

© 2022 Bowen Cheng

FROM PIXELS TO REGIONS:
TOWARD UNIVERSAL IMAGE SEGMENTATION

BY

BOWEN CHENG

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois Urbana-Champaign, 2022

Urbana, Illinois

Doctoral Committee:

Assistant Professor Alexander Schwing, Chair
Adjunct Assistant Professor Humphrey Shi
Professor Mark Hasegawa-Johnson
Professor Trevor Darrell, University of California Berkeley
Professor Zhi-Pei Liang

ABSTRACT

Image segmentation is about grouping pixels with different semantics, *e.g.*, category or instance membership, where each choice of semantics defines a task. While only the semantics of each task differ, current research focuses on designing specialized architectures for each task: semantic segmentation is usually formulated as per-pixel classification and mask classification dominates instance-level segmentation tasks. In this dissertation, we demonstrate how to build a single unified architecture that can address any image segmentation task. We first introduce an effort in unifying image segmentation with either per-pixel classification (Panoptic-DeepLab) or mask classification (MaskFormer). We observe mask classification is sufficiently general to solve both semantic- and instance-level segmentation tasks. Based on this observation we propose Mask2Former, which outperforms even the best specialized architectures by a significant margin on four popular datasets for three image segmentation tasks (panoptic, instance and semantic). Then we discuss how to evaluate image segmentation models with a new Boundary IoU metric. Finally, we conclude this dissertation with promising future directions to explore.

To my parents, for their love and support.

ACKNOWLEDGMENTS

I would like to thank my PhD advisors, my mentors during internships, my collaborators, and my friends and family for their help and support during my PhD. I would like to express my deepest gratitude to my advisors Professor Alexander G. Schwing and Professor Thomas S. Huang for guiding me into computer vision research. I would like to thank Professor Humphrey Shi, Professor Mark Hasegawa-Johnson, Professor Trevor Darrell and Professor Zhi-Pei Liang for being on my committee. I am also grateful to Dr. Alexander Kirillov for his guidance and support during my internship at Facebook AI Research for two years. I have learned a lot from my interaction with all these great advisors and mentors.

I would like to express my thanks to my previous intern mentors: Dr. Alexander Kirillov, Dr. Alexander Berg, Dr. Ross Girshick, Dr. Piotr Dollar, Dr. Rohit Girdhar, and Dr. Ishan Misra (Facebook AI Research); Dr. Liang-Chieh Chen and Dr. Yukun Zhu (Google Research); Dr. Lei Zhang, Dr. Jianfeng Wang and Dr. Rong Xiao (Microsoft Research); and Dr. Jifeng Dai, Dr. Yichen Wei, Dr. Bin Xiao, Dr. Han Hu, and Dr. Zheng Zhang (Microsoft Research Asia). The experiences with them were valuable and these experiments made me a good researcher.

I would also like to thank my lab and office mates at UIUC for their help throughout the year. Importantly, I would like to thank Dr. Zhangyang Wang, Dr. Humphrey Shi, and Dr. Yunchao Wei for their guidance when I was a junior PhD student. I thank Dr. Jiahui Yu and Dr. Yuchen Fan for sharing their personal experience. I thank Dr. Haichao Yu, Dr. Hanchao Yu, Dr. Yuqian Zhou, Yang Fu, Zhonghao Wang, Zhongzheng Ren, and Xiaoming Zhao for all the wonderful dinner time.

Finally, I would like to thank my family for their support throughout my PhD.

TABLE OF CONTENTS

CHAPTER 1	INTRODUCTION	1
1.1	Motivation	1
1.2	Overview	2
CHAPTER 2	IMAGE SEGMENTATION WITH PER-PIXEL CLASSIFICATION	5
2.1	Introduction	5
2.2	Related Works	7
2.3	Panoptic-DeepLab	9
2.4	Experiments	12
2.5	Conclusion	24
2.6	Supplementary Materials	24
CHAPTER 3	IMAGE SEGMENTATION WITH MASK CLASSIFICATION	33
3.1	Introduction	33
3.2	Related Works	35
3.3	From Per-Pixel to Mask Classification	36
3.4	Experiments	41
3.5	Discussion	50
3.6	Conclusion	51
3.7	Supplementary Materials	51
CHAPTER 4	UNIVERSAL IMAGE SEGMENTATION	60
4.1	Introduction	60
4.2	Related Work	62
4.3	Masked-attention Mask Transformer	64
4.4	Experiments	69
4.5	Conclusion	80
4.6	Supplementary Materials	80
CHAPTER 5	IMAGE SEGMENTATION EVALUATION	97
5.1	Introduction	97
5.2	Preliminaries	100
5.3	Sensitivity Analysis	103

5.4	Analysis of Existing Segmentation Measures	104
5.5	Boundary IoU	111
5.6	Applications	113
5.7	Conclusion	117
5.8	Supplementary Materials	118
CHAPTER 6 CONCLUSION AND FUTURE WORK		128
6.1	Conclusion	128
6.2	Future Work	128
REFERENCES		131

CHAPTER 1

INTRODUCTION

I stand at the window and see a house, trees, sky. Theoretically I might say there were 327 brightnesses and nuances of color.

Do I have “327”?

No, I have sky, house, and trees.

Max Wertheimer

1.1 Motivation

Humans have the ability to perceive the world as a set of semantically meaningful objects. Such an ability to group parts into a whole is usually referred to as *perceptual grouping*. The study of perceptual grouping can be dated back to the 1920s when Wertheimer first introduced the Gestalt principles for grouping [1]. Without perceptual grouping we would only be able to see single elements, thus perceptual grouping is essential in the human perception system.

How do we augment an AI agent with the ability to do perceptual grouping? We believe that we can find an answer via studying image segmentation, a computer vision task that is closely related to perceptual grouping. Image segmentation is the task of grouping pixels into meaningful regions. However, image segmentation by itself is an ill-posed problem, because there exist many possible partitions of the same image. To reduce the ambiguity in this problem statement, many better-posed image segmentation tasks have been proposed by clearly defining the criterion to partition an image. For example, semantic segmentation [2] groups pixels by their category memberships.

Instance segmentation [3] groups pixels by their instance membership. And more recently, panoptic segmentation [4] tries to unify both semantic and instance segmentation by grouping pixels with both category and instance membership.

Although image segmentation is about grouping pixels, these different semantics for grouping pixels have led to specialized architectures.¹ For semantic segmentation, per-pixel classification architectures are developed to predict a category label for every pixel. In contrast, instance segmentation usually follows a mask classification paradigm [5]. Instead of classifying each pixel, mask classification-based methods predict a set of binary masks, each associated with a single class prediction. Although these specialized architectures have advanced each individual task, they lack the flexibility to generalize to the other tasks. For example, it is non-trivial to use a per-pixel classification architecture for instance segmentation, leading to the evolution of different architectures for instance segmentation compared to semantic segmentation. Thus, duplicate research and (hardware) optimization effort are spent on each specialized architecture for every task.

Thus, we ask the question: Does there exist a single architecture that can address any image segmentation task? In this dissertation, we will refer to the ability to address any image segmentation task as universal image segmentation. The single architecture addressing the task is referred to as universal architecture. More specifically, our goal is to use a single architecture for multiple segmentation tasks without having to change the architecture, the loss or the training pipeline while outperforming the best specialized architectures for different segmentation tasks.

1.2 Overview

In Chapter 2, we start by exploring unifying image segmentation tasks with a per-pixel classification paradigm. We introduce Panoptic-DeepLab [6], a simple, strong, and fast system for panoptic segmentation, aiming to establish a solid baseline for bottom-up methods that can achieve comparable

¹In this dissertation, we separate the concepts of an architecture and a model: An architecture refers to the structure of a network consisting of layers or modules that perform operations on data, and a model refers to the specific set of weights or parameters of an architecture.

performance of two-stage methods while yielding fast inference speed. In particular, Panoptic-DeepLab adopts the dual-ASPP (Atrous Spatial Pyramid Pooling [7]) and dual-decoder structures specific to semantic, and instance segmentation, respectively. The semantic segmentation branch is the same as the typical design of any semantic segmentation model (*e.g.*, DeepLab), while the instance segmentation branch is class-agnostic, involving a simple instance center regression. As a result, our single Panoptic-DeepLab simultaneously ranks first at all three Cityscapes benchmarks, setting the new state-of-the-art of 84.2 mIoU, 39.0 AP, and 65.5 PQ on test set. Additionally, equipped with MobileNetV3, Panoptic-DeepLab runs nearly in real-time with a single 1025×2049 image (15.8 frames per second), while achieving a competitive performance on Cityscapes (54.1 PQ on test set). On Mapillary Vistas test set, our ensemble of six models attains 42.7 PQ, outperforming the challenge winner in 2018 by a healthy margin of 1.5. Finally, our Panoptic-DeepLab also performs on par with several top-down approaches on the challenging COCO dataset. For the first time, we demonstrate that a bottom-up approach can deliver state-of-the-art results on panoptic segmentation.

Although per-pixel classification models show promising performance, we find they have a hard time handling large number of classes, making them not suitable for unifying image segmentation tasks. In Chapter 3, we explore unifying image segmentation tasks with a mask classification paradigm. Our key insight: mask classification is sufficiently general to solve both semantic- and instance-level segmentation tasks in a unified manner using the exact same model, loss, and training procedure. Following this observation, we propose MaskFormer [8], a simple mask classification model which predicts a set of binary masks, each associated with a single global class label prediction. Overall, the proposed mask classification-based method simplifies the landscape of effective approaches to semantic and panoptic segmentation tasks and shows excellent empirical results. In particular, we observe that MaskFormer outperforms per-pixel classification baselines when the number of classes is large. Our mask classification-based method outperforms both current state-of-the-art semantic (55.6 mIoU on ADE20K) and panoptic segmentation (52.7 PQ on COCO) models.

As mentioned before, we want a single architecture for any image segmentation task. In Chapter 4, we present Masked-attention Mask Transformer

(Mask2Former) [9], a new architecture capable of addressing any image segmentation task (panoptic, instance or semantic). Its key components include masked attention, which extracts localized features by constraining cross-attention within predicted mask regions. In addition to reducing the research effort by at least three times, it outperforms the best specialized architectures by a significant margin on four popular datasets. Most notably, Mask2Former sets a new state-of-the-art for panoptic segmentation (57.8 PQ on COCO), instance segmentation (50.1 AP on COCO) and semantic segmentation (57.7 mIoU on ADE20K).

How do we know whether current evaluation metrics for comparing segmentation models are reliable? In Chapter 5, we present Boundary IoU (Intersection-over-Union) [10], a new segmentation evaluation measure focused on boundary quality. We perform an extensive analysis across different error types and object sizes and show that Boundary IoU is significantly more sensitive than the standard Mask IoU measure to boundary errors for large objects and does not over-penalize errors on smaller objects. The new quality measure displays several desirable characteristics like symmetry w.r.t. prediction/ground truth pairs and balanced responsiveness across scales, which makes it more suitable for segmentation evaluation than other boundary-focused measures like Trimap IoU and F-measure. Based on Boundary IoU, we update the standard evaluation protocols for instance and panoptic segmentation tasks by proposing the Boundary AP (Average Precision) and Boundary PQ (Panoptic Quality) metrics, respectively. Our experiments show that the new evaluation metrics track boundary quality improvements that are generally overlooked by current Mask IoU-based evaluation metrics. We hope that the adoption of the new boundary-sensitive evaluation metrics will lead to rapid progress in segmentation methods that improve boundary quality.

CHAPTER 2

IMAGE SEGMENTATION WITH PER-PIXEL CLASSIFICATION

2.1 Introduction

Panoptic segmentation, unifying semantic segmentation and instance segmentation, has received a lot of attention thanks to the recently proposed panoptic quality metric [4] and associated recognition challenges [11–13]. The goal of panoptic segmentation is to assign a unique value, encoding both semantic label and instance id, to every pixel in an image. It requires identifying the class and extent of each individual “thing” in the image, and labelling all pixels that belong to each “stuff” class.

The task of panoptic segmentation introduces challenges that preceding methods are unsuited to solve. Models typically used in the separate instance and semantic segmentation literature have diverged, and fundamentally different approaches dominate in each setting. For panoptic segmentation, the top-down methods [14–18], attaching another semantic segmentation branch to Mask R-CNN [5], generate overlapping instance masks as well as duplicate pixel-wise semantic predictions. To settle the conflict, the commonly employed heuristic resolves overlapping instance masks by their predicted confidence scores [4], or even by the pairwise relationship between categories [17] (*e.g.*, *ties* should be always in front of *person*). Additionally, the discrepancy between semantic and instance segmentation results are sorted out by favoring the instance predictions. Though effective, it may be hard to implement the hand-crafted heuristics in a fast and parallel fashion. Another effective way is to develop advanced modules to fuse semantic and instance segmentation results [14, 16, 17]. However, these top-down methods are usually slow in speed, resulted from the multiple sequential processes in the pipeline.

On the other hand, bottom-up methods naturally resolve the conflict by predicting non-overlapping segments. Only few works [19, 20] adopt the

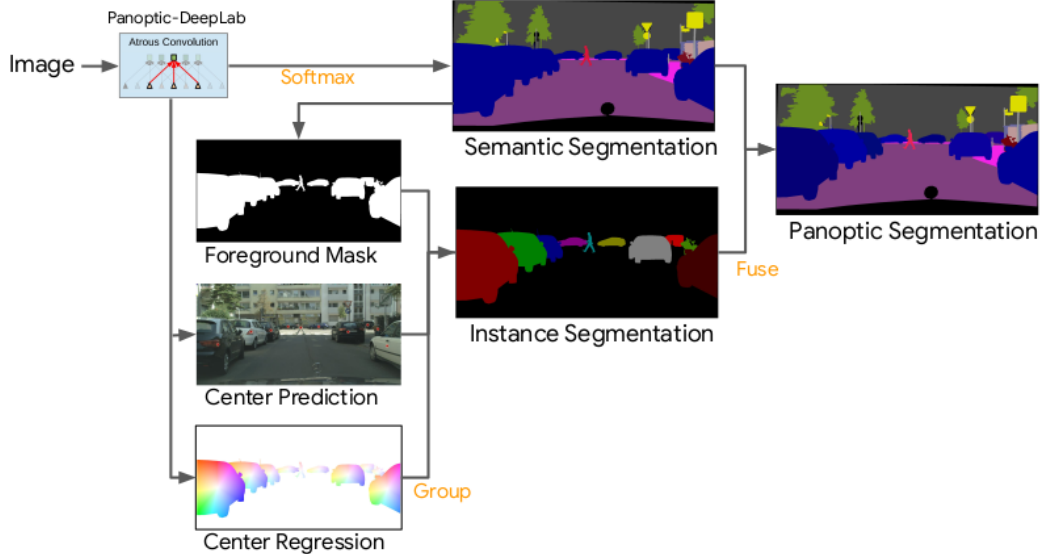


Figure 2.1: Our Panoptic-DeepLab predicts three outputs: semantic segmentation, instance center prediction and instance center regression. Class-agnostic instance segmentation, obtained by grouping predicted foreground pixels to their closest predicted instance centers, is then fused with semantic segmentation by majority-vote rule to generate final panoptic segmentation.

bottom-up approach, which typically starts with a semantic segmentation prediction followed by grouping operations to generate instance masks. Tackling panoptic segmentation in such a sequential order allows a simple and fast scheme, such as majority vote [19], to merge semantic and instance segmentation results. Although obtaining promising fast inference speed, bottom-up approaches still demonstrate inferior performance compared to top-down ones prevailing in public benchmarks [11–13].

