO object detection and segmentation. The object detection and instance segmentation results on COCO are reported in Table 5.2 and Table 5.3. As shown in Table 5.2, DenseCL pre-training improves SOLOv2 instance segmentation by 0.5% AP and FCOS object detection by 1.0% AP, compared to supervised pre-training. As reported in Table 5.3, for object detection, DenseCL outperforms MoCo-v2 by 1.1% AP and 0.5% AP when pre-trained on COCO and ImageNet respectively. Note that fine-tuning on COCO with a COCO pre-trained model is not a typical scenario. But the clear improvements still show the effectiveness.

pre-train	AP	AP_{50}	AP_{75}
random init.	32.8	59.0	31.6
super. IN	54.2	81.6	59.8
MoCo-v2 CC	54.7	81.0	60.6
DenseCL CC	56.7	81.7	63.0
SimCLR IN (Chen et al., 2020b)	51.5	79.4	55.6
BYOL IN (Grill et al., 2020)	51.9	81.0	56.5
MoCo IN (He et al., 2020)	55.9	81.5	62.6
MoCo-v2 IN (Chen et al., 2020c)	57.0	82.4	63.6
MoCo-v2 IN*	57.0	82.2	63.4
DenseCL IN	58.7	82.8	65.2

TABLE 5.1. **Object detection fine-tuned on PASCAL VOC.** 'CC' and 'IN' indicate the pre-training models trained on COCO and ImageNet respectively. The models pre-trained on the same dataset are with the same training epochs, *i.e.*, 800 epochs for COCO and 200 epochs for ImageNet. '*' means re-implementation. The results of other methods are either from their papers or third-party implementation. All the detectors are trained on `trainval07+12` for 24k iterations and evaluated on `test2007`. The metrics include the VOC metric AP_{50} (*i.e.*, IoU threshold is 50%) and COCO-style AP and AP_{75}. The results are averaged over 5 independent trials.

pre-train	AP^m	AP^m_{50}	AP^m_{75}	AP^b	AP^b_{50}	AP^b_{75}
super. IN	35.2	55.3	37.3	39.9	58.9	43.7
MoCo-v2 IN	35.2	54.9	37.7	40.3	58.8	43.5
DenseCL IN	35.7	55.1	38.6	40.9	59.2	44.2

TABLE 5.2. **SOLOv2 instance segmentation and FCOS object detection fine-tuned on COCO.** 'CC' and 'IN' indicate the pre-training models trained on COCO and ImageNet respectively. All the models are trained on `train2017` with default 1× schedule and evaluated on `val2017`. The metrics include mask AP (AP^m) and bounding box AP (AP^b).

In Table 5.4, we further evaluate the pre-trained models on semi-supervised object detection. In this semi-supervised setting, only 10% training data is used during the fine-tuning. DenseCL outperforms MoCo-v2 by 1.3% AP^b and 1.0% AP^b when pre-training on COCO and ImageNet respectively. It should be noted that the gains are more significant than that of the fully-supervised setting which uses all of ~118k images during the fine-tuning. For example, when pre-training on ImageNet, DenseCL surpasses MoCo-v2 by 1.0% AP^b and 0.5% AP^b for semi-supervised setting and fully-supervised setting respectively.

PASCAL VOC semantic segmentation. We show the largest improvements on semantic segmentation. As shown in Table 5.5, DenseCL yields 3% mIoU gains when pre-training on COCO and fine-tuning an FCN on VOC semantic segmentation. The COCO pre-trained DenseCL achieves the same 67.5% mIoU as ImageNet pre-trained MoCo-v2. Note that compared to 200-epoch ImageNet pre-training, 800-epoch COCO

pre-train	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
random init.	32.8	50.9	35.3	29.9	47.9	32.0
super. IN	39.7	59.5	43.3	35.9	56.6	38.6
MoCo-v2 CC	38.5	58.1	42.1	34.8	55.3	37.3
DenseCL CC	39.6	59.3	43.3	35.7	56.5	38.4
SimCLR IN	38.5	58.0	42.0	34.8	55.2	37.2
BYOL IN	38.4	57.9	41.9	34.9	55.3	37.5
MoCo-v2 IN	39.8	59.8	43.6	36.1	56.9	38.7
DenseCL IN	40.3	59.9	44.3	36.4	57.0	39.2

TABLE 5.3. **Object detection and instance segmentation fine-tuned on COCO.** 'CC' and 'IN' indicate the pre-training models trained on COCO and ImageNet respectively. All the detectors are trained on `train2017` with default 1× schedule and evaluated on `val2017`. The metrics include bounding box AP (AP^b) and mask AP (AP^m).

pre-train	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
random init.	20.6	34.0	21.5	18.9	31.7	19.8
super. IN	23.6	37.7	25.4	21.8	35.4	23.2
MoCo-v2 CC	22.8	36.4	24.2	20.9	34.6	21.9
DenseCL CC	24.1	38.1	25.6	21.9	36.0	23.0
MoCo-v2 IN	23.8	37.5	25.6	21.8	35.4	23.2
DenseCL IN	24.8	38.8	26.8	22.6	36.8	23.9

TABLE 5.4. **Semi-supervised object detection and instance segmentation fine-tuned on COCO.** During the fine-tuning, only 10% training data is used. 'CC' and 'IN' indicate the pre-training models trained on COCO and ImageNet respectively. All the detectors are trained on `train2017` for 90k iterations and evaluated on `val2017`. The metrics include bounding box AP (AP^b) and mask AP (AP^m).

pre-training only uses ~1/10 images and ~1/3 iterations. When pre-trained on ImageNet, DenseCL consistently brings 1.9% mIoU gains. It should be noted that the ImageNet pre-trained MoCo-v2 shows no transfer superiority compared with the supervised counterpart (67.5% vs. 67.7% mIoU). But DenseCL outperforms the supervised pre-training by a large margin, *i.e.*, 1.7% mIoU.

Cityscapes semantic segmentation. Cityscapes is a benchmark largely different from the above VOC and COCO. It focuses on urban street scenes. Nevertheless, in Table 5.5, we observe the same performance boost with DenseCL. Even the COCO pre-trained DenseCL can surpass the supervised ImageNet pre-trained model by 1.9% mIoU.

5.3.3 Ablation Study

We conduct extensive ablation experiments to show how each component contributes to DenseCL. We report ablation studies by pre-training on COCO and fine-tuning on VOC0712 object detection, as introduced in Section 5.3.1. All the detection results

(a) PASCAL VOC

pre-train	mIoU
random init.	40.7
super. IN	67.7
MoCo-v2 CC	64.5
DenseCL CC	67.5
SimCLR IN	64.3
BYOL IN	63.3
MoCo-v2 IN	67.5
DenseCL IN	69.4

(b) Cityscapes

pre-train	mIoU
random init.	63.5
super. IN	73.7
MoCo-v2 CC	73.8
DenseCL CC	75.6
SimCLR IN	73.1
BYOL IN	71.6
MoCo-v2 IN	74.5
DenseCL IN	75.7

TABLE 5.5. **Semantic segmentation on PASCAL VOC and Cityscapes.** 'CC' and 'IN' indicate the pre-training models trained on COCO and ImageNet respectively. The metric is the commonly used mean IoU (mIoU). Results are averaged over 5 independent trials.

are averaged over 5 independent trials. We also provide results of VOC2007 SVM Classification, following Goyal et al. (2019) and Zhan et al. (2020) which train linear SVMs on the VOC `train2007` split using the features extracted from the frozen backbone and evaluate on the `test2007` split.

Loss weight λ. The hyper-parameter λ in Equation (5.3) serves as the weight to balance the two contrastive loss terms, *i.e.*, the global term and the dense term. We report the results of different λ in Table 5.6. It shows a trend that the detection performance improves when we increase the λ. For the baseline method, *i.e.*, $\lambda = 0$, the result is 54.7% AP. The AP is 56.2% when $\lambda = 0.3$, which improves the baseline by 1.5% AP. Increasing λ from 0.3 to 0.5 brings another 0.5% AP gains. Although further increasing it to 0.7 still gives minor improvements (0.1% AP) on detection performance, the classification result drops from 82.9% to 81.0%. Considering the trade-off, we use $\lambda = 0.5$ as our default setting in other experiments. It should be noted that when $\lambda = 0.9$, compared to the MoCo-v2 baseline, the classification performance rapidly drops (-4.8% mAP) while the detection performance improves for 0.8% AP. It is in accordance with our intention that DenseCL is specifically designed for dense prediction tasks.

Matching strategy. In Table 5.7, we compare three different matching strategies used to extract correspondence across views. 1) 'random': the dense feature vectors from two views are randomly matched; 2) 'max-sim $\mathbf{\Theta}$': the dense correspondence is extracted using the dense feature vectors $\mathbf{\Theta}_1$ and $\mathbf{\Theta}_2$ generated by the dense projection head; (3) 'max-sim $\mathbf{F}$': the dense correspondence is extracted according to the backbone features $\mathbf{F}_1$ and $\mathbf{F}_2$, as in Equation 5.4. The random matching strategy can also achieve 1.3% AP improvements compared to MoCo-v2, meanwhile the classification performance drops by 0.9% mAP. It may be because 1) the dense output format itself helps, and 2) part of the random matches are somewhat correct. Matching by the outputs of dense projection head, *i.e.*, $\mathbf{\Theta}_1$ and $\mathbf{\Theta}_2$, brings no clear improvement.

λ	Detection AP	Detection AP_{50}	Detection AP_{75}	Classification mAP
0.0	54.7	81.0	60.6	82.6
0.1	55.2	81.4	61.4	82.9
0.3	56.2	81.5	62.6	83.3
0.5	56.7	81.7	63.0	82.9
0.7	56.8	81.9	63.1	81.0
0.9	55.5	80.9	61.3	77.8
1.0*	53.5	79.5	58.8	68.9

TABLE 5.6. **Ablation study of weight λ.** $\lambda = 0$ is the MoCo-v2 baseline. $\lambda = 0.5$ shows the best trade-off between detection and classification. '*' indicates training with warm-up, as discussed in Section 5.3.4.

strategy	Detection AP	Detection AP_{50}	Detection AP_{75}	Classification mAP
random	56.0	81.3	62.0	81.7
max-sim $\mathbf{\Theta}$	56.0	81.5	62.1	81.8
max-sim $\mathbf{F}$	56.7	81.7	63.0	82.9

TABLE 5.7. **Ablation study of matching strategy.** To extract the dense correspondence according to the backbone features $\mathbf{F}_1$ and $\mathbf{F}_2$ shows the best results.