The difficulties faced by top-down methods, and the dearth of previous investigations into complementary approaches motivate us to establish a simple, strong, and fast bottom-up baseline for panoptic segmentation. Our proposed **Panoptic-DeepLab** (Figure 2.1) requires only three loss functions during training, and introduces extra marginal parameters as well as additional slight computation overhead when building on top of a modern semantic segmentation model. The design of the proposed Panoptic-DeepLab is conceptually simple, adopting dual-ASPP and dual-decoder modules specific to semantic segmentation and instance segmentation, respectively. The semantic segmentation branch follows the typical design of any semantic

segmentation model (*e.g.*, DeepLab [21]), while the instance segmentation branch involves a simple instance center regression [22, 23], where the model learns to predict instance centers as well as the offset from each pixel to its corresponding center, resulting in an extremely simple grouping operation by assigning pixels to their closest predicted center. Additionally, with fast GPU implementation of the merging operation, Panoptic-DeepLab delivers near real-time end-to-end panoptic segmentation prediction.

We conduct experiments on several popular panoptic segmentation datasets. On Cityscapes test set [12], a *single* Panoptic-DeepLab model (without fine-tuning on different tasks) achieves state-of-the-art performance of 65.5% PQ, 39.0% AP, and 84.2% mIoU, simultaneously ranking first on *all* three Cityscapes tasks when comparing with published works. On Mapillary Vistas [13], our best *single* model attains 40.6% PQ on val set, while employing an ensemble of 6 models reaches a performance of 42.2% PQ on val set and 42.7% PQ on test set, outperforming the winner of Mapillary Vistas Panoptic Segmentation Challenge in 2018 by a healthy margin of 1.5% PQ. For the first time, we show a bottom-up approach can deliver state-of-the-art panoptic segmentation results on both Cityscapes and Mapillary Vistas. On COCO [11] test-dev set, our Panoptic-DeepLab also demonstrates state-of-the-art results, performing on par with several top-down approaches. Finally, we provide extensive experimental results and disclose every detail in our system. We hope our Panoptic-DeepLab can serve as a solid baseline to facilitate the research on panoptic segmentation, especially from the bottom-up perspective.

2.2 Related Works

We categorize current panoptic segmentation methods [4] into two groups: top-down and bottom-up approaches.

Top-down: Most state-of-the-art methods tackle panoptic segmentation from the top-down or proposal-based perspective. These methods are often referred to as two-stage methods because they require an additional stage to generate proposals. Specifically, Mask R-CNN [5] is commonly deployed to extract *overlapping instances*, followed by some post-processing methods to resolve mask overlaps. The remaining regions are then filled by a lightweight

stuff segmentation branch. For example, TASCNet [16] learns a binary mask to enforce the consistency between “thing” and “stuff” predictions. Liu *et al.* [24] propose the Spatial Ranking module to resolve the overlapping instance masks. AUNet [17] introduces attention modules to guide the fusion between “thing” and “stuff” segmentation. Panoptic FPN [15] endows Mask R-CNN [5] with a semantic segmentation branch. UPSNet [14] develops a parameter-free panoptic head which resolves the conflicts in “thing”-“stuff” fusion by predicting an extra unknown class. Porzi *et al.* [18] integrate the multi-scale features from FPN [25] with a lightweight DeepLab-inspired module [26]. AdaptIS [27] generates instance masks with point proposals.

Bottom-up: On the other hand, there are few bottom-up or proposal-free methods for panoptic segmentation. These works typically get the semantic segmentation prediction before detecting instances by grouping “thing” pixels into clusters. The first bottom-up approach, Deeplab [19], adopts bounding box corners as well as object centers for class-agnostic instance segmentation, coupled with DeepLab semantic segmentation outputs [7, 26]. Recently, SSAP [20] proposes to group pixels based on a pixel-pair affinity pyramid [28] with an efficient graph partition method [29]. Unfortunately, given its simplicity (*i.e.*, a single pass of the system for prediction), bottom-up approaches perform inferiorly to top-down methods at almost *all* public benchmarks. In this work, we aim to push the envelope of bottom-up approaches. We note that there are several instance segmentation works [23, 28, 30–42], which could be potentially extended to bottom-up panoptic segmentation. Additionally, our method bears a similarity to Hough-Voting-based methods [22, 43–45] and recent works by Kendall *et al.* [23], Uhrig *et al.* [46] and Neven *et al.* [41] in the sense that our class-agnostic instance segmentation is obtained by regressing foreground pixels to their centers. However, our method is even simpler than theirs: We directly predict the instance center locations and group pixels to their closest predicted centers. As a result, our method does not require the clustering method OPTICS [47] used in [23], or the advanced clustering loss function proposed in [41]. Finally, our model employs the parallel multi-head prediction framework similar to [31, 48, 49].

Keypoint representation: Recently, keypoint representations have been used for instance segmentation and object detection. Newell *et al.* [36] group pixels by embedding vectors. PersonLab [50] generates person segmentation masks and groups them into instances by learning offset to their detected key-

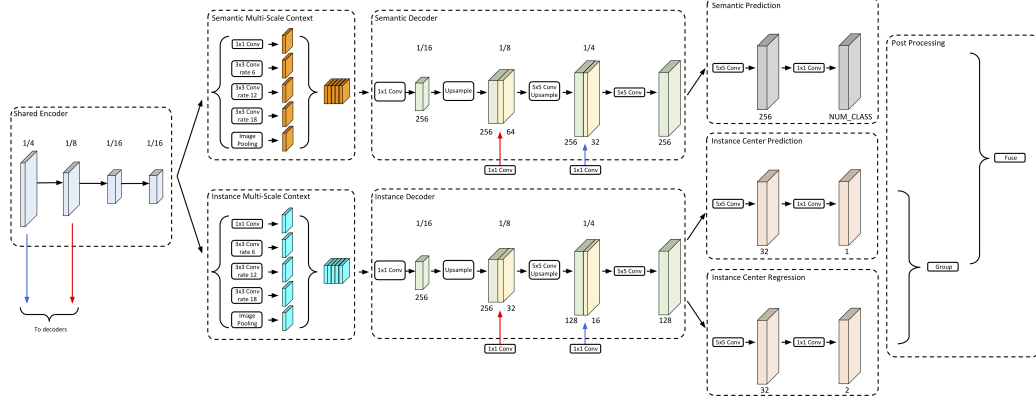


Figure 2.2: Our Panoptic-DeepLab adopts dual-context and dual-decoder modules for semantic segmentation and instance segmentation predictions. We apply atrous convolution in the last block of a network backbone to extract denser feature map. The Atrous Spatial Pyramid Pooling (ASPP) is employed in the context module as well as a lightweight decoder module consisting of a single convolution during each upsampling stage. The instance segmentation prediction is obtained by predicting the object centers and regressing every foreground pixel (*i.e.*, pixels with predicted “thing” class) to their corresponding center. The predicted semantic segmentation and class-agnostic instance segmentation are then fused to generate the final panoptic segmentation result by the “majority vote” proposed by DeeperLab.

points. CornerNet [51] detects objects by predicting paired corners and group corners based on [36]. ExtremeNet [52] groups “extreme points” [53] according to the relation to a center point. Zhou *et al.* [54] and Duan *et al.* [55] exploit instance centers for object detection. Following the same direction, we represent each instance by its center and take a step further by showing that such a simple representation is able to achieve state-of-the-art panoptic segmentation results on several challenging datasets. Different from keypoint-based detection, our Panoptic-DeepLab only requires class-agnostic object center prediction.

2.3 Panoptic-DeepLab

As shown in Figure 2.2, our proposed Panoptic-DeepLab is deployed in a bottom-up and single-shot manner.

2.3.1 Architecture

Panoptic-DeepLab consists of four components: (1) an encoder backbone shared for both semantic segmentation and instance segmentation, (2) decoupled ASPP modules and (3) decoupled decoder modules specific to each task, and (4) task-specific prediction heads.

Basic architecture: The encoder backbone is adapted from an ImageNet-pretrained neural network paired with atrous convolution for extracting denser feature maps in its last block. Motivated by [41, 56, 57], we employ separate ASPP and decoder modules for semantic segmentation and instance segmentation, respectively, based on the hypothesis that those two branches require different contextual and decoding information, which is empirically verified in the following section. Our lightweight decoder module follows DeepLabV3+ [21] with two modifications: (1) We introduce an additional low-level feature with output stride 8 to the decoder, thus the spatial resolution is gradually recovered by a factor of 2, and (2) in each upsampling stage we apply a *single* 5×5 depthwise-separable convolution [58].

Semantic segmentation head: We employ the weighted bootstrapped cross entropy loss, proposed in [19], for semantic segmentation, predicting both “thing” and “stuff” classes. The loss improves over bootstrapped cross entropy loss [59–61] by weighting each pixel differently.

Class-agnostic instance segmentation head: Motivated by Hough Voting [22, 23], we represent each object instance by its center of mass. For every foreground pixel (*i.e.*, pixel whose class is a “thing”), we further predict the offset to its corresponding mass center. During training, groundtruth instance centers are encoded by a 2-D Gaussian with standard deviation of 8 pixels [62]. In particular, we adopt the mean squared error (MSE) loss to minimize the distance between predicted heatmaps and 2D Gaussian-encoded groundtruth heatmaps. We use L_1 loss for the offset prediction, which is only activated at pixels belonging to object instances. During inference, predicted foreground pixels (obtained by filtering out background “stuff” regions from semantic segmentation prediction) are grouped to their closest predicted mass center, forming our class-agnostic instance segmentation results, as detailed below.

2.3.2 Panoptic Segmentation

During inference, we use an extremely simple grouping operation to obtain instance masks, and a highly efficient majority voting algorithm to merge semantic and instance segmentation into final panoptic segmentation.

Simple instance representation: We simply represent each object by its center of mass, $\{\mathcal{C}_n : (i_n, j_n)\}$. To obtain the center point prediction, we first perform a keypoint-based non-maximum suppression (NMS) on the instance center heatmap prediction, essentially equivalent to applying max pooling on the heatmap prediction and keeping locations whose values do not change before and after max pooling. Finally, a hard threshold is used to filter out predictions with low confidence, and only locations with top- k highest confidence scores are kept. In experiments, we use max-pooling with kernel size 7, threshold 0.1, and $k = 200$.

Simple instance grouping: To obtain the instance id for each pixel, we use a simple instance center regression. For example, consider a predicted “thing” pixel at location (i, j) , we predict an offset vector $\mathcal{O}(i, j)$ to its instance center. $\mathcal{O}(i, j)$ is a vector with two elements, representing the offset in horizontal and vertical directions, respectively. The instance id for the pixel is thus the index of the closest instance center after moving the pixel location (i, j) by the offset $\mathcal{O}(i, j)$. That is,

$$\hat{k}_{i,j} = \underset{k}{\operatorname{argmin}} ||\mathcal{C}_k - ((i, j) + \mathcal{O}(i, j))||^2$$

where $\hat{k}_{i,j}$ is the predicted instance id for pixel at (i, j) .

We use semantic segmentation prediction to filter out “stuff” pixels whose instance id are always set to 0.

Efficient merging: Given the predicted semantic segmentation and class-agnostic instance segmentation results, we adopt a fast and parallelizable method to merge the results, following the “majority vote” principle proposed in DeeperLab [19]. In particular, the semantic label of a predicted instance mask is inferred by the majority vote of the corresponding predicted semantic labels. This operation is essentially accumulating the class label histograms, and thus is efficiently implemented in GPU, which takes only 3 ms when operating on a 1025×2049 input.

2.3.3 Instance Segmentation

Panoptic-DeepLab can also generate instance segmentation predictions as a by-product. To properly evaluate the instance segmentation results, one needs to associate a confidence score with each predicted instance mask. Previous bottom-up instance segmentation methods use some heuristics to obtain the confidence scores. For example, DWT [33] and SSAP [20] use an average of semantic segmentation scores for some easy classes and use random scores for other harder classes. Additionally, they remove masks whose areas are below a certain threshold for each class. On the other hand, our Panoptic-DeepLab does not adopt any heuristic or post processing for instance segmentation. Motivated by YOLO [63], we compute the class-specific confidence score for each instance mask as

$$Score(Objectness) \times Score(Class)$$

where $Score(Objectness)$ is unnormalized objectness score obtained from the class-agnostic center point heatmap, and $Score(Class)$ is obtained from the average of semantic segmentation predictions within the predicted mask region.

2.4 Experiments

Cityscapes [12]: The dataset consists of 2975, 500, and 1525 traffic-related images for training, validation, and testing, respectively. It contains 8 “thing” and 11 “stuff” classes.

Mapillary Vistas [13]: A large-scale traffic-related dataset, containing 18K, 2K, and 5K images for training, validation and testing, respectively. It contains 37 “thing” classes and 28 “stuff” classes in a variety of image resolutions, ranging from 1024×768 to more than 4000×6000

COCO [11]: There are 118K, 5K, and 20K images for training, validation, and testing, respectively. The dataset consists of 80 “thing” and 53 “stuff” classes.

Experimental setup: We report mean IoU, average precision (AP), and panoptic quality (PQ) to evaluate the semantic, instance, and panoptic segmentation results.

Table 2.1: Ablation studies on Cityscapes *val* set. **Adam**: Adam optimizer. **MSE**: MSE loss for instance center. **De. x2**: Dual decoder. **ASPP x2**: Dual ASPP. **L-Crop**: Large crop size. **C_{sem}** = 256: 256 (instead of 128) channels in semantic segmentation branch. **C_{Ins}** = 256: 256 (instead of 128) channels in instance segmentation branch. **Sem. Only**: Only semantic segmentation. M-Adds are measured w.r.t. a 1025×2049 input.

Adam	MSE	De. x2	ASPP x2	L-Crop	C _{Sem} = 256	C _{Ins} = 256	Sem. Only	PQ (%)	AP (%)	mIoU (%)	Params (M)	M-Adds (B)
✓								60.3	32.7	78.2	41.85	496.84
✓								61.0	34.3	79.4	41.85	496.84
✓	✓							61.8	33.8	78.6	41.85	496.84
✓	✓	✓						60.8	32.7	79.0	41.93	501.88
✓	✓	✓	✓					62.5	33.9	78.7	43.37	517.17
✓	✓	✓	✓	✓				62.7	34.5	79.6	43.37	517.17
✓	✓	✓	✓	✓	✓			63.0	35.3	80.5	46.72	547.49
✓	✓	✓	✓	✓	✓	✓		62.1	35.1	80.3	46.88	573.86
✓				✓	✓	✓	✓	-	-	80.3	43.60	518.84

Table 2.2: Cityscapes *val* set. **Flip**: Adding left-right flipped inputs. **MS**: Multiscale inputs. **MV**: Mapillary Vistas.

Method	Extra Data	Flip	MS	PQ (%)	AP (%)	mIoU (%)
w/o Extra Data						
TASCNet [16]				55.9	-	-
Panoptic FPN [15]				58.1	33.0	75.7
AUNet [17]				59.0	34.4	75.6
UPNet [14]				59.3	33.3	75.2
UPNet [14]		✓	✓	60.1	33.3	76.8
Seamless [18]				60.3	33.6	77.5
AdaptIS [27]		✓		62.0	36.3	79.2
DeeperLab [19]				56.5	-	-
SSAP [20]		✓	✓	61.1	37.3	-
Panoptic-DeepLab				63.0	35.3	80.5
Panoptic-DeepLab		✓		63.4	36.1	80.9
Panoptic-DeepLab		✓	✓	64.1	38.5	81.5
w/ Extra Data						
TASCNet [16]	COCO			59.3	37.6	78.1
TASCNet [16]	COCO	✓	✓	60.4	39.1	78.7
UPNet [14]	COCO			60.5	37.8	77.8
UPNet [14]	COCO	✓	✓	61.8	39.0	79.2
Seamless [18]	MV			65.0	-	80.7
Panoptic-DeepLab	MV			65.3	38.8	82.5
Panoptic-DeepLab	MV	✓		65.6	39.4	82.6
Panoptic-DeepLab	MV	✓	✓	67.0	42.5	83.1

All our models are trained using TensorFlow on 32 TPUs. We adopt a training protocol similar to that in [21]. In particular, we use the “poly” learning rate policy [67] with an initial learning rate of 0.001, fine-tune the batch normalization [68] parameters, perform random scale data augmentation during training, and optimize with Adam [69] *without weight decay*. On Cityscapes, our best setting is obtained by training with whole image (*i.e.*, crop size equal to 1025×2049) with batch size 32. On Mapillary Vistas, we resize the images to 2177 pixels at the longest side to handle the large input variations, and randomly crop 1025×1025 patches during training with batch size 64. On COCO, we resize the images to 1025 pixels at the longest side and train our models with crop size 1025×1025 with batch size 64. We set training iterations to 60K, 150K, and 200K for Cityscapes, Mapillary Vistas, and COCO, respectively. During evaluation, due to the sensitivity

Table 2.3: Cityscapes *test* set. **C**: Cityscapes coarse annotation. **V**: Cityscapes video. **MV**: Mapillary Vistas.