The best results are obtained by extracting the dense correspondence according to the backbone features $\mathbf{F}_1$ and $\mathbf{F}_2$.

Grid size. In the default setting, the adopted ResNet backbone outputs features with stride 32. For a 224×224-pixel crop, the backbone features $\mathbf{F}$ has the spatial size of 7×7. We set the spatial size of the dense feature vectors $\mathbf{\Theta}$ to 7×7 by default, *i.e.*, $S = 7$. However, S can be flexibly adjusted and $\mathbf{F}$ will be pooled to the designated spatial size by an adaptive average pooling, as introduced in Section 5.2.4. We report the results of using different numbers of grid in Table 5.8. For $S = 1$, it is the same as the MoCo-v2 baseline except for two differences. 1) The parameters of dense projection head are independent with those of global projection head. 2) The dense contrastive learning maintains an independent dictionary. The results are similar to those of MoCo-v2 baseline. It indicates that the extra parameters and dictionary do not bring improvements. The performance improves as the grid size increases. We use grid size being 7 as the default setting, as the performance becomes stable when the S grows beyond 7.

Negative samples. We use the global average pooled features as negatives because it's conceptually simpler. Besides pooling, sampling is an alternative strategy. For keeping the same number of negatives, one can randomly sample a local feature from a different image. The COCO pre-trained model with sampling strategy achieves 56.7% AP on VOC detection, which is the same as the adopted pooling strategy.

grid size	Detection AP	Detection AP_{50}	Detection AP_{75}	Classification mAP
1	54.6	80.8	60.5	82.2
3	55.6	81.3	61.5	81.6
5	56.1	81.4	62.2	82.6
7	56.7	81.7	63.0	82.9
9	56.7	82.1	63.2	82.9

TABLE 5.8. **Ablation study of grid size S.** The results increase as the S gets larger. We use grid size being 7 in other experiments, as the performance becomes stable when the S grows beyond 7.

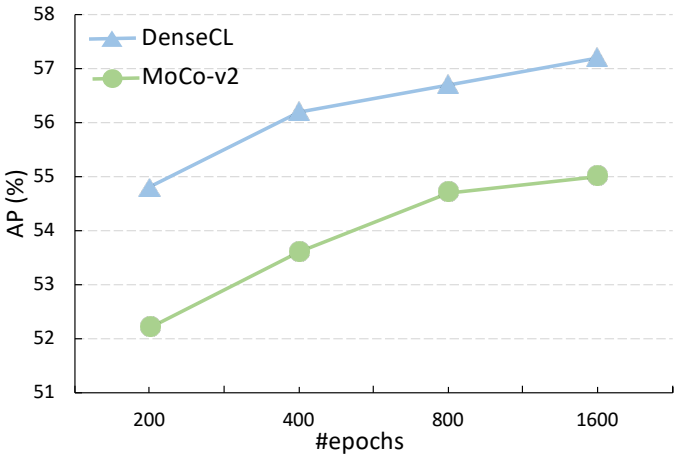


FIGURE 5.3. Different pre-training schedules on COCO. For each pre-trained model, a Faster R-CNN detector is fine-tuned on VOC `trainval07+12` for 24k iterations and evaluated on `test2007`. The metric is the COCO-style AP. Results are averaged over 5 independent trials.

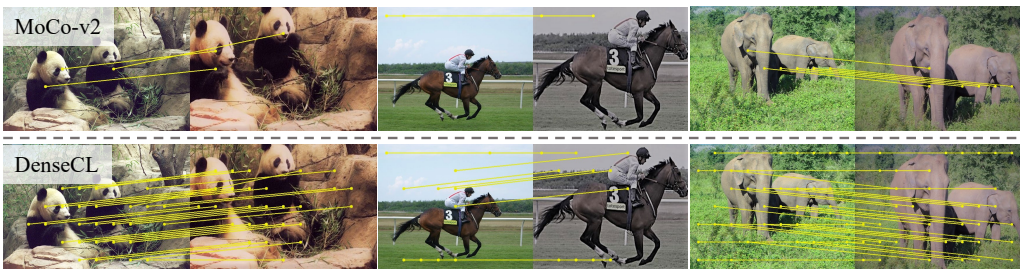


FIGURE 5.4. Visualization of dense correspondence. The correspondence is extracted between two views of the same image, using the 200-epoch ImageNet pre-trained model. DenseCL extracts more high-similarity matches compared with MoCo-v2. Best viewed on screen.

Training schedule. We show the results of using different training schedules in Table 5.9. The performance consistently improves as the training schedule gets longer, from 200 epochs to 1600 epochs. Note that the 1600-epoch COCO pre-trained DenseCL even surpasses the 200-epoch ImageNet pre-trained MoCO-v2 (57.2% AP vs. 57.0% AP). Compared to 200-epoch ImageNet pre-training, 1600-epoch COCO pre-training only uses ~1/10 images and ~7/10 iterations. In Figure 5.3, we further

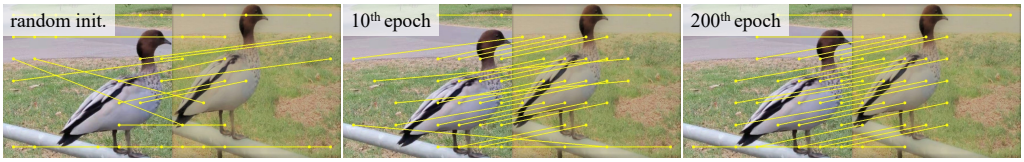


FIGURE 5.5. Comparison of dense correspondence extracted from random initialization to well trained DenseCL. The correspondence is extracted between two views of the same image, using the ImageNet pre-trained model. All the matches are visualized without thresholding.

provide an intuitive comparison with the baseline as the training schedule gets longer. It shows that DenseCL consistently outperforms the MoCo-v2 by at least 2% AP.

	Detection			Classification
#epochs	AP	AP_{50}	AP_{75}	mAP
200	54.8	80.5	60.7	77.6
400	56.2	81.5	62.3	81.3
800	56.7	81.7	63.0	82.9
1600	57.2	82.2	63.6	83.0

TABLE 5.9. **Ablation study of training schedule.** The results consistently improve as the training schedule gets longer. Although 1600-epoch training schedule is 0.5% AP better, we use 800-epoch schedule in other experiments for faster training.

Pre-training time. In Table 5.10, we compare DenseCL with MoCo-v2 in terms of training time. DenseCL is only 1s and 6s slower per epoch when pre-trained on COCO and ImageNet respectively. The overhead is less than 1%. It strongly demonstrates the efficiency of our method.

5.3.4 Discussions on DenseCL

To further study how DenseCL works, in this section, we visualize the learned dense correspondence in DenseCL. The issue of chicken-and-egg during the training is also discussed.

Dense correspondence visualization. We visualize the dense correspondence from two aspects: comparison of the final correspondence extracted from different pre-training methods, *i.e.*, MoCo-v2 vs. DenseCL, and the comparison of different training status, *i.e.*, from the random initialization to well trained DenseCL. Given two views of the same image, we use the pre-trained backbone to extract the features $\mathbf{F}_1$ and $\mathbf{F}_2$. For each feature vector in $\mathbf{F}_1$, we find the corresponding feature vector in $\mathbf{F}_2$ which has the highest cosine similarity. The match is kept if the same match holds from $\mathbf{F}_2$ to $\mathbf{F}_1$. Each match is assigned an averaged similarity. In Figure 5.4, we visualize the high-similarity matches (*i.e.*, similarity ≥ 0.9). DenseCL extracts many more high-similarity matches than its baseline. It is in accordance with our intention that the local features extracted from the two views of the same image should be similar.

time/epoch	COCO	ImageNet
MoCo-v2	1′45″	16′48″
DenseCL	1′46″	16′54″

TABLE 5.10. **Pre-training time comparison.** The training time per epoch is reported. We measure the results on the same 8-GPU machine. The training time overhead introduced by DenseCL is less than 1%.

Figure 5.5 shows how the correspondence changes over training time. The randomly initialized model extracts some random noisy matches. The matches get more accurate as the training time increases.

Chicken-and-egg issue. In our pilot experiments, we observe that the training loss does not converge if we set λ to 1.0, *i.e.*, removing the global contrastive learning, and only applying the dense contrastive learning. It may be because at the beginning of the training, the randomly initialized model is not able to generate correct correspondence across views. It is thus a chicken-and-egg issue that good features will not be learned if incorrect correspondence is extracted, and the correct correspondence will not be available if the features are not sufficiently good. As shown in Figure 5.5, most of the matches are incorrect with the random initialization. The core solution is to provide a guide when training starts, to break the deadlock. We introduce three different solutions to tackle this problem. 1) To initialize the model with the weights of a pre-trained model; 2) To set a warm-up period at the beginning during which the λ is set to 0; 3) To set $\lambda \in (0, 1)$ during the whole training. They all solve this issue well. The second one is reported in Table 5.6, with λ being changed from 0 to 1.0 after the first 10k iterations. We adopt the last one as the default setting for its simplicity.

5.4 Conclusion

In this work we have developed a simple and effective self-supervised learning framework DenseCL, which is designed and optimized for dense prediction tasks. A new contrastive learning paradigm is proposed to perform dense pairwise contrastive learning at the level of pixels (or local features). Our method largely closes the gap between self-supervised pre-training and dense prediction tasks, and shows significant improvements in a variety of tasks and datasets, including PASCAL VOC object detection, COCO object detection, COCO instance segmentation, PASCAL VOC semantic segmentation and Cityscapes semantic segmentation. We expect the proposed effective and efficient self-supervised pre-training techniques could be applied to larger-scale data to fully realize its potential, as well as hoping that DenseCL pre-trained models would completely replace the supervised pre-trained models in many of those dense prediction tasks in computer vision.