Method	Extra Data	PQ (%)	AP (%)	mIoU (%)
Semantic Segmentation				
GFF-Net [64]		-	-	82.3
Zhu <i>et al.</i> [65]	C, V, MV	-	-	83.5
Hyundai Mobis AD Lab	C, MV	-	-	83.8
Instance Segmentation				
AdaptIS [27]		-	32.5	-
UPNet [14]	COCO	-	33.0	-
PANet [66]	COCO	-	36.4	-
Sogou-MM	COCO	-	37.2	-
iFLYTEK-CV	COCO	-	38.0	-
NJUST	COCO	-	38.9	-
AInnoSegmentation	COCO	-	39.5	-
Panoptic Segmentation				
SSAP [20]		58.9	32.7	-
TASCNet [16]	COCO	60.7	-	-
Seamless [18]	MV	62.6	-	-
Panoptic-DeepLab		62.3	34.6	79.4
Panoptic-DeepLab	MV	65.5	39.0	84.2

of PQ [14, 16, 18], we re-assign to “VOID” label all “stuff” segments whose areas are smaller than a threshold. The thresholds on Cityscapes, Mapillary Vistas, and COCO are 2048, 4096, and 4096, respectively. Additionally, we adopt multi-scale inference (scales equal to $\{0.5, 0.75, 1, 1.25, 1.5, 1.75, 2\}$ for Cityscapes and Mapillary Vistas and $\{0.5, 0.75, 1, 1.25, 1.5\}$ for COCO) and left-right flipped inputs, to further improve the performance. For all the reported results, unless specified, Xception-71 [21, 70, 71] is employed as the backbone.

Panoptic-DeepLab is trained with three loss functions: weighted bootstrapped cross entropy loss for semantic segmentation head ($\mathcal{L}_{sem}$) [19]; MSE loss for center heatmap head ($\mathcal{L}_{heatmap}$) [62]; and L1 loss for center offset head ($\mathcal{L}_{offset}$) [50]. The final loss $\mathcal{L}$ is computed as follows.

$$\mathcal{L} = \lambda_{sem}\mathcal{L}_{sem} + \lambda_{heatmap}\mathcal{L}_{heatmap} + \lambda_{offset}\mathcal{L}_{offset}$$

Specifically, we set $\lambda_{sem} = 3$ for pixels belonging to instances with an area smaller than 64×64 and $\lambda_{sem} = 1$ everywhere else, following DeeperLab [19]. To make sure the losses are in the similar magnitude, we set $\lambda_{heatmap} = 200$

and $\lambda_{offset} = 0.01$.

2.4.1 Ablation Studies

We conduct ablation studies on Cityscapes validation set, as shown in Table 2.1. Replacing SGD momentum optimizer with Adam optimizer yields 0.7% PQ improvement. Instead of using the sigmoid cross entropy loss for training the heatmap (*i.e.*, instance center prediction), it brings 0.8% PQ improvement by applying the mean squared error (MSE) loss to minimize the distance between the predicted heatmap and the 2D Gaussian-encoded groundtruth heatmap. It is more effective to adopt both dual-decoder and dual-ASPP, which gives us 0.7% PQ improvement while maintaining similar AP and mIoU. Employing a large crop size 1025×2049 (instead of 513×1025) during training further improves the AP and mIoU by 0.6% and 0.9% respectively. Finally, increasing the feature channels from 128 to 256 in the semantic segmentation branch achieves our best result of 63.0% PQ, 35.3% AP, and 80.5% mIoU.

Multi-task learning: For reference, we train a Semantic-DeepLab under the same setting as the best Panoptic-DeepLab (last row of Table 2.1), showing that multi-task learning does not bring extra gain to mIoU. Note that Panoptic-DeepLab adds marginal parameters and small computation overhead over Semantic-DeepLab.

2.4.2 Cityscapes

Val set: In Table 2.2, we report our Cityscapes validation set results. When using only Cityscapes *fine* annotations, our best Panoptic-DeepLab, with multi-scale inputs and left-right flips, outperforms the best bottom-up approach, SSAP, by 3.0% PQ and 1.2% AP, and is better than the best proposal-based approach, AdaptIS, by 2.1% PQ, 2.2% AP, and 2.3% mIoU. When using extra data, our best Panoptic-DeepLab outperforms UPSNet by 5.2% PQ, 3.5% AP, and 3.9% mIoU, and Seamless by 2.0% PQ and 2.4% mIoU. Note that we do not exploit any other data, such as COCO, Cityscapes *coarse* annotations, depth, or video.

Test set: On the test set, we additionally employ the trick proposed in [21] that applies atrous convolution in the last two blocks within the backbone, with rate 2 and 4 respectively, during inference. This trick brings an extra 0.4% AP and 0.2% mIoU on *val* set but no improvement over PQ. We do not use this trick for the Mapillary Vistas Challenge. As shown in Table 2.3, our *single* unified Panoptic-DeepLab achieves state-of-the-art results, ranking first at *all* three Cityscapes tasks, when comparing with published works. Our model ranks second in the instance segmentation track when also taking into account unpublished entries.

2.4.3 Mapillary Vistas

Val set: In Table 2.4, we report Mapillary Vistas *val* set results. Our best *single* Panoptic-DeepLab model, with multi-scale inputs and left-right flips, outperforms the bottom-up approach, DeeperLab, by 8.3% PQ, and the top-down approach, Seamless, by 2.6% PQ. In Table 2.5, we report our results with three families of network backbones. We observe that naïve HRNet-W48 slightly under-performs Xception-71. Due to the diverse image resolutions in Mapillary Vistas, we found it important to enrich the context information as well as to keep high-resolution features. Therefore, we propose a simple modification for HRNet [72] and Auto-DeepLab [73]. For modified HRNet, called HRNet+, we keep its ImageNet-pretrained head and further attach dual-ASPP and dual-decoder modules. For modified Auto-DeepLab, called Auto-DeepLab+, we remove the stride in the original 1/32 branch (which improves PQ by 1%). To summarize, using Xception-71 strikes the best accuracy and speed trade-off, while HRNet-W48+ achieves the best PQ of 40.6%. Finally, our ensemble of six models attains a 42.2% PQ, 18.2% AP, and 58.7% mIoU.

Test set: Table 2.6 summarizes our Mapillary Vistas test set results along with other top-performing methods. Our entry [74] with an ensemble of six models attain a performance of 42.7% PQ, outperforming the winner of Mapillary Vistas Panoptic Segmentation Challenge in 2018 by 1.5% PQ.

Table 2.4: Mapillary Vistas *val* set. **Flip**: Adding left-right flipped inputs. **MS**: Multiscale inputs.

Method	Flip	MS	PQ (%)	PQ Th (%)	PQ St (%)	AP (%)	mIoU (%)
TASCNet [16]			32.6	31.1	34.4	18.5	-
TASCNet [16]	✓	✓	34.3	34.8	33.6	20.4	-
AdaptIS [27]	✓		35.9	31.5	41.9	-	-
Seamless [18]			37.7	33.8	42.9	16.4	50.4
DeeperLab [19]			32.0	-	-	-	55.3
Panoptic-DeepLab			37.7	30.4	47.4	14.9	55.4
Panoptic-DeepLab	✓		38.0	30.6	47.9	15.2	55.8
Panoptic-DeepLab	✓	✓	40.3	33.5	49.3	17.2	56.8

Table 2.5: Mapillary Vistas *val* set with different backbones.

HRNet-W48+: Modified HRNet-W48 with ImageNet-pretraining head kept. **HRNet-W48+ (Atrous)**: Additionally apply atrous convolution with rate 2 in the output stride 32 branch of HRNet. **HRNet-Wider+**: A wider version of HRNet using separable convolution with large channels. The ImageNet-pretraining head is also kept. **HRNet-Wider+ (Atrous)**: Additionally apply atrous convolution with rate 2 in the output stride 32 branch. **Auto-DeepLab-L+**: Auto-DeepLab with $F = 48$ and remove the stride in the original output stride 32 path. **Auto-DeepLab-XL+**: Auto-DeepLab with $F = 64$ and remove the stride in the original output stride 32 path. **Auto-DeepLab-XL++**: Additionally exploit low-level features from output stride 8 endpoint in the decoder module. We employ dual-ASPP and dual-decoder modules for all model variants except **HRNet-W48** which follows the original design in [72]. Results are obtained with multi-scale and left-right flipped inputs. M-Adds are measured w.r.t. a 2177×2177 input.

Backbone	Params (M)	M-Adds (B)	PQ (%)	AP (%)	mIoU (%)
Xception-65	44.31	1054.05	39.2	16.4	56.9
Xception-71	46.73	1264.32	40.3	17.2	56.8
HRNet-W48 [72]	71.66	2304.87	39.3	17.2	55.4
HRNet-W48+	88.87	2208.04	40.6	17.8	57.6
HRNet-W48+ (Atrous)	88.87	2972.02	40.5	17.7	57.4
HRNet-Wider+	60.05	1315.70	40.0	17.0	57.0
HRNet-Wider+ (Atrous)	60.05	1711.69	39.7	16.8	56.5
Auto-DeepLab-L+	41.54	1493.78	39.3	15.8	56.9
Auto-DeepLab-XL+	71.98	2378.17	40.3	16.3	57.1
Auto-DeepLab-XL++	72.16	2386.81	40.3	16.9	57.6
Ensemble (top-6 models)	-	-	42.2	18.2	58.7

Table 2.6: Performance on Mapillary Vistas *test* set.

Method	PQ	SQ	RQ	PQ Th	SQ Th	RQ Th	PQ St	SQ St	RQ St
DeeperLab [19]	31.6	75.5	40.1	25.0	73.4	33.1	40.3	78.3	49.3
AdaptIS [27]	36.8	76.0	46.3	33.3	75.2	42.6	41.4	77.1	51.3
TRI-ML (2018: 2 nd)	38.7	78.1	48.4	39.0	79.7	48.9	38.2	75.9	47.9
Team R4D (2018: 1 st)	41.2	79.1	50.8	37.9	79.7	47.1	45.6	78.4	55.8
Panoptic-DeepLab	42.7	78.1	52.5	35.9	75.3	46.0	51.6	81.9	61.2

Table 2.7: COCO *val* set. **Flip**: Adding left-right flipped inputs. **MS**: Multiscale inputs.

Method	Backbone	Flip	MS	PQ (%)	PQ Th (%)	PQ St (%)
AUNet [17]	ResNet-50 [77]			39.6	49.1	25.2
Panoptic-FPN [15]	ResNet-101			40.3	47.5	29.5
AdaptIS [27]	ResNeXt-101 [75]	✓		42.3	49.2	31.8
UPSNet [14]	ResNet-50			42.5	48.5	33.4
Detectron2 [78]	ResNet-101			43.0	-	-
UPSNet [14]	ResNet-50	✓	✓	43.2	49.1	34.1
DeeperLab [19]	Xception-71			33.8	-	-
SSAP [20]	ResNet-101	✓	✓	36.5	-	-
Panoptic-DeepLab	Xception-71			39.7	43.9	33.2
Panoptic-DeepLab	Xception-71	✓		40.2	44.4	33.8
Panoptic-DeepLab	Xception-71	✓	✓	41.2	44.9	35.7

2.4.4 COCO

Val set: In Table 2.7, we report COCO *val* set result. With a single scale inference, our Panoptic-DeepLab outperforms the previous best bottom-up SSAP by 3.2% PQ and DeeperLab [19] by 5.9% PQ. With multi-scale inference and horizontal flip, Panoptic-DeepLab achieves 41.2% PQ, setting a new state-of-the-art performance for bottom-up methods, and performing comparably with top-down methods.

Test-dev set: In Table 2.8, we report COCO *test-dev* set result. Our Panoptic-DeepLab is 4.5% PQ better than the previous best bottom-up SSAP on COCO and our 41.4% PQ is comparable to most top-down methods without using heavier backbone [75] or deformable convolution [76].

Table 2.8: COCO *test-dev* set. **Flip**: Adding left-right flipped inputs. **MS**: Multiscale inputs.

Method	Backbone	Flip	MS	PQ (%)	PQ Th (%)	PQ St (%)
TASCNet [16]	ResNet-50			40.7	47.0	31.0
Panoptic-FPN [15]	ResNet-101			40.9	48.3	29.7
AdaptIS [27]	ResNeXt-101	✓		42.8	53.2	36.7
AUNet [17]	ResNeXt-152			46.5	55.8	32.5
UPNet [14]	DCN-101 [76]	✓	✓	46.6	53.2	36.7
DeeperLab [19]	Xception-71			34.3	37.5	29.6
SSAP [20]	ResNet-101	✓	✓	36.9	40.1	32.0
Panoptic-DeepLab	Xception-71	✓	✓	41.4	45.1	35.9

2.4.5 Runtime

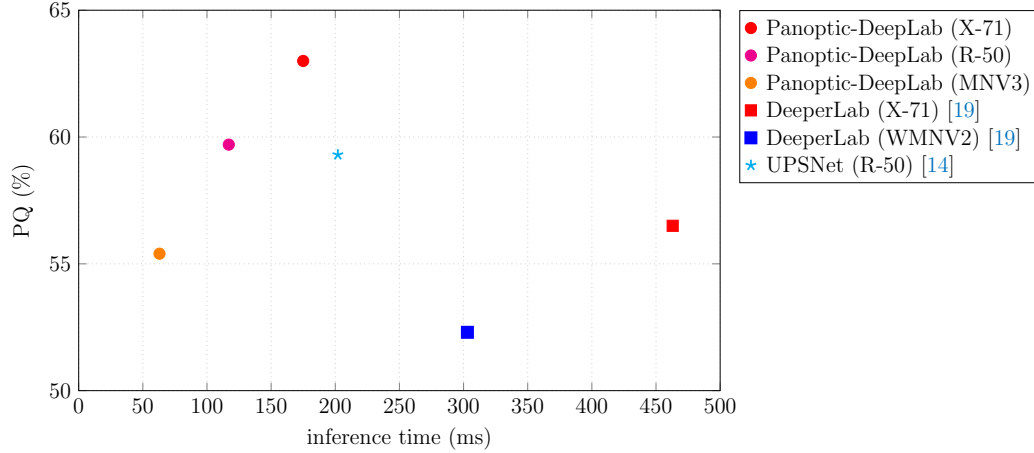
In Table 2.9, we report the end-to-end runtime (*i.e.*, inference time from an input image to final panoptic segmentation, including *all* operations such as merging semantic and instance segmentation) of Panoptic-DeepLab with three different network backbones (MobileNetV3 [79], ResNet-50 [77], and Xception-71 [70, 71]) on all three datasets. The inference speed is measured on a Tesla V100-SXM2 GPU *with batch size of one*. We further plot the speed-accuracy trade-off curve in Figure 2.3. Our Panoptic-DeepLab achieves the best trade-off across all three datasets.

2.4.6 Discussion

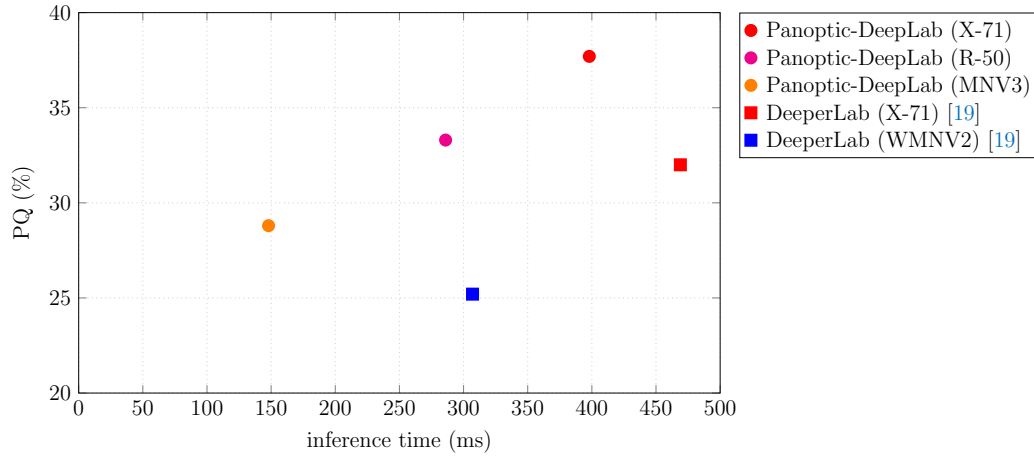
Herein, we list a few interesting aspects in the hope of inspiring future works on bottom-up panoptic segmentation.

Scale variation: Figure 2.4 shows visualization of Panoptic-DeepLab. In particular, the crossroad (in last 2 rows), with a large scale variation, is segmented into multiple small instances. On the other hand, top-down methods handle scale variation to some extent by the ROI Pooling [82] or ROI Align [5] operations which normalize regional features to a *canonical* scale [83, 84]. Additionally, incorporating scale-aware information to feature pyramid [25] or image pyramid [85] may improve the performance of bottom-up methods.

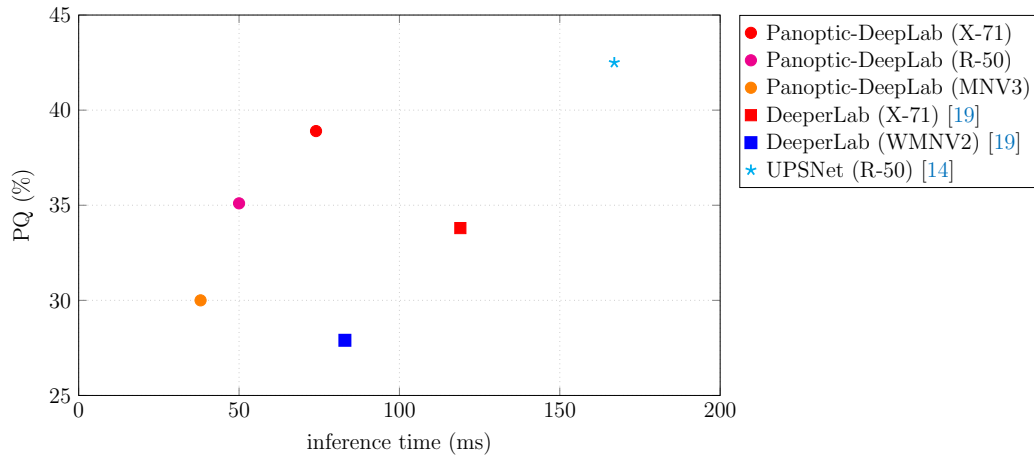
PQ^{Thing} vs. PQ^{Stuff}: As shown in Table 2.6 and Table 2.8, Panoptic-DeepLab has higher PQ^{Stuff} but lower PQ^{Thing} when compared with other



(a) PQ *vs.* inference time (ms) on Cityscapes val set.



(b) PQ *vs.* inference time (ms) on Mapillary Vistas val set



(c) PQ *vs.* inference time (ms) on COCO val set

Figure 2.3: PQ *vs.* Seconds. Our Panoptic-DeepLab model variants attain a better speed/accuracy trade-off across challenging datasets. The inference time is measured *end-to-end* from input image to panoptic segmentation output. **X-71**: Xception-71. **R-50**: ResNet-50. **MNV3**: MobileNetV3. Data points from Table 2.9.

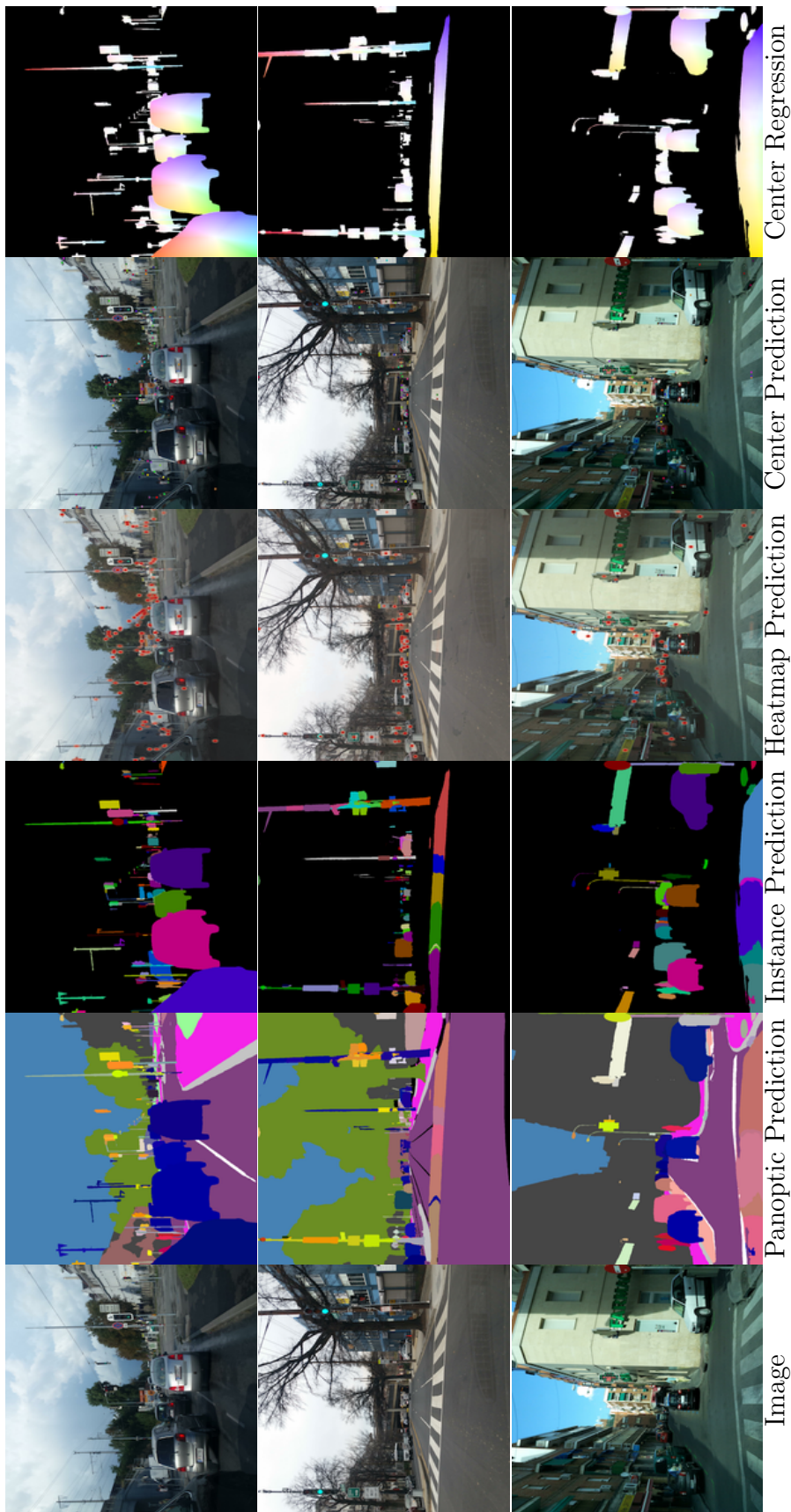


Figure 2.4: Visualization of Panoptic-DeepLab with Xception-71 on Mapillary Vistas *val* set. Only single scale inference is used and the model achieves 37.7% PQ. We encode 2D offset vectors into RGB values, same as [81]. The cross road in last 2 rows is segmented into multiple instances due to large scale variation. More visualizations are included in Section 2.6.6.

Table 2.9: End-to-end runtime, including merging semantic and instance segmentation. All results are obtained by (1) a single-scale input without flipping, and (2) built-in TensorFlow library without extra inference optimization. **MNV3**: MobileNet-V3. **PQ [val]**: PQ (%) on val set. **PQ [test]**: PQ (%) on test(-dev) set. Note the channels in last block of MNV3 are reduced by a factor of 2 [79].

Method	Backbone	Input Size	PQ [val]	PQ [test]	Speed (ms)	M-Adds (B)
Cityscapes						
DeeperLab [19]	W-MNV2 [80]	1025×2049	52.3	-	303	-
DeeperLab [19]	Xception-71	1025×2049	56.5	-	463	-
UPSNet [14]	ResNet-50	1024×2048	59.3	-	202	-
Panoptic-DeepLab	MNV3	1025×2049	55.4	54.1	63	54.17
Panoptic-DeepLab	ResNet-50	1025×2049	59.7	58.0	117	381.39
Panoptic-DeepLab	Xception-71	1025×2049	63.0	60.7	175	547.49
Mapillary Vistas						
DeeperLab [19]	W-MNV2	1441×1441	25.2	25.3	307	-
DeeperLab [19]	Xception-71	1441×1441	32.0	31.6	469	-
Panoptic-DeepLab	MNV3	2177×2177	28.8	-	148	138.12
Panoptic-DeepLab	ResNet-50	2177×2177	33.3	-	286	910.47
Panoptic-DeepLab	Xception-71	2177×2177	37.7	-	398	1264.32
COCO						
DeeperLab [19]	W-MNV2	641×641	27.9	28.1	83	-
DeeperLab [19]	Xception-71	641×641	33.8	34.3	119	-
UPSNet [14]	ResNet-50	800×1333	42.5	-	167	-
Panoptic-DeepLab	MNV3	641×641	30.0	29.8	38	12.24
Panoptic-DeepLab	ResNet-50	641×641	35.1	35.2	50	77.79
Panoptic-DeepLab	Xception-71	641×641	38.9	38.8	74	109.21
Panoptic-DeepLab	Xception-71	1025×1025	39.7	39.6	132	279.25

top-down approaches which better handle instances of large scale variation as discussed above. Combining the best from both bottom-up and top-down approaches is thus interesting to explore but beyond the scope of current work.

Panoptic vs. instance annotations: Most bottom-up panoptic segmentation methods only exploit the panoptic annotations. We notice there are two types of annotations in the COCO dataset, panoptic annotations and instance annotations. The former do not allow overlapping masks (thus creating occlusions among masks), while the latter allows overlaps, which might make the training target easier to optimize, similar to amodal segmentation [86, 87].

End-to-end training: Current bottom-up panoptic segmentation methods still require some post-processing steps to obtain the final panoptic segmentation, which may make it hard to end-to-end train the whole system.

2.5 Conclusion

We have presented Panoptic-DeepLab, a simple, strong, and fast baseline for bottom-up panoptic segmentation. Panoptic-DeepLab is simple in design, requiring only three loss functions during training and adds marginal parameters to a modern semantic segmentation model. Panoptic-DeepLab is the first bottom-up and single-shot panoptic segmentation model that attains state-of-the-art performance on several public benchmarks, and delivers near real-time end-to-end inference speed. We hope our simple and effective model could establish a solid baseline and further benefit the research community.

2.6 Supplementary Materials

2.6.1 HRNet Variant

We introduce our modifications to the HRNet [72] that are used in our ensemble model [74] for Mapillary Vistas. All hyper-parameters for training HRNet variants are the same as Xception, except that the learning rate is set to $7.5e - 4$.

2.6.1.1 HRNet

The original segmentation head for HRNet is shown in Figure 2.5. Features from all four resolutions are first upsampled to the $1/4$ resolution and concatenated, followed by another 1×1 convolution to fuse features.

To pre-train the HRNet on ImageNet [88], Wang *et al.* [72] designed a specific image classification head which gradually downsamples the feature maps, as shown in Figure 2.6 (a). Specifically, a bottleneck residual module [77] is applied to every output resolution to increase the channels. The feature map from the finest spatial resolution (*i.e.*, $1/4$ resolution) is then

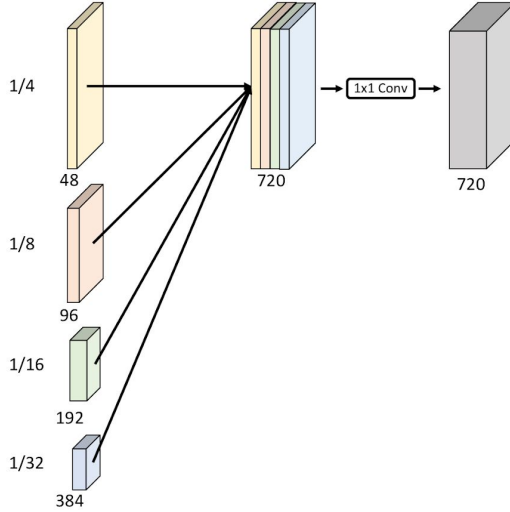


Figure 2.5: Semantic segmentation head proposed in HRNet [72].

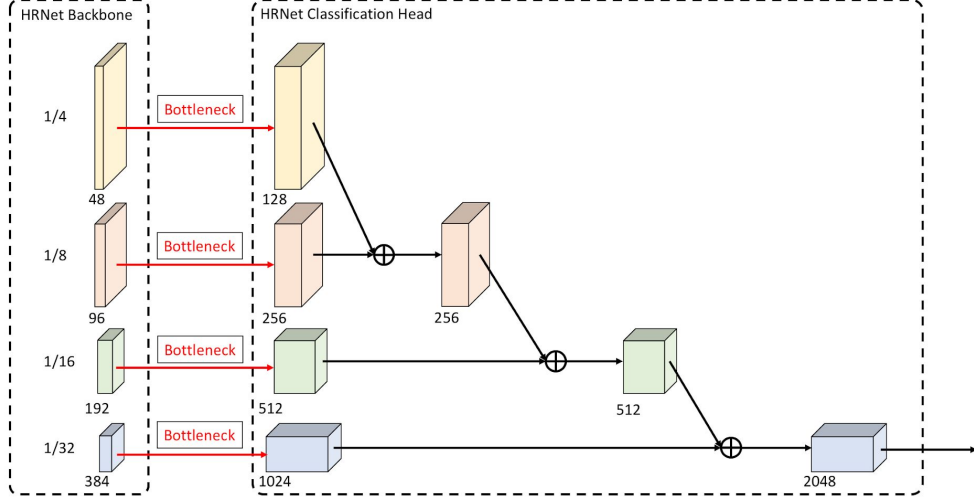
downsampled by sequentially using a 3×3 convolution with stride 2. At the final $1/32$ resolution feature map, a global average pooling and a fully connected layer are attached for ImageNet classification.

Table 2.10: Comparison between the decoder design of DeepLabV3+ [21] and Panoptic-DeepLab on Cityscapes validation set.