Statement of Authorship

Title of Paper	FreeSOLO: Learning to Segment Objects without Annotations
Publication Status	☐ Published ☑ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and Unsubmitted work written in manuscript style
Publication Details	Accepted by CVPR 2022

Principal Author

Name of Principal Author (Candidate)	Xinlong Wang		
Contribution to the Paper	Proposed the ideas, conducted experiments, and wrote the manuscript.		
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	17/05/2022

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	Zhiding Yu		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	

Name of Co-Author	Shalini De Mello		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	May 23, 2022

Name of Co-Author	Jan Kautz		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	

Name of Co-Author	Anima Anandkumar		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	May 31, 2022

Name of Co-Author	Chunhua Shen		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	27/05/2022

Name of Co-Author	Jose M. Alvarez		
Contribution to the Paper	Discussion and writing revision.		
Signature		Date	May 18th 2022

Chapter 6

FreeSOLO: Learning to Segment Objects without Annotations

6.1 Introduction

Recently, significant progress (Li et al., 2017; He et al., 2017a; De Brabandere, Neven, and Van Gool, 2017; Chen et al., 2019a; Bolya et al., 2019; Tian, Shen, and Chen, 2020; Wang et al., 2021c) has been made to address the instance segmentation task. As in Chapter 3 and Chapter 3, SOLO and SOLOv2 demonstrate a very simple framework with superior speed and accuracy. However, the dense prediction nature of the task requires rich and expensive annotations during training. Weakly-supervised instance segmentation methods are thus proposed to relax the annotation requirements (Khoreva et al., 2017; Hsu et al., 2019; Liu et al., 2020b; Tian et al., 2021; Cheng, Parkhi, and Kirillov, 2021; Lan et al., 2021). Latest methods such as BoxInst (Tian et al., 2021) and DiscoBox (Lan et al., 2021) have significantly closed the gap to fully supervised methods. However, their competitive result still relies on box or point annotations that contain strong localization information.

In this work, we explore *learning class-agnostic instance segmentation without any annotations*. The work here is built upon SOLOv2, a simple yet strong instance segmentation framework introduced in Chapter 4, and the self-supervised dense feature learning method of DenseCL proposed in Chapter 5. SOLO adopts a one-stage design, which contains a category branch and a mask branch to encode the object category information and segmentation proposals, respectively. Our main intuition is that this "top-down meets bottom-up" design allows us to unify pixel grouping, object localization and feature pre-training in a fully self-supervised manner.

Our proposed framework, **FreeSOLO**, contains two major pillars: Free Mask and Self-supervised SOLO, as shown in Figure 6.1. Specifically, Free Mask contains self-supervised design elements that promote objectness in network attention. It contains a "query-key" attention design, where the queries and keys are constructed from self-supervised features. The method takes the cosine similarity between each query with all the keys, thus obtaining a set of query-conditioned (seeded) attention maps as

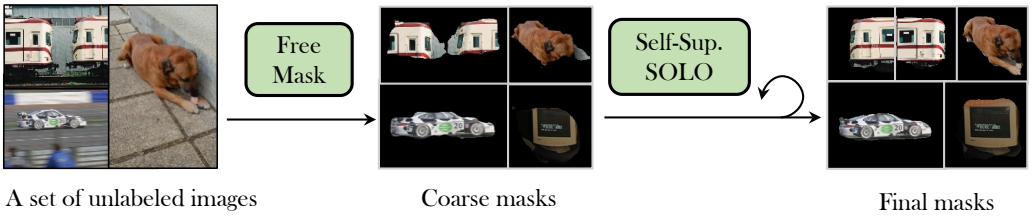


FIGURE 6.1. **Overview of FreeSOLO**. Unlabeled images are first input to Free Mask to generate coarse object masks. The segmentation masks as well as their associated semantic embeddings are used to train a SOLO-based instance segmentation model via weak supervision. We use self-training to improve object mask segmentation.

coarse masks. The coarse masks are ranked and filtered by their maskness scores, followed by non-maximum suppression (NMS) to further remove the redundant masks. Self-Supervised SOLO then takes the coarse masks as pseudo-labels to train a SOLO model. Since the coarse masks can be inaccurate, Self-Supervised SOLO contains a weakly-supervised design to better accommodate the label noise. This is followed by a self-training strategy to further refine mask quality and to improve accuracy. Our network design is almost the same as SOLO with minimal modifications, thus leading to simple and fast inference process.

FreeSOLO provides an effective solution to the challenging problem of self-supervised instance segmentation. With the bounding boxes obtained from the predicted masks, FreeSOLO also shows a significant advantage as an unsupervised object discovery method. In addition to the above roles, we further consider FreeSOLO as a strong self-supervised pretext task for instance segmentation by jointly learning object-level and pixel-level representations. Compared to pre-training for image classification (He et al., 2020; Chen et al., 2020b; Grill et al., 2020), object detection (Dai et al., 2021; Hénaff et al., 2021) and semantic segmentation (Pinheiro et al., 2020b; Chaitanya et al., 2020b), pre-training for instance segmentation is still under-studied. General instance segmentation requires not only localizing objects at the pixel level, but also recognizing their semantic categories. Interestingly, the design of FreeSOLO allows us to directly learn object-level semantic representations in an unsupervised manner. Upon completing the pre-training, all the learned parameters except for the last classification layer can be used to initialize the supervised instance segmentation models to improve accuracy.

Our contributions can be summarized as follows.

- We propose the Free Mask approach, which leverages the specific design of SOLO to effectively extract coarse object masks and semantic embeddings in an unsupervised manner.
- We further propose Self-Supervised SOLO, which takes the coarse masks and semantic embeddings from Free Mask and trains the SOLO instance segmentation model, with several novel design elements to overcome label noise in the coarse

masks.

- With the above methods, FreeSOLO presents a simple and effective framework that demonstrates unsupervised instance segmentation successfully for the first time. Notably, it outperforms some proposal generation methods that use manual annotations. FreeSOLO also outperforms state-of-the-art methods for unsupervised object detection/discovery by a significant margin (relative +100% in COCO AP).
- In addition, FreeSOLO serves as a strong self-supervised pretext task for representation learning for instance segmentation. For example, when fine-tuning on COCO dataset with 5% labeled masks, FreeSOLO outperforms DenseCL (Wang et al., 2021b) by +9.8% AP.

6.2 Method

6.2.1 Overview of FreeSOLO

We propose a novel framework for self-supervised instance segmentation, termed FreeSOLO. FreeSOLO does not require any type of annotations, neither pixel-level nor image-level labels, and simply uses a collection of unlabeled images for training. Its overall pipeline is illustrated in Figure 6.1. We first propose the Free Mask approach to generate segmentation masks from a self-supervised pre-trained model. For each unlabeled image, the coarse object masks can be generated fast with simple operations, *e.g.*, at 21 FPS on a V100 GPU with a ResNet-50-based backbone. We further propose Self-Supervised SOLO, which trains the SOLO-based instance segmenter using the coarse masks and semantic embeddings from Free Mask, with several novel design elements including weaky-supervised design, self-training, and semantic embedding learning.

With FreeSOLO, we obtain an instance segmentation model given only unlabeled images. In addition to unsupervised instance segmentation itself, the well-trained model serves as a strong pre-trained model for downstream fine-tuning. All its parameters except the last classification layer can be transferred to supervised instance segmentation as a strong initialization.

6.2.2 Free Mask

Free Mask generates object masks from unlabeled images. As shown in Figure 6.2, given an input image, dense feature maps $\mathbf{I} \in \mathbb{R}^{H \times W \times E}$ are extracted by a backbone model trained via self-supervision, *e.g.*, ResNet (He et al., 2016b) or any other convolutional neural network. This pre-trained model can be from supervised or unsupervised pre-training, as discussed below. We first construct queries $\mathbf{Q}$ and keys $\mathbf{K}$ from the features $\mathbf{I}$, which work together to generate the coarse segmentation masks. We downsample $\mathbf{I}$ by bilinear interpolation to form the queries $\mathbf{Q} \in \mathbb{R}^{H' \times W' \times E}$, where H' and W' denote the downsampled spatial size. $\mathbf{I}$ itself is used as the set of keys $\mathbf{K}$. For each query in $\mathbf{Q}$, we compute its cosine similarity with every key in $\mathbf{K}$, thus

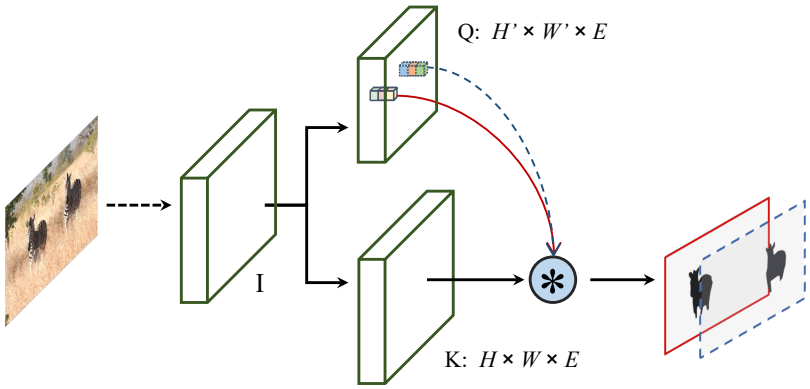


FIGURE 6.2. **The Free Mask approach.** Given queries and keys from the backbone feature **I**, the keys are convolved by the queries to generate segmentation masks. The masks go through NMS to form the object mask outputs.

obtaining the score maps $\mathbf{S} \in \mathbb{R}^{H \times W \times N}$, where $N = H' \times W'$ is the total number of queries. This operation can be written as:

$$\mathbf{S}_{i,j,q} = \texttt{sim}(\mathbf{Q}_q, \mathbf{K}_{i,j}), \tag{6.1}$$

where $\mathbf{Q}_q \in \mathbb{R}^E$ is the q^{th} query, and $\mathbf{K}_{i,j} \in \mathbb{R}^E$ is the key at spatial location (i, j). $\texttt{sim}(\boldsymbol{u}, \boldsymbol{v})$ denotes the cosine similarity, calculated by the dot product between ℓ_2-normalized $\boldsymbol{u}$ and $\boldsymbol{v}$, *i.e.*, $\texttt{sim}(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^\top \boldsymbol{v} / \|\boldsymbol{u}\| \|\boldsymbol{v}\|$. The process can also be viewed as a convolution where the ℓ_2 normalized queries $\mathbf{Q}'$ and keys $\mathbf{K}'$ are respectively the convolutional kernels and the features to be convolved together. Each of the normalized queries is treated as a 1×1 convolutional kernel. Thus the operation can also be written as:

$$\mathbf{S} = \mathbf{Q}' \circledast \mathbf{K}'. \tag{6.2}$$

The score maps are then normalized as soft masks by shifting the scores to the range $[0, 1]$. We compute the '`maskness`' score defined further below for each of the N soft masks, which serves as a confidence score of each extracted mask. The soft masks are converted to binary masks using a threshold τ. We then sort the binary masks by their maskness scores and remove the redundant masks via mask non-maximum-suppression (NMS). The overall process can be formulated as:

$$\mathbf{M} = \texttt{NMS}\big(\texttt{Maskness}(\texttt{Norm}(\mathbf{Q}' \circledast \mathbf{K}'))\big), \tag{6.3}$$

where $\mathbf{M}$ denotes the object masks that Free Mask outputs.