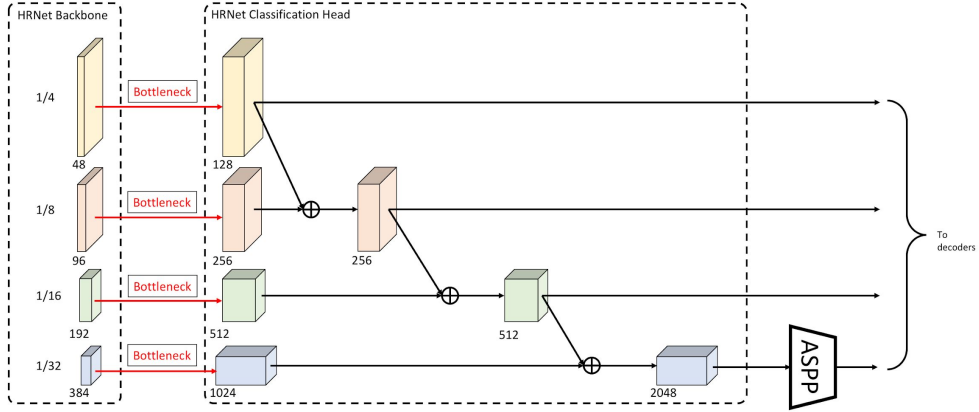
Decoder	PQ (%)	AP (%)	mIoU (%)	Speed (ms)	Params (M)	M-Adds (B)
DeepLabV3+ [21]	62.5	34.5	80.2	176	46.61	553.41
Panoptic-DeepLab	63.0	35.3	80.5	175	46.72	547.49

2.6.1.2 HRNet+

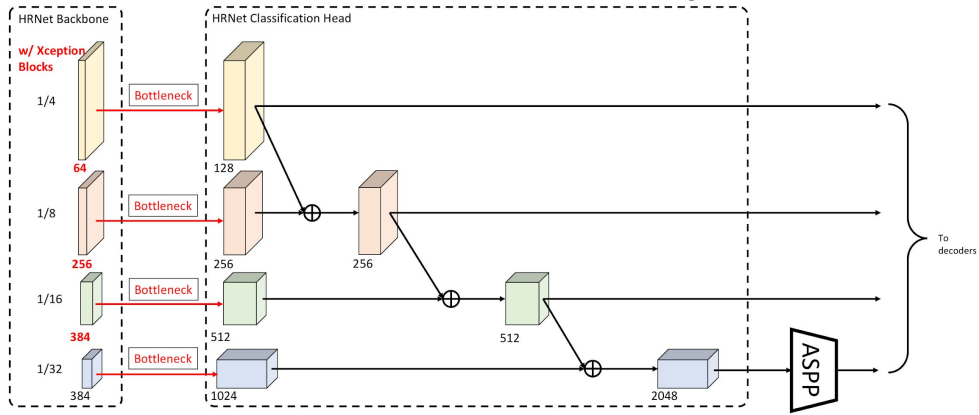
After pre-training on ImageNet, Wang *et al.* [72] removed the image classification head. However, we observe that the classification head takes around 20% of the total parameters, which is a waste of information if discarded. Therefore, we propose to keep this classification head in our modified HRNet+ (Figure 2.6 (b)). Starting from the image classification HRNet, we replace the final global average pooling and linear classifier with an ASPP module, and build a similar decoder as shown in Figure 2.2 with some differences that the output stride of encoder is now 32 instead of 16 and we introduce one more encoder feature map of stride 16 to the decoder by first projecting its channels to 96.



(a) Image classification head proposed in HRNet [72], which is discarded after pre-training on ImageNet.



(b) Our proposed HRNet+, which keeps the image classification head and attaches the ASPP module as well as the decoder module for segmentation tasks.



(c) Our proposed HRNet-Wider+, which reduces the model parameters and computations by adopting the Xception module.

Figure 2.6: Demonstration of our proposed variants of HRNet [72].

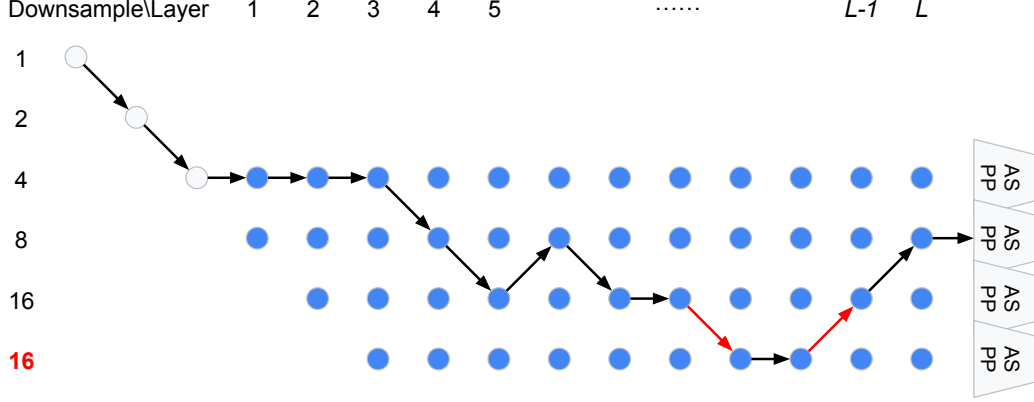


Figure 2.7: Our proposed Auto-DeepLab+, which keeps the high spatial resolution of feature maps by removing the last stride, *i.e.*, no spatial resolution changes marked in the red arrows.

2.6.1.3 HRNet-Wider+

We additionally propose HRNet-Wider+ (Figure 2.6 (c)) that replaces the basic residual module [77] with the Xception module [70], significantly reducing the model parameters and computation FLOPs at the cost of marginal degradation in performance. Additionally, we employ the number of channels $\{64, 256, 384, 384\}$ for each resolution (instead of $\{48, 96, 192, 384\}$).

2.6.1.4 Atrous HRNet

Another modification of HRNet that we have explored is referred to as HRNet+ (Atrous), where we remove all the downsampling operations that generate $1/32$ resolution feature maps and apply atrous convolution with rate equal to 2 in that branch. This modification increases the computation FLOPs but does not improve the performance compared to its HRNet+ counterpart.

2.6.2 Auto-DeepLab Variant

We make a simple modification to the Auto-DeepLab [73] in Figure 2.7 by removing the stride in the convolution that generates the $1/32$ feature map in order to keep high spatial resolution within the network backbone. We find this modification improves 1% PQ on Mapillary Vistas validation set.

2.6.3 Comparison with DeepLabV3+ Decoder

As mentioned before, the decoder of Panoptic-DeepLab is slightly different from the one in DeepLabv3+ [21]. Herein, we compare their performance on Cityscapes validation set, as shown in Table 2.10. Panoptic-DeepLab outperforms DeepLabv3+ by 0.5% PQ, 0.8% AP, and 0.3% mIOU, showing more improvement in the instance segmentation task. Additionally, Panoptic-DeepLab is slightly faster than DeepLabv3+ at the cost of extra marginal parameters.

2.6.4 Comparison with Different Instance Scores

Table 2.11: Ablation study on using different confidence scores. Note the choice of confidence scores only affects AP.

Instance score	PQ (%)	AP (%)	mIoU (%)
Score(Objectness)	63.0	28.9	80.5
Score(Class)	63.0	35.1	80.5
Score(Objectness) x Score(Class)	63.0	35.3	80.5

In Table 2.11, we experiment with different confidence scores when evaluating instance segmentation results. We found that using Score(Objectness) alone leads to 28.9% AP, Score(Class) alone produces 35.1% AP, while employing Score(Objectness) $\times$ Score(Class) generates the best result (35.3% AP). We would like to highlight that the choice of different confidence score does not affect our final mIoU and PQ results, since our Panoptic-DeepLab does not produce overlapping predictions and therefore does not require a confidence score to rank predictions (or to resolve the conflict among overlapping predictions) like Panoptic-FPN [15]. Confidence score is only used in computing AP to rank instance mask predictions. Since it is only used for the purpose of ranking, the confidence score does not necessarily need to be a probability.

2.6.5 Instance and Panoptic Annotation

Figure 2.8 shows an example to illustrate the difference between instance annotation and panoptic annotation on the COCO dataset. Instance annota-

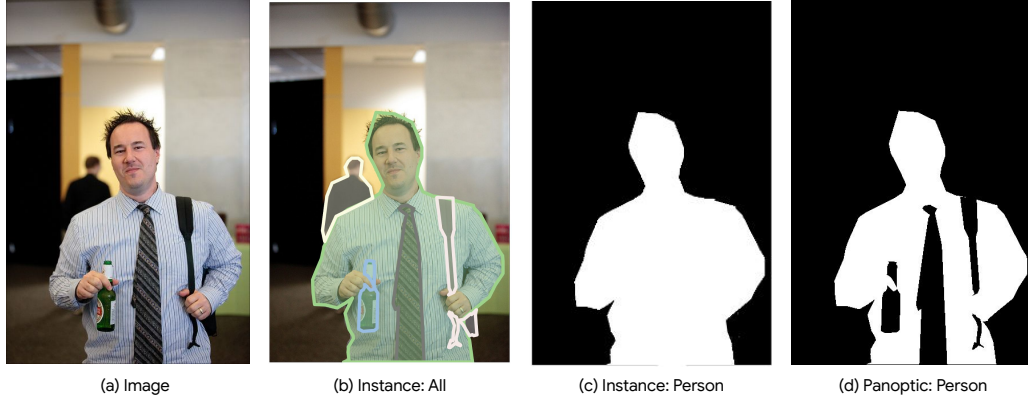


Figure 2.8: Illustration of the difference between instance and panoptic annotation on COCO.

tion, unlike panoptic annotation, allows overlapping groundtruth masks. For example, the “person” mask ignores the existence of the “tie” and “bottle” masks in the instance annotation, while the “person” mask has occlusions caused by other instances in the panoptic annotation.

We notice that all top-down methods based on Mask R-CNN [5] use the *instance annotation* [14, 15, 17] when trained on COCO, while bottom-up methods [19] including our Panoptic-DeepLab use the *panoptic annotation* on all datasets.

2.6.6 More Visualization

We provide more visualization results of our Panoptic-DeepLab in Figure 2.9, Figure 2.10, and Figure 2.11.

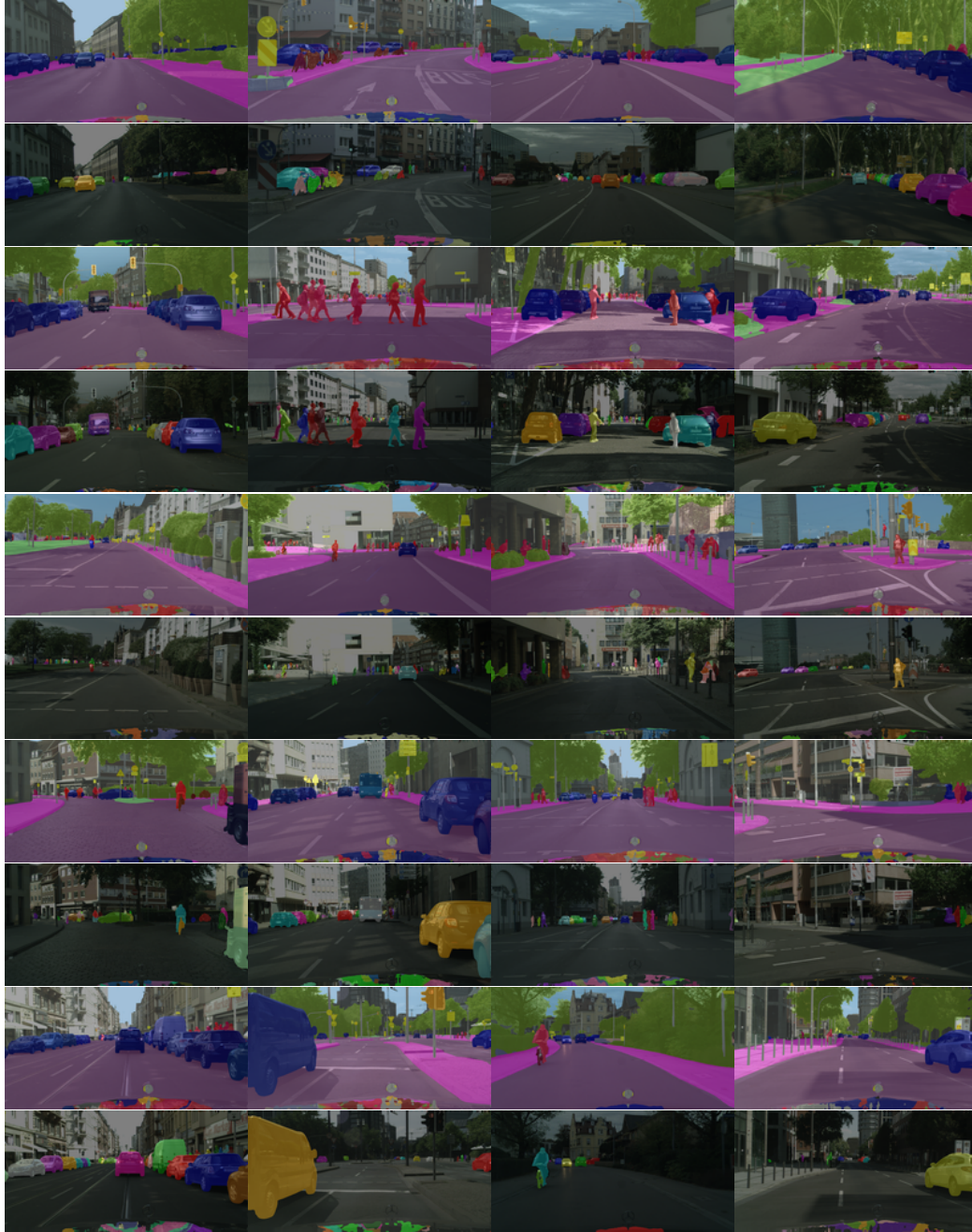


Figure 2.9: Visualization of Panoptic-DeepLab with Xception-71 on Cityscapes *val* set. Only single scale inference is used and the model achieves 63.0% PQ. The first row is panoptic prediction and the second row is instance prediction.



Figure 2.10: Visualization of Panoptic-DeepLab with Xception-71 on Mapillary Vistas *val* set. Only single scale inference is used and the model achieves 37.7% PQ. The first row is panoptic prediction and the second row is instance prediction.



Figure 2.11: Visualization of Panoptic-DeepLab with Xception-71 on COCO *val* set. Only single scale inference is used and the model achieves 39.7% PQ. The first row is panoptic prediction and the second row is instance prediction.

CHAPTER 3

IMAGE SEGMENTATION WITH MASK CLASSIFICATION

3.1 Introduction

The goal of semantic segmentation is to partition an image into regions with different semantic categories. Starting from Fully Convolutional Networks (FCNs) work of Long *et al.* [89], most *deep learning-based* semantic segmentation approaches formulate semantic segmentation as *per-pixel classification* (Figure 3.1 left), applying a classification loss to each output pixel [21, 90]. Per-pixel predictions in this formulation naturally partition an image into regions of different classes.

Mask classification is an alternative paradigm that disentangles the image partitioning and classification aspects of segmentation. Instead of classifying each pixel, mask classification-based methods predict a set of binary masks, each associated with a *single* class prediction (Figure 3.1 right). The more flexible mask classification dominates the field of instance-level segmentation. Both Mask R-CNN [5] and DETR [91] yield a single class prediction per segment for instance and panoptic segmentation. In contrast, per-pixel classification assumes a static number of outputs and cannot return a variable number of predicted regions/segments, which is required for instance-level tasks.

Our key observation: Mask classification is sufficiently general to solve both semantic- and instance-level segmentation tasks. In fact, before FCN [89], the best performing semantic segmentation methods like O2P [92] and SDS [3] used a mask classification formulation. Given this perspective, a natural question emerges: *Can a single mask classification model simplify the landscape of effective approaches to semantic- and instance-level segmentation tasks? And can such a mask classification model outperform existing per-pixel classification methods for semantic segmentation?*

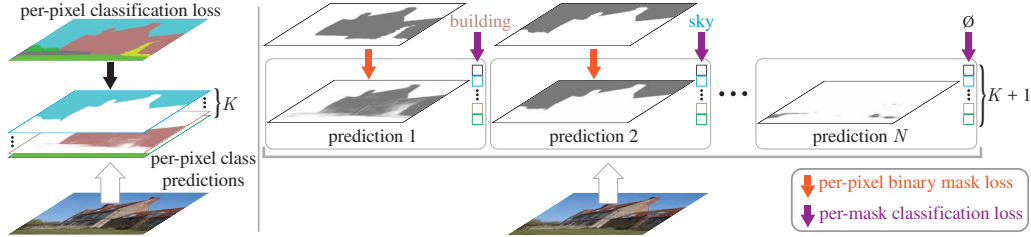


Figure 3.1: **Per-pixel classification vs. mask classification.** (left)

Semantic segmentation with per-pixel classification applies the same classification loss to each location. (right) Mask classification predicts a set of binary masks and assigns a single class to each mask. Each prediction is supervised with a per-pixel binary mask loss and a classification loss. Matching between the set of predictions and ground truth segments can be done either via *bipartite matching* similarly to DETR [91] or by *fixed matching* via direct indexing if the number of predictions and classes match, *i.e.*, if $N = K$.

To address both questions we propose a simple **MaskFormer** approach that seamlessly converts any existing per-pixel classification model into a mask classification. Using the set prediction mechanism proposed in DETR [91], MaskFormer employs a Transformer decoder [93] to compute a set of pairs, each consisting of a class prediction and a mask embedding vector. The mask embedding vector is used to get the binary mask prediction via a dot product with the per-pixel embedding obtained from an underlying fully-convolutional network. The new model solves both semantic- and instance-level segmentation tasks in a unified manner: No changes to the model, losses, and training procedure are required. Specifically, for semantic and panoptic segmentation tasks alike, MaskFormer is supervised with the same per-pixel binary mask loss and a single classification loss per mask. Finally, we design a simple inference strategy to blend MaskFormer outputs into a task-dependent prediction format.

We evaluate MaskFormer on five semantic segmentation datasets with various numbers of categories: Cityscapes [12] (19 classes), Mapillary Vistas [13] (65 classes), ADE20K [94] (150 classes), COCO-Stuff-10K [95] (171 classes), and ADE20K-Full [94] (847 classes). While MaskFormer performs on par with per-pixel classification models for Cityscapes, which has a few diverse classes, the new model demonstrates superior performance for datasets with larger vocabulary. We hypothesize that a single class prediction per mask models fine-grained recognition better than per-pixel class predictions.

MaskFormer achieves the new state-of-the-art on ADE20K (**55.6 mIoU**) with Swin-Transformer [96] backbone, outperforming a per-pixel classification model [96] with the same backbone by 2.1 mIoU, while being more efficient (10% reduction in parameters and 40% reduction in FLOPs).

Finally, we study MaskFormer’s ability to solve instance-level tasks using two panoptic segmentation datasets: COCO [4, 11] and ADE20K [94]. MaskFormer outperforms a more complex DETR model [91] with the same backbone and the same post-processing. Moreover, MaskFormer achieves the new state-of-the-art on COCO (**52.7 PQ**), outperforming prior state-of-the-art [97] by 1.6 PQ. Our experiments highlight MaskFormer’s ability to unify instance- and semantic-level segmentation.

3.2 Related Works

Both per-pixel classification and mask classification have been extensively studied for semantic segmentation. In early work, Konishi and Yuille [98] apply per-pixel Bayesian classifiers based on local image statistics. Then, inspired by early works on non-semantic groupings [99, 100], mask classification-based methods became popular demonstrating the best performance in PASCAL VOC challenges [2]. Methods like O2P [92] and CFM [101] have achieved state-of-the-art results by classifying mask proposals [102–104]. In 2015, FCN [89] extended the idea of per-pixel classification to deep nets, significantly outperforming all prior methods on mIoU (a per-pixel evaluation metric which particularly suits the per-pixel classification formulation of segmentation).

Per-pixel classification became the dominant way for *deep-net-based* semantic segmentation since the seminal work of fully convolutional networks (FCNs) [89]. Modern semantic segmentation models focus on aggregating long-range context in the final feature map: ASPP [7, 26] uses atrous convolutions with different atrous rates; PPM [90] uses pooling operators with different kernel sizes; DANet [105], OCNet [106], and CCNet [107] use different variants of non-local blocks [108]. Recently, SETR [109] and Segformer [110] replace traditional convolutional backbones with Vision Transformers (ViT) [111] that capture long-range context starting from the very first layer. However, these concurrent Transformer-based [93] semantic seg-

mentation approaches still use a per-pixel classification formulation. Note that our MaskFormer module can convert any per-pixel classification model to the mask classification setting, allowing seamless adoption of advances in per-pixel classification.

Mask classification is commonly used for instance-level segmentation tasks [3, 4]. These tasks require a dynamic number of predictions, making application of per-pixel classification challenging as it assumes a static number of outputs. Omnipresent Mask R-CNN [5] uses a global classifier to classify mask proposals for instance segmentation. DETR [91] further incorporates a Transformer [93] design to handle thing and stuff segmentation simultaneously for panoptic segmentation [4]. However, these mask classification methods require predictions of bounding boxes, which may limit their usage in semantic segmentation. The recently proposed Max-DeepLab [97] removes the dependence on box predictions for panoptic segmentation with conditional convolutions [112, 113]. However, in addition to the main mask classification losses it requires multiple auxiliary losses (*i.e.*, instance discrimination loss, mask-ID cross entropy loss, and the standard per-pixel classification loss).

3.3 From Per-Pixel to Mask Classification

In this section, we first describe how semantic segmentation can be formulated as either a per-pixel classification or a mask classification problem. Then, we introduce our instantiation of the mask classification model with the help of a Transformer decoder [93]. Finally, we describe simple inference strategies to transform mask classification outputs into task-dependent prediction formats.

3.3.1 Per-pixel Classification Formulation

For per-pixel classification, a segmentation model aims to predict the probability distribution over all possible K categories for every pixel of an $H \times W$ image: $y = \{p_i | p_i \in \Delta^K\}_{i=1}^{H \cdot W}$. Here Δ^K is the K -dimensional probability simplex. Training a per-pixel classification model is straightforward: Given ground truth category labels $y^{\text{gt}} = \{y_i^{\text{gt}} | y_i^{\text{gt}} \in \{1, \dots, K\}\}_{i=1}^{H \cdot W}$ for every pixel, a per-pixel cross-entropy (negative log-likelihood) loss is usually applied, *i.e.*,

$$\mathcal{L}_{\text{pixel-clas}}(y, y^{\text{gt}}) = \sum_{i=1}^{H \cdot W} -\log p_i(y_i^{\text{gt}}).$$

3.3.2 Mask Classification Formulation

Mask classification splits the segmentation task into (1) partitioning/grouping the image into N regions (N does not need to equal K), represented with binary masks $\{m_i | m_i \in [0, 1]^{H \times W}\}_{i=1}^N$; and (2) associating each region as a whole with some distribution over K categories. To jointly group and classify a segment, *i.e.*, to perform mask classification, we define the desired output z as a set of N probability-mask pairs, *i.e.*, $z = \{(p_i, m_i)\}_{i=1}^N$. In contrast to per-pixel class probability prediction, for mask classification the probability distribution $p_i \in \Delta^{K+1}$ contains an auxiliary “no object” label ($\emptyset$) in addition to the K category labels. The $\emptyset$ label is predicted for masks that do not correspond to any of the K categories. Note that mask classification allows multiple mask predictions with the same associated class, making it applicable to both semantic- and instance-level segmentation tasks.

To train a mask classification model, a matching σ between the set of predictions z and the set of N^{gt} ground truth segments $z^{\text{gt}} = \{(c_i^{\text{gt}}, m_i^{\text{gt}}) | c_i^{\text{gt}} \in \{1, \dots, K\}, m_i^{\text{gt}} \in \{0, 1\}^{H \times W}\}_{i=1}^{N^{\text{gt}}}$ is required.¹ Here c_i^{gt} is the ground truth class of the i^{th} ground truth segment. Since the sizes of prediction set $|z| = N$ and ground truth set $|z^{\text{gt}}| = N^{\text{gt}}$ generally differ, we assume $N \geq N^{\text{gt}}$ and pad the set of ground truth labels with “no object” tokens $\emptyset$ to allow one-to-one matching.

For semantic segmentation, a trivial *fixed matching* is possible if the number of predictions N matches the number of category labels K . In this case, the i^{th} prediction is matched to a ground truth region with class label i and to $\emptyset$ if a region with class label i is not present in the ground truth. In our experiments, we found that a *bipartite matching*-based assignment demonstrates better results than the fixed matching. Unlike DETR [91] that uses bounding boxes to compute the assignment costs between prediction z_i and ground truth z_j^{gt} for the matching problem, we directly use class and mask predictions, *i.e.*, $-p_i(c_j^{\text{gt}}) + \mathcal{L}_{\text{mask}}(m_i, m_j^{\text{gt}})$, where $\mathcal{L}_{\text{mask}}$ is a binary mask loss.

To train model parameters, given a matching, the main mask classification

¹Different mask classification methods utilize various matching rules. For instance, Mask R-CNN [5] uses a heuristic procedure based on anchor boxes and DETR [91] optimizes a bipartite matching between z and z^{gt} .

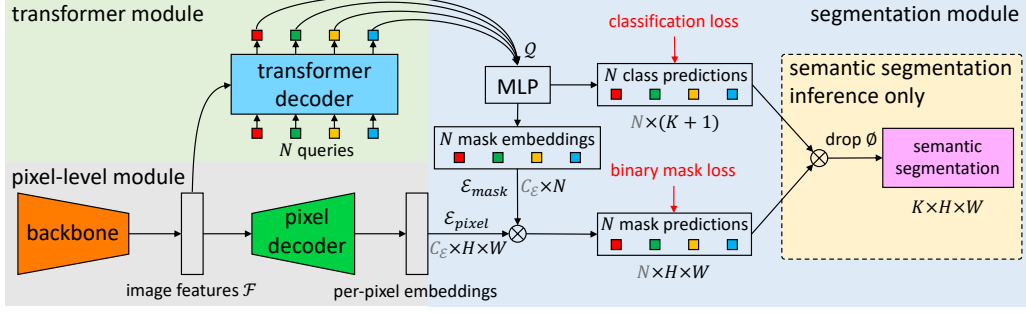


Figure 3.2: **MaskFormer overview.** We use a backbone to extract image features $\mathcal{F}$. A pixel decoder gradually upsamples image features to extract per-pixel embeddings $\mathcal{E}_{\text{pixel}}$. A transformer decoder attends to image features and produces N per-segment embeddings $\mathcal{Q}$. The embeddings independently generate N class predictions with N corresponding mask embeddings $\mathcal{E}_{\text{mask}}$. Then, the model predicts N possibly overlapping binary mask predictions via a dot product between pixel embeddings $\mathcal{E}_{\text{pixel}}$ and mask embeddings $\mathcal{E}_{\text{mask}}$ followed by a sigmoid activation. For semantic segmentation task we can get the final prediction by combining N binary masks with their class predictions using a simple matrix multiplication (see Section 3.3.4). Note that the dimensions for multiplication $\otimes$ are shown in gray.

loss $\mathcal{L}_{\text{mask-cls}}$ is composed of a cross-entropy classification loss and a binary mask loss $\mathcal{L}_{\text{mask}}$ for each predicted segment:

$$\mathcal{L}_{\text{mask-cls}}(z, z^{\text{gt}}) = \sum_{j=1}^N \left[-\log p_{\sigma(j)}(c_j^{\text{gt}}) + \mathbb{1}_{c_j^{\text{gt}} \neq \emptyset} \mathcal{L}_{\text{mask}}(m_{\sigma(j)}, m_j^{\text{gt}}) \right]. \quad (3.1)$$

Note that most existing mask classification models use auxiliary losses (*e.g.*, a bounding box loss [5, 91] or an instance discrimination loss [97]) in addition to $\mathcal{L}_{\text{mask-cls}}$. In the next section we present a simple mask classification model that allows end-to-end training with $\mathcal{L}_{\text{mask-cls}}$ alone.

3.3.3 MaskFormer

We now introduce MaskFormer, the new mask classification model, which computes N probability-mask pairs $z = \{(p_i, m_i)\}_{i=1}^N$. The model contains three modules (see Figure 3.2): (1) a pixel-level module that extracts per-pixel embeddings used to generate binary mask predictions; (2) a transformer module, where a stack of Transformer decoder layers [93] computes N per-segment embeddings; and (3) a segmentation module, which generates pre-

dictionaries $\{(p_i, m_i)\}_{i=1}^N$ from these embeddings. During inference, discussed in Sec. 3.3.4, p_i and m_i are assembled into the final prediction.

Pixel-level module takes an image of size $H \times W$ as input. A backbone generates a (typically) low-resolution image feature map $\mathcal{F} \in \mathbb{R}^{C_{\mathcal{F}} \times \frac{H}{S} \times \frac{W}{S}}$, where $C_{\mathcal{F}}$ is the number of channels and S is the stride of the feature map ($C_{\mathcal{F}}$ depends on the specific backbone and we use $S = 32$ in this work). Then, a pixel decoder gradually upsamples the features to generate per-pixel embeddings $\mathcal{E}_{\text{pixel}} \in \mathbb{R}^{C_{\mathcal{E}} \times H \times W}$, where $C_{\mathcal{E}}$ is the embedding dimension. Note that any per-pixel classification-based segmentation model fits the pixel-level module design including recent Transformer-based models [96, 109, 110]. MaskFormer seamlessly converts such a model to mask classification.

Transformer module uses the standard Transformer decoder [93] to compute from image features $\mathcal{F}$ and N learnable positional embeddings (*i.e.*, queries) its output, *i.e.*, N per-segment embeddings $\mathcal{Q} \in \mathbb{R}^{C_{\mathcal{Q}} \times N}$ of dimension $C_{\mathcal{Q}}$ that encode global information about each segment MaskFormer predicts. Similarly to [91], the decoder yields all predictions in parallel.

Segmentation module applies a linear classifier, followed by a softmax activation, on top of the per-segment embeddings $\mathcal{Q}$ to yield class probability predictions $\{p_i \in \Delta^{K+1}\}_{i=1}^N$ for each segment. Note that the classifier predicts an additional “no object” category ($\emptyset$) in case the embedding does not correspond to any region. For mask prediction, a multi-layer perceptron (MLP) with 2 hidden layers converts the per-segment embeddings $\mathcal{Q}$ to N mask embeddings $\mathcal{E}_{\text{mask}} \in \mathbb{R}^{C_{\mathcal{E}} \times N}$ of dimension $C_{\mathcal{E}}$. Finally, we obtain each binary mask prediction $m_i \in [0, 1]^{H \times W}$ via a dot product between the i^{th} mask embedding and per-pixel embeddings $\mathcal{E}_{\text{pixel}}$ computed by the pixel-level module. The dot product is followed by a sigmoid activation, *i.e.*, $m_i[h, w] = \text{sigmoid}(\mathcal{E}_{\text{mask}}[:, i]^T \cdot \mathcal{E}_{\text{pixel}}[:, h, w])$.

Note that we empirically find it is beneficial to *not* enforce mask predictions to be mutually exclusive to each other by using a softmax activation. During training, the $\mathcal{L}_{\text{mask-cls}}$ loss combines a cross entropy classification loss and a binary mask loss $\mathcal{L}_{\text{mask}}$ for each predicted segment. For simplicity we use the same $\mathcal{L}_{\text{mask}}$ as DETR [91], *i.e.*, a linear combination of a focal loss [114] and a dice loss [115] multiplied by hyper-parameters λ_{focal} and λ_{dice} respectively.

3.3.4 Mask-classification Inference

First, we present a simple *general inference* procedure that converts mask classification outputs $\{(p_i, m_i)\}_{i=1}^N$ to either panoptic or semantic segmentation output formats. Then, we describe a *semantic inference* procedure specifically designed for semantic segmentation. We note that the specific choice of inference strategy largely depends on the evaluation metric rather than the task.