Self-supervised pre-training. Free Mask uses a pre-trained backbone via self-supervision as the starting point. We propose to leverage the self-supervised model pre-trained with dense correspondence. Specifically, we find that dense contrastive learning (Wang et al., 2021b) achieves considerably better results with our Free Mask approach, compared to the conventional self-supervised learning by global image-level

contrasting. This can be attributed to the similar objective of Free Mask and dense contrastive learning. Here we briefly introduce how the dense contrastive learning is performed. It optimizes a pairwise (dis)similarity loss at the level of local features between two views of the input image. A local feature vector, *i.e.*, a query vector, should be similar to the corresponding positive key in the other view while being dissimilar to other negative keys. Observe that this is also aligned with Equation (6.1) where the cosine similarity between a query and the keys is evaluated. This also explains why Free Mask extracts reasonable masks. We believe that there could be even better pre-training methods for Free Mask, *e.g.*, those which tackle how to learn fine-grained representations at higher resolutions to generate better masks. We leave this for future research.

Pyramid queries. When constructing the queries $\mathbf{Q}$ from $\mathbf{I}$, we design a pyramid queries method to generate masks for instances at different scales. Specifically, we set a list of scale factors, *e.g.*, $[1.0, 0.5, 0.25]$, when downsampling $\mathbf{I}$, thus leading to a list of $\mathbf{Q}$ at different scales from large to small. All pyramid queries are flattened and concatenated together as the final $\mathbf{Q}$.

Maskness score. A scoring function is required for evaluating the quality of each generated coarse mask, which cannot be learned from annotations. We use the non-parametric maskness method (Wang et al., 2020b), *i.e.*, $\texttt{maskness} = \frac{1}{N_f}\sum_i^{N_f} \mathbf{p}_i$, to obtain the confidence score of an extracted mask. Here N_f denotes the number of foreground pixels of the soft mask $\mathbf{p}$, *i.e.*, the pixels that have values greater than threshold τ. Intuitively, this score weighs more heavily on masks that have high confidence on foreground pixels and down weights masks with uncertain foreground pixels.

Unified with SOLO. We can see that the pipeline in Equation (6.3) is unified with that of SOLO, as introduced in the above background section. They both go through FCN, dynamic convolution, normalization and NMS operations to generate object masks. However, the two are proposed to solve different problems. The latter aims to learn instance segmentation with rich annotated data, while the former is for segmenting objects in unlabeled images. This provides a unifying perspective on segmenting objects in images.

6.2.3 Self-Supervised SOLO

We aim to train the SOLO-based instance segmenter using the segmentation masks and semantic embeddings, *i.e.*, feature embeddings with high-level semantics, from Free Mask. We separately introduce the methods for learning with coarse masks, self-training, and the semantic representation learning.

Learning with coarse masks. In SOLO, the Dice loss (Milletari, Navab, and Ahmadi, 2016) is used to supervise the predicted masks with their ground truth labels. However, this is not ideally suited for our case of learning with noisy masks. As the

masks are coarse, directly using them as ground-truth masks can lead to unsatisfactory results. We propose to use the coarse masks as a type of weak annotation and perform weakly supervised instance segmentation with them.

Inspired by the latest weakly-supervised method of BoxInst (Tian et al., 2021), we project the predicted masks and the coarse masks on to the x-axis and the y-axis via a `max` operation along each axis. The model is supervised to minimize the discrepancy between the projections of predicted masks and the coarse masks. The loss term can be defined as:

$$\begin{aligned}\mathcal{L}_{max\_proj} = &\mathcal{L}(\max_x(\boldsymbol{m}), \max_x(\boldsymbol{m}^*)) \\ &+ \mathcal{L}(\max_y(\boldsymbol{m}), \max_y(\boldsymbol{m}^*)),\end{aligned} \tag{6.4}$$

where $\mathcal{L}(\cdot, \cdot)$ is the Dice loss, $\boldsymbol{m}$ and $\boldsymbol{m}^*$ are the predicted mask and the coarse mask. $\max_x$ and $\max_y$ denote the `max` operations along each axis.

We further propose to project the predicted and coarse masks onto the x and y axes via an `average` operation along each axis. The motivation is that the `max` operation may emphasize outlier segmentations in coarse masks, while the `average` operation de-emphasize the outliers. In addition, `average` operation preserves solid shape of the object mask, which can benefit the training. The loss term can be written as:

$$\begin{aligned}\mathcal{L}_{avg\_proj} = &\mathcal{L}(\text{avg}_x(\boldsymbol{m}), \text{avg}_x(\boldsymbol{m}^*)) \\ &+ \mathcal{L}(\text{avg}_y(\boldsymbol{m}), \text{avg}_y(\boldsymbol{m}^*)),\end{aligned} \tag{6.5}$$

where avg_x and avg_y denote the `average` operation along each axis. We also employ a pairwise affinity loss $\mathcal{L}_{pairwise}$ (Tian et al., 2021) to leverage the prior that the proximal pixels are likely to be in the same class, *i.e.*, foreground or background, if they have similar colors in the raw image.

Overall, the total loss for mask prediction can be formulated as:

$$\mathcal{L}_{mask} = \alpha\mathcal{L}_{avg\_proj} + \mathcal{L}_{max\_proj} + \mathcal{L}_{pairwise}, \tag{6.6}$$

where α acts as the weight to balance the various loss terms.

Self-training. With our carefully-designed loss function, we are able to train a SOLO-based instance segmenter with the free and noisy coarse masks. As shown in Figure 6.1, the object masks predicted by the instance segmenter are considerably better than the original coarse masks from Free Mask, which is also validated by the boosted accuracy in Table 6.6(c). As such, we propose to perform self-training with the initially trained instance segmenter to further improve accuracy. We input unlabeled images into the instance segmenter and collect their predicted object masks. The low-confidence predictions are removed and the remaining ones are treated as a new set of coarse masks. We again train an instance segmenter with the unlabeled images and the new masks, using the loss function in Equation (6.6). Performing self-training once already brings clear improvements and more iterations do not provide

additional gains.

Semantic representation learning. General instance segmentation requires not only localizing objects at the pixel level, but also recognizing their semantic categories. In SOLO, the category branch predicts the semantic categories (including background) for each of the objects. In our case without annotations, we propose to decouple the category branch to perform two sub-tasks: foreground/background binary classification, and semantic embedding learning. The former task is trained with the conventional Focal loss (Lin et al., 2017b), termed $\mathcal{L}_{focal}$. For the latter task, we propose a simple approach for learning object-level semantic representations. From Free Mask (introduced in Section 6.2.2), in addition to the segmentation masks, we can also directly obtain the semantic embedding of the discovered objects. As shown in Figure 6.2, each mask is associated with a query feature vector $\mathbf{Q}_q \in \mathbb{R}^E$. When training the instance segmenter, we add a branch in parallel to the last layer of the original category branch, which consists of a single convolution layer to predict the semantic embedding of each object. Given the predicted and extracted embeddings $\boldsymbol{q}$ and $\boldsymbol{q}^*$, we train the model by minimizing their negative cosine similarity:

$$L_{sem} = 1 - \frac{\boldsymbol{q}}{\|\boldsymbol{q}\|_2} \cdot \frac{\boldsymbol{q}^*}{\|\boldsymbol{q}^*\|_2}. \tag{6.7}$$

The total loss for the category branch can be written as:

$$\mathcal{L}_{cate} = \mathcal{L}_{focal} + \beta \mathcal{L}_{sem}, \tag{6.8}$$

where β acts as the weight to balance the two terms. Overall, we train the instance segmenter with a combination of $\mathcal{L}_{mask}$ and $\mathcal{L}_{cate}$, corresponding to the losses for the mask branch and category branch, respectively.

6.3 Experiments

6.3.1 Experimental Settings

Technical details. For Free Mask, the shorter side of the input image is set to 800 pixels. Threshold τ is set to 0.5. DenseCL (Wang et al., 2021b) with a pre-trained ResNet-50 (He et al., 2016b) architecture is adopted as the backbone unless specified. Matrix NMS (Wang et al., 2020b) is used for mask NMS. After NMS, we filter out the low-quality masks with a maskness threshold of 0.7. When training the SOLO model, we initialize the backbone with the pre-trained model used in Free Mask. We set the α and β parameters to 0.1 and 4.0, respectively. We employ the simple copy-paste strategy (Ghiasi et al., 2021) for data augmentation. During self-training, we set the confidence threshold for removing the low-confidence predictions to 0.3.

Datasets. For FreeSOLO, we use the images in COCO `train2017` and COCO `unlabeled2017` (Lin et al., 2014) as the set of unlabeled images, containing a total of

FIGURE 6.3. **Qualitative results of FreeSOLO for the task of class-agnostic instance segmentation.** The model is trained *without any kind of manual annotations* and can infer at 16 FPS on a V100 GPU. Best viewed on screen.

~241k images. These unlabeled images are input to Free Mask and are used to train the instance segmenter. The self-supervised backbone in Free Mask is pre-trained on ImageNet with ~1.28 million unlabeled images. We further employ COCO `val2017`, UVO `val` (Wang et al., 2021a), and PASCAL VOC `trainval07` (Everingham et al., 2010) datasets for evaluation.

Evaluation protocol. We evaluate self-supervised instance segmentation with the standard COCO protocol. We report class-agnostic COCO mask average precision (AP) and average recall (AR) on 5k `val2017` split, which is averaged over 10 intersection-over-union (IoU) thresholds evenly-spaced between 0.5 and 0.95. AP considers recall and precision simultaneously, which computes the average precision value for recall values over 0 to 1. AR allows redundant or random detection results, as it computes the maximum recall given a fixed number of detections per image.

To compare with unsupervised object detection methods, we convert the masks to boxes and report the box AP on both the COCO `val2017`, COCO `20k`, and VOC `trainval07`. We further evaluate the pre-trained model by fine-tuning with annotations. Specifically, we fine-tune the instance segmenter on COCO `train2017` and evaluate on COCO `val2017`. We provide two settings, *i.e.*, limited fully annotated images, and limited segmentation masks. For the setting of limited images, *i.e.*, 5% and 10% images from COCO `train2017`, we train for 20k iterations with an initial learning rate of 0.01, which is then divided by 10 at 12k and 18k iterations. To train with limited mask annotations, we use 5% and 10% segmentation masks from COCO `train2017`. Specifically, we use all the class labels to supervise the category branch, but only use a part of the annotated masks to supervise the mask branch. The model is trained for 90k iterations with the standard schedule. Mask AP averaged across all 10 IoU thresholds and all 80 categories is reported.

method	AP_{50}	AP_{75}	AP	AR_1	AR_{10}	AR_{100}
w/ anns:						
MCG (Arbeláez et al., 2014)	4.6	0.8	1.6	1.9	7.4	18.2
COB (Maninis et al., 2018)	8.8	1.9	3.3	2.9	10.1	22.7
w/o anns:						
FreeSOLO	9.8	2.9	4.0	4.1	10.5	12.7

TABLE 6.1. **Class-agnostic instance segmentation** on MS COCO `val2017`. Both MCG and COB require annotations more or less.

method	AP_{50}	AP_{75}	AP
w/ full anns:			
SOLOv2 w/ COCO	38.0	20.9	21.4
Mask R-CNN w/ COCO	31.0	14.2	15.9
SOLOv2 w/ LVIS	14.8	5.9	7.1
Mask R-CNN w/ LVIS	18.1	4.1	6.8
w/o anns:			
FreeSOLO	12.7	3.0	4.8

TABLE 6.2. **Class-agnostic instance segmentation** on UVO `val` split. Results of Mask R-CNN are from the paper of UVO (Wang et al., 2021a).

6.3.2 Main Results

Self-supervised instance segmentation. For evaluating the self-supervised instance segmenter, we first provide qualitative results to show how FreeSOLO performs at the task of class-agnostic instance segmentation. As shown in Figure 6.3, without any annotations, FreeSOLO is able to segment object instances of many different categories. To provide a quantitative comparison with previous methods, we report the results of unsupervised class-agnostic instance segmentation in Table 6.1 and Table 6.2. As there is no reported result for this new problem, we evaluate a few popular segmentation proposal methods on this benchmark. Among the compared methods, MCG (Arbeláez et al., 2014) uses the annotated BSDS500 dataset (Martin et al., 2001) for training a boundary detector, and COB (Maninis et al., 2018) trains its hierarchies and combinatorial grouping on PASCAL Context dataset (Mottaghi et al., 2014). By contrast, our FreeSOLO method achieves better results without any annotations. We further compare against the supervised methods trained with full annotations. It is worth noting that FreeSOLO even performs closely to the fully supervised Mask R-CNN (He et al., 2017a) trained on the LVIS dataset (Gupta, Dollar, and Girshick, 2019), *e.g.*, 4.8% vs 6.8% AP on the UVO dataset.

Self-supervised object detection. By converting the masks into boxes, our self-supervised instance segmenter naturally serves as a self-supervised object detector as well. We report the results of class-agnostic object detection on COCO `val2017` benchmark in Table 6.3. Our method shows significantly superior performance. To compare with existing object discovery methods, we also evaluate FreeSOLO on VOC

method	AP_{50}	AP_{75}	AP	AR_1	AR_{10}	AR_{100}
UP-DETR (Dai et al., 2021)	0.0	0.0	0.0	0.0	0.0	0.4
Selective Search (Uijlings et al., 2013)	0.5	0.1	0.2	0.2	1.5	10.9
DETReg (Bar et al., 2021)	3.1	0.6	1.0	0.6	3.6	12.7
FreeSOLO	12.2	4.2	5.5	4.6	11.4	15.3

TABLE 6.3. **Unsupervised class-agnostic object detection** on MS COCO `val2017`. Compared results are directly from DETReg.

method	VOC			COCO		
	AP_{50}	AP_{75}	AP	AP_{50}	AP_{75}	AP
Kim et al. (Kim and Torralba, 2009)	9.5	-	2.5	3.9	-	1.0
DDT+ (Wei et al., 2019)	8.7	-	3.0	2.4	-	0.7
rOSD (Vo, Pérez, and Ponce, 2020)	13.1	-	4.3	5.2	-	1.6
LOD (Vo et al., 2021)	13.9	-	4.5	6.6	-	2.0
LOST* (Siméoni et al., 2021)	19.8	-	6.7	7.9	-	2.5
FreeSOLO	24.5	7.2	10.2	12.4	4.4	5.6

TABLE 6.4. **Multi-object discovery** on PASCAL VOC `trainval07` and MS COCO `20k`. LOST* is a concurrent work.

`trainval07` and COCO `20k` for multi-object discovery. As shown in Table 6.4, our method largely outperforms the state-of-the-art object discovery methods, including a concurrent work (Siméoni et al., 2021). Its relative improvements are up to 100% on the COCO dataset.

Supervised fine-tuning. In addition to evaluating the self-supervised instance segmenter directly, we also evaluate the performance of our approach in a supervised setting by fine-tuning the self-supervised instance segmenter with annotations. As shown in Table 6.5, FreeSOLO pre-training outperforms ImageNet supervised pre-training by 4.0% AP when using 5% COCO training images. The gains over the state-of-the-art self-supervised pre-training methods are also clear, *e.g.*, 2.0% AP better than DenseCL (Wang et al., 2021b).

To further compare the pre-training methods with different amount of mask annotations, in Table 6.6, we conduct fine-tuning experiments with only limited masks available. When fine-tuning with 5% masks, FreeSOLO achieves significant gains of 9.8% AP over supervised pre-training. These fine-tuning experiments demonstrate that FreeSOLO serves as a strong instance segmentation pre-training method, outperforming both the supervised and state-of-the-art self-supervised pre-training methods.

6.3.3 Ablation Study

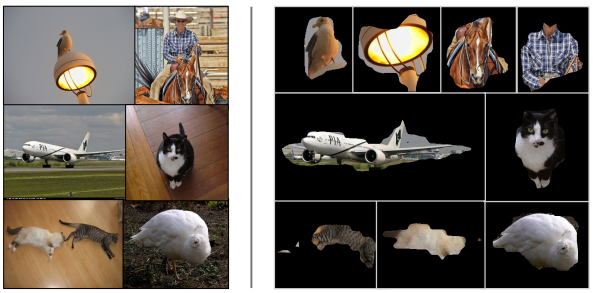
We conduct ablation experiments to show how each component contributes to FreeSOLO. The ablation studies are performed on the COCO `val2017` split.

	pre-train	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
5% images	sup.	18.0	32.2	17.6	5.5	18.9	27.8
	MoCo-v2 (Chen et al., 2020c)	19.0	32.7	19.2	5.4	19.9	28.9
	DenseCL (Wang et al., 2021b)	20.0	33.7	20.5	5.5	21.5	30.1
	FreeSOLO	22.0	36.0	22.9	6.5	23.2	33.8
10% images	sup.	22.3	38.0	22.9	6.3	24.0	34.8
	MoCo-v2 (Chen et al., 2020c)	23.2	39.0	23.9	6.7	24.6	36.2
	DenseCL (Wang et al., 2021b)	23.7	39.3	24.5	7.3	25.2	37.1
	FreeSOLO	25.6	41.6	26.7	8.3	27.5	40.3

TABLE 6.5. **Supervised instance segmentation** with limited fully annotated images.

	pre-train	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
5% masks	sup.	17.8	36.1	15.9	6.3	19.5	27.4
	MoCo-v2 (Chen et al., 2020c)	17.2	34.9	14.9	5.8	19.0	26.2
	DenseCL (Wang et al., 2021b)	20.1	39.0	18.3	7.6	21.4	31.2
	FreeSOLO	29.9	50.5	30.5	10.7	32.5	46.7
10% masks	sup.	25.4	45.6	25.1	8.8	26.9	40.7
	MoCo-v2 (Chen et al., 2020c)	25.6	45.1	25.5	8.7	27.2	40.4
	DenseCL (Wang et al., 2021b)	26.1	45.2	26.3	9.1	28.0	40.8
	FreeSOLO	31.1	51.4	32.0	11.2	34.1	48.4

TABLE 6.6. **Supervised instance segmentation** with limited segmentation masks.



Unlabeled images Free Mask output

FIGURE 6.4. Qualitative results of the Free Mask. Free Mask extracts coarse masks of the common objects in unlabeled images.

Free Mask with different pre-trained backbones. In Table 6.6(a), we show how Free Mask performs with different pre-trained backbones. The conventional self-supervised learning methods that contrast the global representations of image pairs, *e.g.*, SimCLR and MoCo-v2, show worse results compared to supervised ImageNet pre-training. The self-supervised learning methods that consider dense correspondence, *e.g.*, EsViT and DenesCL, yield better results than those that do not. DenseCL shows the best results compared to both supervised and other self-supervised methods. This aligns with our hypothesis in Section 6.2.2 that DenseCL's objective is consistent with Free Mask's. We provide some visualizations of Free Mask in Figure 6.4.

(a) **Different pre-training methods** with Free Mask. DenseCL works the best.

pre-train	AR	AR_S	AR_M	AR_L
sup.	7.8	0.1	11.3	16.4
SimCLR (Chen et al., 2020b)	6.1	1.0	12.1	6.7
MoCo-v2 (Chen et al., 2020c)	4.7	1.6	8.1	5.4
DINO (Caron et al., 2021)	3.2	2.8	5.2	0.9
EsViT (Li et al., 2021)	6.3	0.0	6.0	17.8
DenseCL (Wang et al., 2021b)	11.5	0.1	6.0	39.5

(b) **Pyramid queries** in Free Mask. Pyramid queries improve over single scale queries.

scale	AR	AR_S	AR_M	AR_L
0.25	10.1	0.0	1.9	39.5
1.0	11.3	0.1	6.0	38.6
pyramid	11.5	0.1	6.0	39.5

(c) **Self-training iterations.** '-1' refers to coarse masks. '0' means no self-training.

iters	AP_{50}	AP_{75}	AP
-1	2.3	0.2	0.7
0	7.9	2.5	3.3
1	8.3	2.8	3.7
2	7.7	2.9	3.5

(d) **Mask loss terms.** Each loss component contributes to the final results.

mask loss	AP_{50}	AP_{75}	AP
combination	7.9	2.5	3.3
- w/o $\mathcal{L}_{avg\_proj}$	3.8	1.6	2.0
- w/o $\mathcal{L}_{max\_proj}$	7.1	1.6	2.6
- w/o $\mathcal{L}_{pairwise}$	6.1	0.9	2.1

(e) **Full vs. weak supervision.** Weakly-supervised design is effective.

mask loss	AP_{50}	AP_{75}	AP
full	6.2	1.6	2.4
weak	7.9	2.5	3.3

(f) **Semantic embedding.** Semantic embedding learning improves the fine-tuning results.