General inference partitions an image into segments by assigning each pixel $[h, w]$ to one of the N predicted probability-mask pairs via $\arg \max_{i: c_i \neq \emptyset} p_i(c_i) \cdot m_i[h, w]$. Here c_i is the most likely class label $c_i = \arg \max_{c \in \{1, \dots, K, \emptyset\}} p_i(c)$ for each probability-mask pair i . Intuitively, this procedure assigns a pixel at location $[h, w]$ to probability-mask pair i only if both the *most likely* class probability $p_i(c_i)$ and the mask prediction probability $m_i[h, w]$ are high. Pixels assigned to the same probability-mask pair i form a segment where each pixel is labelled with c_i . For semantic segmentation, segments sharing the same category label are merged; whereas for instance-level segmentation tasks, the index i of the probability-mask pair helps to distinguish different instances of the same class. Finally, to reduce false positive rates in panoptic segmentation we follow previous inference strategies [4, 91]. Specifically, we filter out low-confidence predictions prior to inference and remove predicted segments that have large parts of their binary masks ($m_i > 0.5$) occluded by other predictions.

Semantic inference is designed specifically for semantic segmentation and is done via a simple matrix multiplication. We empirically find that marginalization over probability-mask pairs, *i.e.*, $\arg \max_{c \in \{1, \dots, K\}} \sum_{i=1}^N p_i(c) \cdot m_i[h, w]$, yields better results than the hard assignment of each pixel to a probability-mask pair i used in the general inference strategy. The argmax does not include the “no object” category ($\emptyset$) as standard semantic segmentation requires each output pixel to take a label. Note that this strategy returns a per-pixel class probability $\sum_{i=1}^N p_i(c) \cdot m_i[h, w]$. However, we observe that directly maximizing per-pixel class likelihood leads to poor performance. We hypothesize that gradients are evenly distributed to every query, which complicates training.

3.4 Experiments

We demonstrate that MaskFormer seamlessly unifies semantic- and instance-level segmentation tasks by showing state-of-the-art results on both semantic segmentation and panoptic segmentation datasets. Then, we ablate the MaskFormer design confirming that observed improvements in semantic segmentation indeed stem from the shift from per-pixel classification to mask classification.

Datasets. We study MaskFormer using four widely used semantic segmentation datasets: ADE20K [94] (150 classes) from the SceneParse150 challenge [116], COCO-Stuff-10K [95] (171 classes), Cityscapes [12] (19 classes), and Mapillary Vistas [13] (65 classes). In addition, we use the ADE20K-Full [94] dataset annotated in an open vocabulary setting (we keep 874 classes that are present in both train and validation sets). For panoptic segmentation evaluation we use COCO [4, 11, 95] (80 “things” and 53 “stuff” categories) and ADE20K-Panoptic [4, 94] (100 “things” and 50 “stuff” categories). Please see Section 3.7 for detailed descriptions of all used datasets.

Evaluation metrics. For *semantic segmentation* the standard metric is **mIoU** (mean Intersection-over-Union) [2], a per-pixel metric that directly corresponds to the per-pixel classification formulation. To better illustrate the difference between segmentation approaches, in our ablations we supplement mIoU with **PQSt** (PQ stuff) [4], a per-region metric that treats all classes as “stuff” and evaluates each segment equally, irrespective of its size. We report the median of 3 runs for all datasets, except for Cityscapes where we report the median of 5 runs. For *panoptic segmentation*, we use the standard **PQ** (panoptic quality) metric [4] and report single run results due to prohibitive training costs.

Baseline models. We sketch the used per-pixel classification baselines in Figure 3.3. The **PerPixelBaseline** uses the pixel-level module of MaskFormer and directly outputs per-pixel class scores. For a fair comparison, we design **PerPixelBaseline+** which adds the transformer module and mask embedding MLP to the PerPixelBaseline. Thus, PerPixelBaseline+ and MaskFormer differ only in the formulation: per-pixel *vs.* mask classification. Note that these baselines are for ablation and we compare MaskFormer with state-of-the-art per-pixel classification models as well.

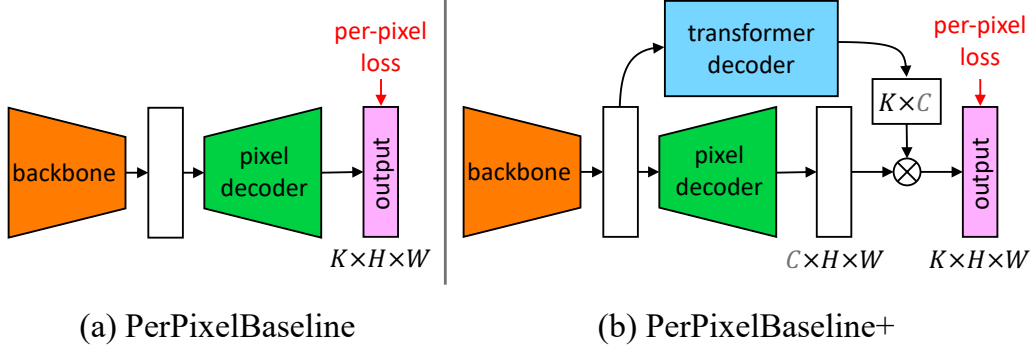


Figure 3.3: Per-pixel classification baselines.

3.4.1 Implementation Details

Backbone. MaskFormer is compatible with any backbone architecture. In our work we use the standard convolution-based ResNet [77] backbones (R50 and R101 with 50 and 101 layers respectively) and recently proposed Transformer-based Swin-Transformer [96] backbones. In addition, we use the R101c model [26] which replaces the first 7×7 convolution layer of R101 with 3 consecutive 3×3 convolutions and which is popular in the semantic segmentation community [6, 7, 21, 90, 107, 117].

Pixel decoder. The pixel decoder in Figure 3.2 can be implemented using any semantic segmentation decoder (*e.g.*, [6, 21, 118]). Many per-pixel classification methods use modules like ASPP [26] or PSP [90] to collect and distribute context across locations. The Transformer module attends to all image features, collecting global information to generate class predictions. This setup reduces the need of the per-pixel module for heavy context aggregation. Therefore, for MaskFormer, we design a lightweight pixel decoder based on the popular FPN [25] architecture.

Following FPN, we $2\times$ upsample the low-resolution feature map in the decoder and sum it with the projected feature map of corresponding resolution from the backbone; projection is done to match channel dimensions of the feature maps with a 1×1 convolution layer followed by GroupNorm (GN) [119]. Next, we fuse the summed features with an additional 3×3 convolution layer followed by GN and ReLU activation. We repeat this process starting with the stride 32 feature map until we obtain a final feature map of stride 4. Finally, we apply a single 1×1 convolution layer to get the per-pixel embeddings. All feature maps in the pixel decoder have a dimension of 256

channels.

Transformer decoder. We use the same Transformer decoder design as DETR [91]. The N query embeddings are initialized as zero vectors, and we associate each query with a learnable positional encoding. We use 6 Transformer decoder layers with 100 queries by default, and, following DETR, we apply the same loss after each decoder. In our experiments we observe that MaskFormer is competitive for semantic segmentation with a single decoder layer too, whereas for instance-level segmentation multiple layers are necessary to remove duplicates from the final predictions.

Segmentation module. The multi-layer perceptron (MLP) in Figure 3.2 has 2 hidden layers of 256 channels to predict the mask embeddings $\mathcal{E}_{\text{mask}}$, analogously to the box head in DETR. Both per-pixel $\mathcal{E}_{\text{pixel}}$ and mask $\mathcal{E}_{\text{mask}}$ embeddings have 256 channels.

Loss weights. We use focal loss [114] and dice loss [115] for our mask loss: $\mathcal{L}_{\text{mask}}(m, m^{\text{gt}}) = \lambda_{\text{focal}}\mathcal{L}_{\text{focal}}(m, m^{\text{gt}}) + \lambda_{\text{dice}}\mathcal{L}_{\text{dice}}(m, m^{\text{gt}})$, and set the hyper-parameters to $\lambda_{\text{focal}} = 20.0$ and $\lambda_{\text{dice}} = 1.0$. Following DETR [91], the weight for the “no object” ($\emptyset$) in the classification loss is set to 0.1.

3.4.2 Training Settings

Semantic segmentation. We use Detectron2 [78] and follow the commonly used training settings for each dataset. More specifically, we use AdamW [120] and the *poly* [26] learning rate schedule with an initial learning rate of 10^{-4} and a weight decay of 10^{-4} for ResNet [77] backbones, and an initial learning rate of $6 \cdot 10^{-5}$ and a weight decay of 10^{-2} for Swin-Transformer [96] backbones. Backbones are pre-trained on ImageNet-1K [121] if not stated otherwise. A learning rate multiplier of 0.1 is applied to CNN backbones and 1.0 is applied to Transformer backbones. The standard random scale jittering between 0.5 and 2.0, random horizontal flipping, random cropping as well as random color jittering are used as data augmentation [122]. For the ADE20K dataset, if not stated otherwise, we use a crop size of 512×512 , a batch size of 16 and train all models for 160k iterations. For the ADE20K-Full dataset, we use the same setting as ADE20K except that we train all models for 200k iterations. For the COCO-Stuff-10k dataset, we use a crop size of 640×640 , a batch size of 32 and train all models for

Table 3.1: **Semantic segmentation on ADE20K val with 150 categories.** Mask classification-based MaskFormer outperforms the best per-pixel classification approaches while using fewer parameters and less computation. We report both single-scale (s.s.) and multi-scale (m.s.) inference results with $\pm std$. FLOPs are computed for the given crop size. Frames-per-second (fps) is measured on a V100 GPU with a batch size of 1. Backbones pre-trained on ImageNet-22K are marked with $\dagger$.

	method	backbone	crop size	mIoU (s.s.)	mIoU (m.s.)	#params	FLOPs	fps
CNN backbones	OCRNet [117]	R101c	520×520	-	45.3	-	-	-
	DeepLabV3+ [21]	R50c	512×512	44.0	44.9	44M	177G	21.0
		R101c	512×512	45.5	46.4	63M	255G	14.2
	MaskFormer (ours)	R50	512×512	44.5 ± 0.5	46.7 ± 0.6	41M	53G	24.5
		R101	512×512	45.5 ± 0.5	47.2 ± 0.2	60M	73G	19.5
		R101c	512×512	46.0 ± 0.1	48.1 ± 0.2	60M	80G	19.0
Transformer backbones	SETR [109]	ViT-L $\dagger$	512×512	-	50.3	308M	-	-
	Swin-UperNet [96, 123]	Swin-T	512×512	-	46.1	60M	236G	18.5
		Swin-S	512×512	-	49.3	81M	259G	15.2
		Swin-B $\dagger$	640×640	-	51.6	121M	471G	8.7
		Swin-L $\dagger$	640×640	-	53.5	234M	647G	6.2
	MaskFormer (ours)	Swin-T	512×512	46.7 ± 0.7	48.8 ± 0.6	42M	55G	22.1
		Swin-S	512×512	49.8 ± 0.4	51.0 ± 0.4	63M	79G	19.6
		Swin-B	640×640	51.1 ± 0.2	52.3 ± 0.4	102M	195G	12.6
		Swin-B $\dagger$	640×640	52.7 ± 0.4	53.9 ± 0.2	102M	195G	12.6
		Swin-L $\dagger$	640×640	54.1 ± 0.2	55.6 ± 0.1	212M	375G	7.9

60k iterations. All models are trained with 8 V100 GPUs. We report both performance of single scale (s.s.) inference and multi-scale (m.s.) inference with horizontal flip and scales of 0.5, 0.75, 1.0, 1.25, 1.5, 1.75. See Section 3.7 for Cityscapes and Mapillary Vistas settings.

Panoptic segmentation. We follow exactly the same architecture, loss, and training procedure as we use for semantic segmentation. The only difference is supervision: *i.e.*, category region masks in semantic segmentation *vs.* object instance masks in panoptic segmentation. We strictly follow the DETR [91] setting to train our model on the COCO panoptic segmentation dataset [4] for a fair comparison. On the ADE20K panoptic segmentation dataset, we follow the semantic segmentation setting but train for longer (720k iterations) and use a larger crop size (640×640). COCO models are trained using 64 V100 GPUs and ADE20K experiments are trained with 8 V100 GPUs. We use the general inference (Section 3.3.4) with the following parameters: We filter out masks with class confidence below 0.8 and set masks whose contribution to the final panoptic segmentation is less than 80% of its mask area to VOID. We report performance of single scale inference.

Table 3.2: **MaskFormer *vs.* per-pixel classification baselines on 4 semantic segmentation datasets.** MaskFormer improvement is larger when the number of classes is larger. We use a ResNet-50 backbone and report single scale mIoU and PQSt for ADE20K, COCO-Stuff and ADE20K-Full, whereas for higher-resolution Cityscapes we use a deeper ResNet-101 backbone following [7, 21].

	Cityscapes (19 classes)		ADE20K (150 classes)		COCO-Stuff (171 classes)		ADE20K-Full (847 classes)	
	mIoU	PQ St	mIoU	PQ St	mIoU	PQ St	mIoU	PQ St
PerPixelBaseline	77.4	58.9	39.2	21.6	32.4	15.5	12.4	5.8
PerPixelBaseline+	78.5	60.2	41.9	28.3	34.2	24.6	13.9	9.0
MaskFormer (ours)	78.5 (+0.0)	63.1 (+2.9)	44.5 (+2.6)	33.4 (+5.1)	37.1 (+2.9)	28.9 (+4.3)	17.4 (+3.5)	11.9 (+2.9)

3.4.3 Main Results

Semantic segmentation. In Table 3.1, we compare MaskFormer with state-of-the-art per-pixel classification models for semantic segmentation on the ADE20K *val* set. With the same standard CNN backbones (*e.g.*, ResNet [77]), MaskFormer outperforms DeepLabV3+ [21] by 1.7 mIoU. MaskFormer is also compatible with recent Vision Transformer [111] backbones (*e.g.*, the Swin Transformer [96]), achieving a new state-of-the-art of 55.6 mIoU, which is 2.1 mIoU better than the prior state-of-the-art [96]. Observe that MaskFormer outperforms the best per-pixel classification-based models while having fewer parameters and faster inference time. This result suggests that the mask classification formulation has significant potential for semantic segmentation. See Section 3.7 for results on *test* set.

Beyond ADE20K, we further compare MaskFormer with our baselines on COCO-Stuff-10K, ADE20K-Full as well as Cityscapes in Table 3.2 and we refer the reader to Section 3.7 for comparison with state-of-the-art methods on these datasets. The improvement of MaskFormer over PerPixelBaseline+ is larger when the number of classes is larger: For Cityscapes, which has only 19 categories, MaskFormer performs similarly well as PerPixelBaseline+, while for ADE20K-Full, which has 847 classes, MaskFormer outperforms PerPixelBaseline+ by 3.5 mIoU.