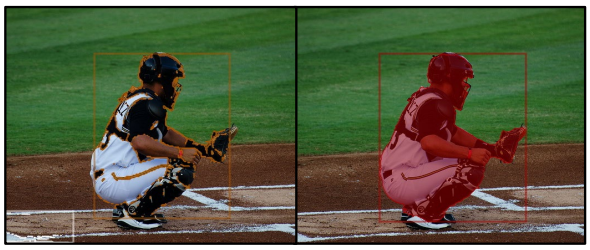
$\mathcal{L}_{sem}$?	AP	AP_{50}	AP_{75}
	24.9	40.5	26.1
✓	25.6	41.6	26.7

TABLE 6.7. **FreeSOLO ablation experiments.** All the experiments are with a ResNet-50 backbone. We report class-agnostic instance segmentation results (a-e) and supervised fine-tuning results (f) on the COCO `val2017` split.

Pyramid queries. We compare different scales of the queries **Q** used in Free Mask in Table 6.6(b). A smaller scale is better for large objects but worse for medium and small objects. A large scale is just the opposite. Pyramid queries with scales $[1.0, 0.5, 0.25]$ yield the best results.

Loss functions. In Table 6.6(e), we compare our weakly-supervised design against the full mask supervision, *i.e.*, the original Dice loss used in SOLO computed with the full masks. Directly using the coarse masks to provide full supervision to the instance segmenter leads to unsatisfactory results. Our weakly-supervised loss outperforms the original full mask loss by a large margin. In Table 6.6(d), we study the mask loss terms in Equation (6.6). The performance drops sharply when learning without $\mathcal{L}_{avg\_proj}$, *i.e.*, with only the projection loss from `max` operation and pairwise loss as in Tian et al. (2021). The model even collapses to only segmenting the contours when trained longer (Figure 6.5). Our method tackles this problem by leveraging the projection from `average` operation, which not only preserves the shape but is also less sensitive to outlier pixels.

Self-training. Our method performs self-training by selecting high-confidence predictions of the self-supervised instance segmenter and training the instance segmenter



w/o $\mathcal{L}_{avg\_proj}$ w/ $\mathcal{L}_{avg\_proj}$

FIGURE 6.5. Qualitative comparison of with and without $\mathcal{L}_{avg\_proj}$ when learning from coarse masks. The model trained without $\mathcal{L}_{avg\_proj}$ tends to only segment the contours when trained longer.

FIGURE 6.6. **More qualitative results of FreeSOLO for the task of class-agnostic instance segmentation.** The model is trained *without any kind of manual annotations* and can infer at 16 FPS on a V100 GPU. Best viewed on screen.

again with them. We compare the results of performing different iterations of self-training in Table 6.6(c). '−1' refers to the initial coarse masks. Zero iteration refers to learning from the coarse masks without self-training. We show that performing self-training once already brings clear improvements, but additional iterations do not provide additional gains.

Semantic embedding. To validate the effectiveness of the semantic embedding learning, in Table 6.6(f) we compare the models trained with or without the semantic embedding loss defined in Equation (6.7). The models are fine-tuned with 10% of fully annotated COCO images. It shows that the semantic embedding loss yields clear improvements when fine-tuning instance segmentation with annotations.

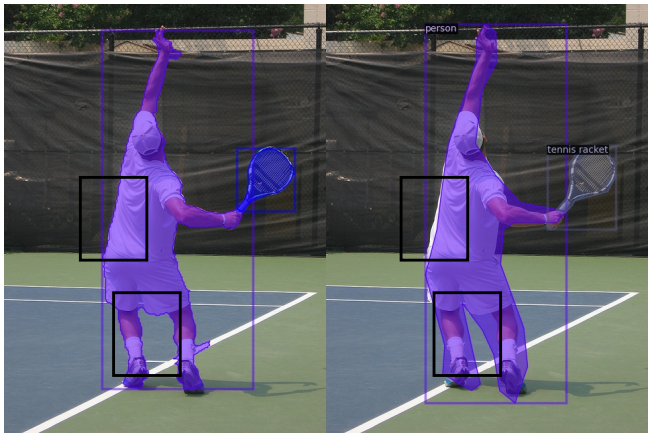


FIGURE 6.7. Qualitative comparison of FreeSOLO's predicted masks and ground truth masks. At some object boundaries, FreeSOLO can produce *even more precise* segmentation than manual annotations in some cases.

FIGURE 6.8. Failure cases of FreeSOLO. Our method could fail to localize objects that are truncated, crowded or small.

Qualitative analysis. We show qualitative results of our method for the task of class-agnostic instance segmentation in Figure 6.6. As shown in Figure 6.7, we further show that FreeSOLO can even produce more precise segmentation results than manual annotations at some object boundaries, which indicates FreeSOLO's great potential for tasks such as auto-labeling. In addition, we show the limitations of our method in Figure 6.8 that it could fail in some scenarios. We believe there is plenty of room to improve based on our method.

6.4 Conclusion

In this work, we have developed a simple and effective self-supervised instance segmentation framework FreeSOLO. FreeSOLO enables learning to segment objects without

any annotations, neither pixel-level nor image-level labels. We hope that its novel design elements provide insights for future works on unsupervised visual learning, *e.g.*, unsupervised panoptic segmentation, and beyond.

Chapter 7

Conclusion

In this thesis, we have proposed a series of novel methods to solve the object segmentation problems with deep neural networks. We further generalize our methods to solve extensive vision problems including object detection, panoptic segmentation, and image matting. We have also explored self-supervised learning for object segmentation, of object segmentation, and by object segmentation.

First, we propose SOLO for simple and direct instance segmentation. SOLO defines a novel concept of "instance category" and assigns a category for each pixel, thus nicely converting the complex instance segmentation to a per-pixel classification problem, which can be solved by a simple fully convolutional network. Without bounding-box detection or pixel grouping post-processing, SOLO directly maps a raw input image to the desired instance masks with constant inference time. The basic principle of SOLO has also inspired and derived several variants including the decoupled and dynamic ones. To our knowledge, SOLO is the first one-stage instance segmentation method that demonstrates a very simple approach achieving on par performance to the dominant "detect-then-segment" method on challenging benchmarks such as MS COCO.

Then, we propose SOLOv2 for high-performance instance segmentation which improves SOLO in terms of both speed and accuracy. In SOLOv2, we first introduce the dynamic mask prediction which uses dynamic convolutions to decouple the original mask head into mask kernel prediction and unified mask feature prediction. It leads to a much more compact yet more powerful mask head design and achieves better results and more accurate boundaries. We further develop a simple Matrix NMS strategy which is much faster and more accurate than the existing strategies, thus largely speeding up the NMS processing of object masks. With a much simpler framework, SOLOv2 outperforms the state-of-the-art instance segmentation methods by a large margin. Besides the superior performance in instance segmentation, we demonstrate the flexibility and effectiveness of SOLOv2 by extending it with minimal modifications to solve object detection, panoptic segmentation, and instance-level image matting problems.

Moreover, to leverage the unlabeled data, we present a self-supervised learning framework DenseCL, which is designed and optimized for object segmentation and other dense prediction tasks. The core idea of DenseCL is to perform dense pairwise contrastive learning at the level of pixels (or local features). After pre-trained on large-scale unlabeled data, the model is transferred to downstream tasks and fine-tuned with limited manual annotations to achieve improved performance. Our method largely closes the gap between self-supervised pre-training and dense prediction tasks. DenseCL pre-training yields significant improvements in a variety of tasks including instance segmentation, semantic segmentation, and object detection.

Finally, built on SOLO, SOLOv2, and DenseCL, we propose FreeSOLO, which learns to segment objects without any annotations. Given a set of unlabeled images, we propose the Free Mask approach to extract the coarse masks using the model pre-trained by dense contrastive learning. These coarse masks are then fed to self-supervised SOLO to train the instance segmentation model with careful designs on loss functions and iterative self-training strategies. Specifically, we leverage coarse masks as a type of weak mask annotations. We use the object embeddings from Free Mask as semantic guidance. And the high-confidence predictions are treated as a new set of coarse masks for self-training. For the first time, we demonstrate unsupervised instance segmentation successfully. In addition, FreeSOLO also serves as a strong pre-training pretext task for instance segmentation.

We believe that, with our fast, label-efficient, and sufficiently strong methods proposed in this thesis, instance segmentation can be a popular alternative to the widely used object bounding box detection. We have seen many works apply our models or built upon our methods for various research topics and real-world problems so far. The future work would be focusing on solving segmentation problems in more challenging open world, *e.g.*, segment everything with open-set categories. We hope that our proposed methods can serve as a cornerstone for object segmentation and many other visual perception tasks, *e.g.*, panoptic segmentation, self-supervised learning, and beyond.

Bibliography

Arbeláez, P., J. Pont-Tuset, J. Barron, F. Marques, and J. Malik (2014). "Multiscale Combinatorial Grouping". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Bar, Amir, Xin Wang, Vadim Kantorov, Colorado J Reed, Roei Herzig, Gal Chechik, Anna Rohrbach, Trevor Darrell, and Amir Globerson (2021). "DETReg: Unsupervised Pretraining with Region Priors for Object Detection". In: *arXiv: Comp. Res. Repository.*

Bodla, Navaneeth, Bharat Singh, Rama Chellappa, and Larry Davis (2017). "Soft-NMS: improving object detection with one line of code". In: *Proc. IEEE Int. Conf. Comp. Vis.*

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

Caron, Mathilde, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin (2021). "Emerging Properties in Self-Supervised Vision Transformers". In: *Proc. IEEE Int. Conf. Comp. Vis.*

Chaitanya, Krishna, Ertunc Erdil, Neerav Karani, and Ender Konukoglu (2020a). "Contrastive learning of global and local features for medical image segmentation with limited annotations". In: *arXiv preprint arXiv:2006.10511.*

Chaitanya, Krishna, Ertunc Erdil, Neerav Karani, and Ender Konukoglu (2020b). "Contrastive learning of global and local features for medical image segmentation with limited annotations". In: *Proc. Advances in Neural Inf. Process. Syst.*

Chen, Hao, Kunyang Sun, Zhi Tian, Chunhua Shen, Yongming Huang, and Youliang Yan (2020a). "BlendMask: Top-Down Meets Bottom-Up for Instance Segmentation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

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.*

Chen, Liang-Chieh, Alexander Hermans, George Papandreou, Florian Schroff, Peng Wang, and Hartwig Adam (2018a). "Masklab: Instance segmentation by refining object detection with semantic and direction features". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Chen, Qifeng, Dingzeyu Li, and Chi-Keung Tang (2013). "KNN matting". In: *IEEE Trans. Pattern Anal. Mach. Intell.*

Chen, Quan, Tiezheng Ge, Yanyu Xu, Zhiqiang Zhang, Xinxin Yang, and Kun Gai (2018b). "Semantic human matting". In: *Proc. ACM Int. Conf. Multimedia*, pp. 618–626.

Chen, Ting, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton (2020b). "A Simple Framework for Contrastive Learning of Visual Representations". In: *Proc. Int. Conf. Mach. Learn.*

Chen, Xinlei, Haoqi Fan, Ross Girshick, and Kaiming He (2020c). "Improved Baselines with Momentum Contrastive Learning". In: *arXiv: Comp. Res. Repository.*

Chen, Xinlei, Ross Girshick, Kaiming He, and Piotr Dollar (2019b). "TensorMask: A Foundation for Dense Object Segmentation". In: *Proc. IEEE Int. Conf. Comp. Vis.*

Chen, Yun-Chun, Yen-Yu Lin, Ming-Hsuan Yang, and Jia-Bin Huang (2021). "Show, Match and Segment: Joint Weakly Supervised Learning of Semantic Matching and Object Co-Segmentation". In: *IEEE Trans. Pattern Anal. Mach. Intell.*

Cheng, Bowen, Maxwell Collins, Yukun Zhu, Ting Liu, Thomas Huang, Hartwig Adam, and Liang-Chieh Chen (2020). "Panoptic-DeepLab: A Simple, Strong, and Fast Baseline for Bottom-Up Panoptic Segmentation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Cheng, Bowen, Omkar Parkhi, and Alexander Kirillov (2021). "Pointly-Supervised Instance Segmentation". In: *arXiv preprint arXiv:2104.06404.*

Cho, Minsu, Suha Kwak, Cordelia Schmid, and Jean Ponce (2015). "Unsupervised object discovery and localization in the wild: Part-based matching with bottom-up region proposals". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Chuang, Yung-Yu, Brian Curless, David Salesin, and Richard Szeliski (2001). "A Bayesian approach to digital matting". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Cordts, Marius, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele (2016). "The cityscapes dataset for semantic urban scene understanding". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Pp. 3213–3223.

Dai, Jifeng, Kaiming He, and Jian Sun (2016). "Instance-aware semantic segmentation via multi-task network cascades". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

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

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

Dai, Zhigang, Bolun Cai, Yugeng Lin, and Junying Chen (2021). "UP-DETR: Unsupervised Pre-Training for Object Detection With Transformers". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

De Brabandere, Bert, Davy Neven, and Luc Van Gool (2017). "Semantic instance segmentation with a discriminative loss function". In: *arXiv:1708.02551.*

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.* Pp. 248–255.

Doersch, Carl, Abhinav Gupta, and Alexei Efros (2015). "Unsupervised visual representation learning by context prediction". In: *Proc. IEEE Int. Conf. Comp. Vis.* Pp. 1422–1430.

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.

Faktor, Alon and Michal Irani (2014). ""Clustering by Composition" - Unsupervised Discovery of Image Categories". In: *IEEE Trans. Pattern Anal. Mach. Intell.*

Gansbeke, Wouter Van, Simon Vandenhende, Stamatios Georgoulis, and Luc Van Gool (2021). "Unsupervised Semantic Segmentation by Contrasting Object Mask Proposals". In: *arXiv: Comp. Res. Repository.*

Gao, Naiyu, Yanhu Shan, Yupei Wang, Xin Zhao, Yinan Yu, Ming Yang, and Kaiqi Huang (2019). "SSAP: Single-Shot Instance Segmentation With Affinity Pyramid". In: *Proc. IEEE Int. Conf. Comp. Vis.*

Ghiasi, Golnaz, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D. Cubuk, Quoc V. Le, and Barret Zoph (2021). "Simple Copy-Paste Is a Strong Data Augmentation Method for Instance Segmentation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Gidaris, Spyros, Praveer Singh, and Nikos Komodakis (2018). "Unsupervised Representation Learning by Predicting Image Rotations". In: *Proc. Int. Conf. Learn. Representations.*

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). "Generative adversarial nets". In: *Proc. Advances in Neural Inf. Process. Syst.* Pp. 2672–2680.

Goyal, Priya, Dhruv Mahajan, Abhinav Gupta, and Ishan Misra (2019). "Scaling and Benchmarking Self-Supervised Visual Representation Learning". In: *Proc. IEEE Int. Conf. Comp. Vis.*

Grill, Jean-Bastien, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. (2020). "Bootstrap your own latent: A new approach to self-supervised learning". In: *arXiv: Comp. Res. Repository.*

Gupta, Agrim, Piotr Dollar, and Ross Girshick (2019). "LVIS: A Dataset for Large Vocabulary Instance Segmentation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Han, Tengda, Weidi Xie, and Andrew Zisserman (2020). "Self-supervised Co-training for Video Representation Learning". In: *Proc. Advances in Neural Inf. Process. Syst.* 33.

Hariharan, Bharath, Pablo Andrés Arbeláez, Ross B. Girshick, and Jitendra Malik (2014). "Simultaneous Detection and Segmentation". In: *Proc. Eur. Conf. Comp. Vis.*

He, Kaiming, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick (2020). "Momentum Contrast for Unsupervised Visual Representation Learning". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

He, Kaiming, Ross Girshick, and Piotr Dollár (2019). "Rethinking imagenet pretraining". In: *Proc. IEEE Int. Conf. Comp. Vis.* Pp. 4918–4927.

He, Kaiming, Georgia Gkioxari, Piotr Dollár, and Ross Girshick (2017a). "Mask R-CNN". In: *Proc. IEEE Int. Conf. Comp. Vis.* Pp. 2961–2969.

He, Kaiming, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick (2017b). "Mask R-CNN". In: *Proc. IEEE Int. Conf. Comp. Vis.*

He, Kaiming, Christoph Rhemann, Carsten Rother, Xiaoou Tang, and Jian Sun (2011). "A global sampling method for alpha matting". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Pp. 2049–2056.

He, Kaiming, Jian Sun, and Xiaoou Tang (2010). "Fast matting using large kernel matting laplacian matrices". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

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

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

Hénaff, Olivier J., Skanda Koppula, Jean-Baptiste Alayrac, Aäron van den Oord, Oriol Vinyals, and João Carreira (2021). "Efficient Visual Pretraining with Contrastive Detection". In: *Proc. IEEE Int. Conf. Comp. Vis.*

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.*

Hsu, Kuang-Jui, Yen-Yu Lin, and Yung-Yu Chuang (2018). "Co-attention CNNs for Unsupervised Object Co-segmentation". In: *Proc. Int. Joint Conf. Artificial Intell.*

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

Hwang, Jyh-Jing, Stella X. Yu, Jianbo Shi, Maxwell D. Collins, Tien-Ju Yang, Xiao Zhang, and Liang-Chieh Chen (2019). "SegSort: Segmentation by Discriminative Sorting of Segments". In: *Proc. IEEE Int. Conf. Comp. Vis.*

Islam*, Md Amirul, Sen Jia*, and Neil D. B. Bruce (2020). "How much Position Information Do Convolutional Neural Networks Encode?" In: *Proc. Int. Conf. Learn. Representations.*

Ji, Xu, Andrea Vedaldi, and João F. Henriques (2019). "Invariant Information Clustering for Unsupervised Image Classification and Segmentation". In: *Proc. IEEE Int. Conf. Comp. Vis.*

Joulin, Armand, Francis R. Bach, and Jean Ponce (2010). "Discriminative clustering for image co-segmentation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Khoreva, Anna, Rodrigo Benenson, Jan Hendrik Hosang, Matthias Hein, and Bernt Schiele (2017). "Simple Does It: Weakly Supervised Instance and Semantic Segmentation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Kim, Gunhee and Antonio Torralba (2009). "Unsupervised Detection of Regions of Interest Using Iterative Link Analysis". In: *Proc. Advances in Neural Inf. Process. Syst.*

Kirillov, Alexander, Ross Girshick, Kaiming He, and Piotr Dollár (2019a). "Panoptic feature pyramid networks". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Kirillov, Alexander, Kaiming He, Ross B. Girshick, Carsten Rother, and Piotr Dollár (2019b). "Panoptic Segmentation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Kong, Tao, Fuchun Sun, Huaping Liu, Yuning Jiang, Lei Li, and Jianbo Shi (2020). "FoveaBox: Beyond Anchor-based Object Detector". In: *IEEE Trans. Image Process.*, pp. 7389–7398.

Kong, Tao, Anbang Yao, Yurong Chen, and Fuchun Sun (2016). "HyperNet: Towards accurate region proposal generation and joint object detection". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Pp. 845–853.

Lan, Shiyi, Zhiding Yu, Christopher Choy, Subhashree Radhakrishnan, Guilin Liu, Yuke Zhu, Larry S Davis, and Anima Anandkumar (2021). "DiscoBox: Weakly Supervised Instance Segmentation and Semantic Correspondence from Box Supervision". In: *Proc. IEEE Int. Conf. Comp. Vis.*

Levin, Anat, Dani Lischinski, and Yair Weiss (2007). "A closed-form solution to natural image matting". In: *IEEE Trans. Pattern Anal. Mach. Intell.*

Li, Chunyuan, Jianwei Yang, Pengchuan Zhang, Mei Gao, Bin Xiao, Xiyang Dai, Lu Yuan, and Jianfeng Gao (2021). "Efficient Self-supervised Vision Transformers for Representation Learning". In: *arXiv: Comp. Res. Repository.*

Li, Hengduo, Bharat Singh, Mahyar Najibi, Zuxuan Wu, and Larry S. Davis (2019a). "An analysis of pre-training on object detection". In: *arXiv: Comp. Res. Repository.*

Li, Yanwei, Xinze Chen, Zheng Zhu, Lingxi Xie, Guan Huang, Dalong Du, and Xingang Wang (2019b). "Attention-guided unified network for panoptic segmentation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Li, Yi, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei (2017). "Fully Convolutional Instance-Aware Semantic Segmentation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Li, Zeming, Chao Peng, Gang Yu, Xiangyu Zhang, Yangdong Deng, and Jian Sun (2018). "DetNet: Design backbone for object detection". In: *Proc. Eur. Conf. Comp. Vis.* Pp. 334–350.

Lin, Tsung-Yi, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie (2017a). "Feature Pyramid Networks for Object Detection". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

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

Lin, Tsung-Yi, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick (2014). "Microsoft COCO: Common Objects in Context". In: *Proc. Eur. Conf. Comp. Vis.*

Liu, Jinlin, Yuan Yao, Wendi Hou, Miaomiao Cui, Xuansong Xie, Changshui Zhang, and Xian-Sheng Hua (2020a). "Boosting Semantic Human Matting With Coarse Annotations". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Liu, Rosanne, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski (2018a). "An intriguing failing of convolutional neural networks and the coordconv solution". In: *Proc. Advances in Neural Inf. Process. Syst.*

Liu, Shu, Jiaya Jia, Sanja Fidler, and Raquel Urtasun (2017). "Sequential Grouping Networks for Instance Segmentation". In: *Proc. IEEE Int. Conf. Comp. Vis.*

Liu, Shu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia (2018b). "Path Aggregation Network for Instance Segmentation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

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

Liu, Yun, Yu-Huan Wu, Peisong Wen, Yujun Shi, Yu Qiu, and Ming-Ming Cheng (2020b). "Leveraging Instance-, Image- and Dataset-Level Information for Weakly Supervised Instance Segmentation". In: *IEEE Trans. Pattern Anal. Mach. Intell.*

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

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

Lu, Hao, Yutong Dai, Chunhua Shen, and Songcen Xu (2019). "Indices matter: Learning to index for deep image matting". In: *Proc. IEEE Int. Conf. Comp. Vis.*

Mahajan, Dhruv, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten (2018). "Exploring the limits of weakly supervised pretraining". In: *Proc. Eur. Conf. Comp. Vis.* Pp. 181–196.

Maninis, Kevis-Kokitsi, Jordi Pont-Tuset, Pablo Arbeláez, and Luc Van Gool (2018). "Convolutional Oriented Boundaries: From Image Segmentation to High-Level Tasks". In: *IEEE Trans. Pattern Anal. Mach. Intell.*

Martin, David R., Charless C. Fowlkes, Doron Tal, and Jitendra Malik (2001). "A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics". In: *Proc. IEEE Int. Conf. Comp. Vis.*

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.*

Mottaghi, Roozbeh, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan L. Yuille (2014). "The Role of Context for Object Detection and Semantic Segmentation in the Wild". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

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.*

Noroozi, Mehdi and Paolo Favaro (2016). "Unsupervised learning of visual representations by solving jigsaw puzzles". In: *Proc. Eur. Conf. Comp. Vis.* Pp. 69–84.

Novotný, David, Samuel Albanie, Diane Larlus, and Andrea Vedaldi (2018). "Semi-convolutional Operators for Instance Segmentation". In: *Proc. Eur. Conf. Comp. Vis.*

Oord, Aaron van den, Yazhe Li, and Oriol Vinyals (2018). "Representation learning with contrastive predictive coding". In: *arXiv: Comp. Res. Repository.*

OpenMMLab (2020). *mmsegmentation.* https://github.com/open-mmlab/mmsegmentation.

Pathak, Deepak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei Efros (2016). "Context encoders: Feature learning by inpainting". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Pp. 2536–2544.

Pinheiro, Pedro, Amjad Almahairi, Ryan Y Benmaleck, Florian Golemo, and Aaron Courville (2020a). "Unsupervised Learning of Dense Visual Representations". In: *arXiv: Comp. Res. Repository.*

Pinheiro, Pedro H. O., Ronan Collobert, and Piotr Dollár (2015). "Learning to Segment Object Candidates". In: *Proc. Advances in Neural Inf. Process. Syst.*

Pinheiro, Pedro O., Amjad Almahairi, Ryan Y. Benmalek, Florian Golemo, and Aaron C. Courville (2020b). "Unsupervised Learning of Dense Visual Representations". In: *Proc. Advances in Neural Inf. Process. Syst.*

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 (2018a). "YOLOv3: An Incremental Improvement". In: *arXiv.org*, pp. 1–6. URL: http://arxiv.org/abs/1804.02767v1.

Redmon, Joseph and Ali Farhadi (2018b). "Yolov3: An incremental improvement". In: *arXiv:1804.02767*.

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

Ren, Shaoqing, Kaiming He, Ross B. Girshick, and Jian Sun (2015b). "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Proc. Advances in Neural Inf. Process. Syst.*

Russell, Bryan C., William T. Freeman, Alexei A. Efros, Josef Sivic, and Andrew Zisserman (2006). "Using Multiple Segmentations to Discover Objects and their Extent in Image Collections". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Sengupta, Soumyadip, Vivek Jayaram, Brian Curless, Steve Seitz, and Ira Kemelmacher-Shlizerman (2020). "Background Matting: The World is Your Green Screen". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Shen, Xiaoyong, Xin Tao, Hongyun Gao, Chao Zhou, and Jiaya Jia (2016). "Deep automatic portrait matting". In: *Proc. Eur. Conf. Comp. Vis.* Pp. 92–107.

Siméoni, Oriane, Gilles Puy, Huy V. Vo, Simon Roburin, Spyros Gidaris, Andrei Bursuc, Patrick Pérez, Renaud Marlet, and Jean Ponce (2021). "Localizing Objects with Self-Supervised Transformers and no Labels". In: *arXiv: Comp. Res. Repository.*

Sivic, Josef, Bryan C. Russell, Alexei A. Efros, Andrew Zisserman, and William T. Freeman (2005). "Discovering object categories in image collections". In: *Proc. IEEE Int. Conf. Comp. Vis.*

Sofiiuk, Konstantin, Olga Barinova, and Anton Konushin (2019). "AdaptIS: Adaptive Instance Selection Network". In: *Proc. IEEE Int. Conf. Comp. Vis.*

Sun, Ke, Bin Xiao, Dong Liu, and Jingdong Wang (2019). "Deep high-resolution representation learning for human pose estimation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Pp. 5693–5703.

Tan, Mingxing, Ruoming Pang, and Quoc V Le (2020). "EfficientDet: Scalable and efficient object detection". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.* Pp. 10781–10790.

Tian, Yonglong, Dilip Krishnan, and Phillip Isola (2019). "Contrastive multiview coding". In: *arXiv preprint arXiv:1906.05849.*

Tian, Yonglong, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola (2020). "What makes for good views for contrastive learning". In: *NeurIPS.*

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 (2019). "FCOS: Fully Convolutional One-Stage Object Detection". In: *Proc. IEEE Int. Conf. Comp. Vis.*

Tian, Zhi, Chunhua Shen, Xinlong Wang, and Hao Chen (2021). "BoxInst: High-Performance Instance Segmentation with Box Annotations". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Uijlings, Jasper R. R., Koen E. A. van de Sande, Theo Gevers, and Arnold W. M. Smeulders (2013). "Selective Search for Object Recognition". In: *Int. J. Comput. Vision.*

Vo, Huy V., Francis R. Bach, Minsu Cho, Kai Han, Yann LeCun, Patrick Pérez, and Jean Ponce (2019). "Unsupervised Image Matching and Object Discovery as Optimization". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Vo, Huy V, Patrick Pérez, and Jean Ponce (2020). "Toward unsupervised, multi-object discovery in large-scale image collections". In: *Proc. Eur. Conf. Comp. Vis.*

Vo, Huy V., Elena Sizikova, Cordelia Schmid, Patrick Pérez, and Jean Ponce (2021). "Large-Scale Unsupervised Object Discovery". In: *arXiv: Comp. Res. Repository.*

Wang, Weiyao, Matt Feiszli, Heng Wang, and Du Tran (2021a). "Unidentified Video Objects: A Benchmark for Dense, Open-World Segmentation". In: *arXiv: Comp. Res. Repository.*

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

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, Chunhua Shen, Tao Kong, and Lei Li (2021b). "Dense Contrastive Learning for Self-Supervised Visual Pre-Training". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Wang, Xinlong, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li (2021c). "SOLO: A Simple Framework for Instance Segmentation". In: *IEEE Trans. Pattern Anal. Mach. Intell.*

Wang, Yuqing, Zhaoliang Xu, Hao Shen, Baoshan Cheng, and Lirong Yang (2020c). "CenterMask: single shot instance segmentation with point representation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Wei, Xiu-Shen, Chen-Lin Zhang, Jianxin Wu, Chunhua Shen, and Zhi-Hua Zhou (2019). "Unsupervised object discovery and co-localization by deep descriptor transformation". In: *Pattern Recogn.*

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

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

Wu, Zhirong, Yuanjun Xiong, Stella Yu, and Dahua Lin (2018). "Unsupervised feature learning via non-parametric instance discrimination". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Xie, Enze, Peize Sun, Xiaoge Song, Wenhai Wang, Xuebo Liu, Ding Liang, Chunhua Shen, and Ping Luo (2020a). "PolarMask: Single Shot Instance Segmentation with Polar Representation". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Xie, Saining, Jiatao Gu, Demi Guo, Charles R Qi, Leonidas J Guibas, and Or Litany (2020b). "PointContrast: Unsupervised Pre-training for 3D Point Cloud Understanding". In: *arXiv preprint arXiv:2007.10985*.

Xiong, Yuwen, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun (2019). "UPSNet: A unified panoptic segmentation network". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Xu, Ning, Brian Price, Scott Cohen, and Thomas Huang (2017). "Deep image matting". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

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

Zhan, Xiaohang, Jiahao Xie, Ziwei Liu, Yew-Soon Ong, and Chen Change Loy (2020). "Online Deep Clustering for Unsupervised Representation Learning". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Zhang, Richard, Phillip Isola, and Alexei Efros (2016). "Colorful image colorization". In: *Proc. Eur. Conf. Comp. Vis.* Pp. 649–666.

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, Shifeng, Longyin Wen, Xiao Bian, Zhen Lei, and Stan Z Li (2018). "Single-shot refinement neural network for object detection". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Zhang, Yunke, Lixue Gong, Lubin Fan, Peiran Ren, Qixing Huang, Hujun Bao, and Weiwei Xu (2019). "A late fusion CNN for digital matting". In: *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*

Zhao, Nanxuan, Zhirong Wu, Rynson Lau, and Stephen Lin (2020). "What makes instance discrimination good for transfer learning?" In: *arXiv preprint arXiv:2006.06606*.

Zhou, Dongzhan, Xinchi Zhou, Hongwen Zhang, Shuai Yi, and Wanli Ouyang (2020). "Cheaper Pre-training Lunch: An Efficient Paradigm for Object Detection". In: *arXiv preprint arXiv:2004.12178*.