Although MaskFormer shows no improvement in mIoU for Cityscapes, the PQSt metric increases by 2.9 PQSt. We find MaskFormer performs better in terms of recognition quality (RQSt) while lagging in per-pixel segmentation quality (SQSt) (refer to Section 3.7 for detailed numbers). This observation

Table 3.3: **Panoptic segmentation on COCO panoptic val with 133 categories.** MaskFormer seamlessly unifies semantic- and instance-level segmentation without modifying the model architecture or loss. Our model, which achieves better results, can be regarded as a box-free simplification of DETR [91]. The major improvement comes from “stuff” classes (PQ^{St}) which are ambiguous to represent with bounding boxes. For MaskFormer (DETR) we use the exact same post-processing as DETR. Note that in this setting MaskFormer performance is still better than DETR (+2.2 PQ). Our model also outperforms recently proposed Max-DeepLab [97] without the need of sophisticated auxiliary losses, while being more efficient. FLOPs are computed as the average FLOPs over 100 validation images (COCO images have varying sizes). Frames-per-second (fps) is measured on a V100 GPU with a batch size of 1 by taking the average runtime on the entire val set *including post-processing time*. Backbones pre-trained on ImageNet-22K are marked with $\dagger$.

	method	backbone	PQ	PQ^{Th}	PQ^{St}	SQ	RQ	param	FLOPs	fps
CNN	DETR [91]	R50 + 6 Enc	43.4	48.2	36.3	79.3	53.8	-	-	-
	MaskFormer (DETR)	R50 + 6 Enc	45.6	50.0 (+1.8)	39.0 (+2.7)	80.2	55.8	-	-	-
	MaskFormer (ours)	R50 + 6 Enc	46.5	51.0 (+2.8)	39.8 (+3.5)	80.4	56.8	45M	181G	17.6
	DETR [91]	R101 + 6 Enc	45.1	50.5	37.0	79.9	55.5	-	-	-
	MaskFormer (ours)	R101 + 6 Enc	47.6	52.5 (+2.0)	40.3 (+3.3)	80.7	58.0	64M	248G	14.0
Transformer	Max-DeepLab [97]	Max-S	48.4	53.0	41.5	-	-	62M	324G	7.6
		Max-L	51.1	57.0	42.2	-	-	451M	3692G	-
	MaskFormer (ours)	Swin-T	47.7	51.7	41.7	80.4	58.3	42M	179G	17.0
		Swin-S	49.7	54.4	42.6	80.9	60.4	63M	259G	12.4
		Swin-B	51.1	56.3	43.2	81.4	61.8	102M	411G	8.4
		Swin-B $\dagger$	51.8	56.9	44.1	81.4	62.6	102M	411G	8.4
		Swin-L $\dagger$	52.7	58.5	44.0	81.8	63.5	212M	792G	5.2

suggests that on datasets where class recognition is relatively easy to solve, the main challenge for mask classification-based approaches is pixel-level accuracy (*i.e.*, mask quality).

Panoptic segmentation. In Table 3.3, we compare the same exact MaskFormer model with DETR [91] on the COCO panoptic val set. To match the standard DETR design, we add 6 additional Transformer encoder layers after the CNN backbone. Unlike DETR, our model does not predict bounding boxes but instead predicts masks directly. MaskFormer achieves better results while being simpler than DETR. To disentangle the improvements from the model itself and our post-processing inference strategy we run our model following DETR post-processing (MaskFormer (DETR)) and observe that this setup outperforms DETR by 2.2 PQ. Overall, we observe a larger improvement in PQ^{St} compared to PQ^{Th} . This suggests that de-

Table 3.4: **Per-pixel *vs.* mask classification for semantic segmentation.** All models use 150 queries for a fair comparison. We evaluate the models on ADE20K val with 150 categories. 3.4a: PerPixelBaseline+ and MaskFormer-fixed use similar fixed matching (*i.e.*, matching by category index), this result confirms that the shift from per-pixel to mask classification is the key. 3.4b: bipartite matching is not only more flexible (can make less prediction than total class count) but also gives better results.

(a) Per-pixel *vs.* mask classification.

	mIoU	PQ St
PerPixelBaseline+	41.9	28.3
MaskFormer-fixed	43.7 (+1.8)	30.3 (+2.0)

(b) Fixed *vs.* bipartite matching assignment.

	mIoU	PQ St
MaskFormer-fixed	43.7	30.3
MaskFormer-bipartite (ours)	44.2 (+0.5)	33.4 (+3.1)

tecting “stuff” with bounding boxes is suboptimal, and therefore, box-based segmentation models (*e.g.*, Mask R-CNN [5]) do not suit semantic segmentation. MaskFormer also outperforms recently proposed Max-DeepLab [97] without the need of special network design as well as sophisticated auxiliary losses (*i.e.*, instance discrimination loss, mask-ID cross entropy loss, and per-pixel classification loss in [97]). *MaskFormer, for the first time, unifies semantic- and instance-level segmentation with the exact same model, loss, and training pipeline.*

We further evaluate our model on the panoptic segmentation version of the ADE20K dataset. Our model also achieves state-of-the-art performance. Refer to Section 3.7 for detailed results.

3.4.4 Ablation Studies

We perform a series of ablation studies of MaskFormer using a single ResNet-50 backbone [77].

Per-pixel *vs.* mask classification. In Table 3.4, we verify that the gains demonstrated by MaskFormer come from shifting the paradigm to mask classification. We start by comparing PerPixelBaseline+ and MaskFormer. The models are very similar and there are only 3 differences: (1) per-pixel *vs.* mask classification used by the models, (2) MaskFormer uses bipartite matching,

Table 3.5: **Number of queries.** The model with 100 queries consistently performs the best across the studied datasets.

# of queries	ADE20K		COCO-Stuff		ADE20K-Full	
	mIoU	PQ St	mIoU	PQ St	mIoU	PQ St
PerPixelBaseline+	41.9	28.3	34.2	24.6	13.9	9.0
20	42.9	32.6	35.0	27.6	14.1	10.8
50	43.9	32.7	35.5	27.9	15.4	11.1
100	44.5	33.4	37.1	28.9	16.0	11.9
150	44.2	33.4	37.0	28.9	15.5	11.5
300	43.5	32.3	36.1	29.1	14.2	10.3
1000	35.4	26.7	34.4	27.6	8.0	5.8

and (3) the new model uses a combination of focal and dice losses as a mask loss, whereas PerPixelBaseline+ utilizes per-pixel cross entropy loss. First, we rule out the influence of loss differences by training PerPixelBaseline+ with exactly the same losses and observing no improvement. Next, in Table 3.4a, we compare PerPixelBaseline+ with MaskFormer trained using a fixed matching (MaskFormer-fixed), *i.e.*, $N = K$ and assignment done based on category label indices identically to the per-pixel classification setup. We observe that MaskFormer-fixed is 1.8 mIoU better than the baseline, suggesting that shifting from per-pixel classification to mask classification is indeed the main reason for the gains of MaskFormer. In Table 3.4b, we further compare MaskFormer-fixed with MaskFormer trained with bipartite matching (MaskFormer-bipartite) and find bipartite matching is not only more flexible (allowing to predict less masks than the total number of categories) but also produces better results.

Number of queries. Table 3.5 shows results of MaskFormer trained with a varying number of queries on datasets with different numbers of categories. The model with 100 queries consistently performs the best across the studied datasets. This suggest we may not need to adjust the number of queries w.r.t. the number of categories or datasets much. Interestingly, even with 20 queries MaskFormer outperforms our per-pixel classification baseline.

We further calculate the number of classes which are on average present in a *training set* image. We find these statistics to be similar across datasets despite the fact that the datasets have different numbers of total categories: 8.2 classes per image for ADE20K (150 classes), 6.6 classes per image for COCO-Stuff-10K (171 classes) and 9.1 classes per image for ADE20K-Full (847 classes). We hypothesize that each query is able to capture masks from multiple categories.

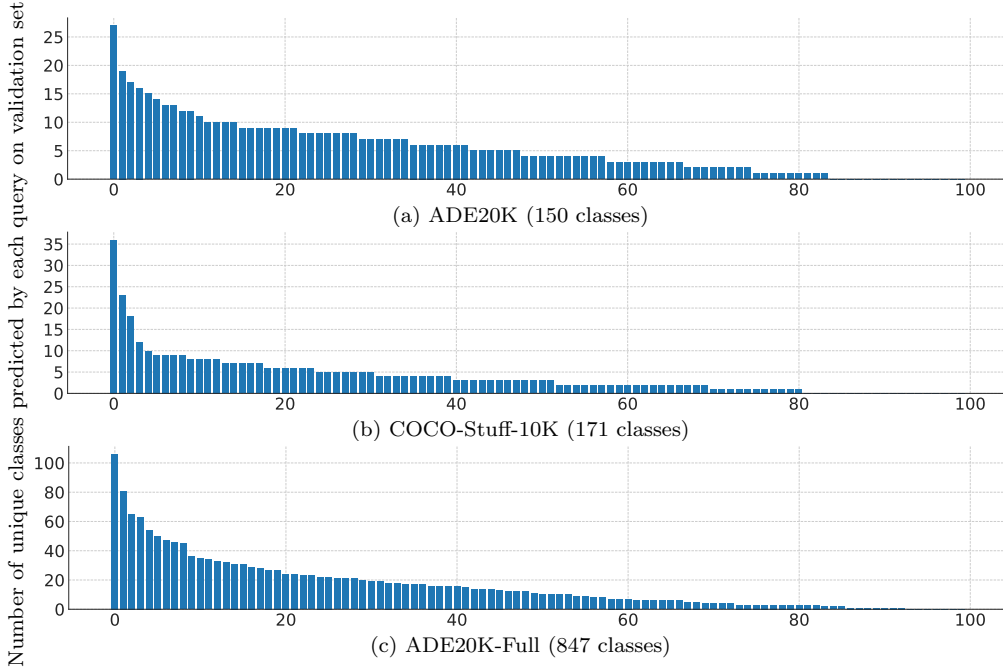


Figure 3.4: Per-pixel classification baselines.

Figure 3.4 shows the number of unique categories predicted by each query (sorted in descending order) of our MaskFormer model on the validation sets of the corresponding datasets. Interestingly, the number of unique categories per query does not follow a uniform distribution: Some queries capture more classes than others. We try to analyze how MaskFormer queries group categories, but we do not observe any obvious pattern: There are queries capturing categories with similar semantics or shapes (*e.g.*, “house” and “building”), but there are also queries capturing completely different categories (*e.g.*, “water” and “sofa”).

Number of Transformer decoder layers. Interestingly, MaskFormer with even a single Transformer decoder layer already performs well for semantic segmentation and achieves better performance than our 6-layer-decoder PerPixelBaseline+. For panoptic segmentation, however, multiple decoder layers are required to achieve competitive performance. Please see Section 3.7 for a detailed discussion.

Table 3.6: **Matching with masks *vs.* boxes.** We compare DETR [91] which uses box-based matching with two MaskFormer models trained with box- and mask-based matching respectively. To use box-based matching in MaskFormer we add to the model an additional box prediction head as in DETR. Note, that with box-based matching MaskFormer performs on par with DETR, whereas with mask-based matching it shows better results. The evaluation is done on COCO panoptic **val set**.

method	backbone	matching	PQ	PQ Th	PQ St
DETR [91]	R50 + 6 Enc	by box	43.4	48.2	36.3
MaskFormer (ours)	R50 + 6 Enc	by box	43.7	49.2	35.3
	R50 + 6 Enc	by mask	46.5	51.0	39.8

3.5 Discussion

Our main goal is to show that mask classification is a general segmentation paradigm that could be a competitive alternative to per-pixel classification for semantic segmentation. To better understand its potential for segmentation tasks, we focus on exploring mask classification independently of other factors like architecture, loss design, or augmentation strategy. We pick the DETR [91] architecture as our baseline for its simplicity and deliberately make as few architectural changes as possible. Therefore, MaskFormer can be viewed as a “box-free” version of DETR.

In this section, we discuss in detail the differences between MaskFormer and DETR and show how these changes are required to ensure that mask classification performs well. First, to achieve a pure mask classification setting we remove the box prediction head and perform matching between prediction and ground truth segments with masks instead of boxes. Secondly, we replace the compute-heavy *per-query* mask head used in DETR with a more efficient *per-image* FPN-based head to make end-to-end training without box supervision feasible.

Matching with masks is superior to matching with boxes. We compare MaskFormer models trained using matching with boxes or masks in Table 3.6. To do box-based matching, we add to MaskFormer an additional box prediction head as in DETR [91]. Observe that MaskFormer, which directly matches with mask predictions, has a clear advantage. We hypothesize that matching with boxes is more ambiguous than matching with masks, especially for stuff categories where completely different masks can have similar boxes as stuff regions often spread over a large area in an image.

MaskFormer mask head reduces computation. Results in Table 3.6 also show that MaskFormer performs on par with DETR when the same matching strategy is used. This suggests that the difference in mask head designs between the models does not significantly influence the prediction quality. The new head, however, has significantly lower computational and memory costs in comparison with the original mask head used in DETR. In MaskFormer, we first upsample image features to get high-resolution per-pixel embeddings and directly generate binary mask predictions at a high-resolution. Note that the per-pixel embeddings from the upsampling module (*i.e.*, pixel decoder) are shared among all queries. In contrast, DETR first generates low-resolution attention maps and applies an independent upsampling module to each query. Thus, the mask head in DETR is N times more computationally expensive than the mask head in MaskFormer (where N is the number of queries).

3.6 Conclusion

The paradigm discrepancy between semantic- and instance-level segmentation results in entirely different models for each task, hindering development of image segmentation as a whole. We show that a simple mask classification model can outperform state-of-the-art per-pixel classification models, especially in the presence of large number of categories. Our model also remains competitive for panoptic segmentation, without a need to change model architecture, losses, or training procedure. We hope this unification spurs a joint effort across semantic- and instance-level segmentation tasks.

3.7 Supplementary Materials

We first provide more information regarding the datasets used in our experimental evaluation of MaskFormer (Section 3.7.1). Then, we provide detailed results of our model on more semantic (Section 3.7.2) and panoptic (Section 3.7.3) segmentation datasets. Finally, we provide additional ablation studies (Section 3.7.4) and visualization (Section 3.7.5).

3.7.1 Datasets Description

We study MaskFormer using five semantic segmentation datasets and two panoptic segmentation datasets. Here, we provide more detailed information about these datasets.

3.7.1.1 Semantic segmentation datasets

ADE20K [94] contains 20k images for training and 2k images for validation. The data comes from the ADE20K-Full dataset where 150 semantic categories are selected to be included in evaluation from the SceneParse150 challenge [116]. The images are resized such that the shortest side is no greater than 512 pixels. During inference, we resize the shorter side of the image to the corresponding crop size.

COCO-Stuff-10K [95] has 171 semantic-level categories. There are 9k images for training and 1k images for testing. Images in the COCO-Stuff-10K datasets are a subset of the COCO dataset [11]. During inference, we resize the shorter side of the image to the corresponding crop size.

ADE20K-Full [94] contains 25k images for training and 2k images for validation. The ADE20K-Full dataset is annotated in an open-vocabulary setting with more than 3000 semantic categories. We filter these categories by selecting those that are present in both training and validation sets, resulting in a total of 847 categories. We follow the same process as ADE20K-SceneParse150 to resize images such that the shortest side is no greater than 512 pixels. During inference, we resize the shorter side of the image to the corresponding crop size.

Cityscapes [12] is an urban egocentric street-view dataset with high-resolution images (1024×2048 pixels). It contains 2975 images for training, 500 images for validation, and 1525 images for testing with a total of 19 classes. During training, we use a crop size of 512×1024 , a batch size of 16 and train all models for 90k iterations. During inference, we operate on the whole image (1024×2048).

Mapillary Vistas [13] is a large-scale urban street-view dataset with 65 categories. It contains 18k, 2k, and 5k images for training, validation and testing with a variety of image resolutions, ranging from 1024×768 to 4000×6000 . During training, we resize the short side of images to 2048 before

applying scale augmentation. We use a crop size of 1280×1280 , a batch size of 16 and train all models for 300k iterations. During inference, we resize the longer side of the image to 2048 and only use three scales (0.5, 1.0 and 1.5) for multi-scale testing due to GPU memory constraints.

3.7.1.2 Panoptic segmentation datasets

COCO panoptic [4] is one of the most commonly used datasets for panoptic segmentation. It has 133 categories (80 “thing” categories with instance-level annotation and 53 “stuff” categories) in 118k images for training and 5k images for validation. All images are from the COCO dataset [11].

ADE20K panoptic [94] combines the ADE20K semantic segmentation annotation for semantic segmentation from the SceneParse150 challenge [116] and ADE20K instance annotation from the COCO+Places challenge [124]. Among the 150 categories, there are 100 “thing” categories with instance-level annotation. We find filtering masks with a lower threshold (we use 0.7 for ADE20K) than COCO (which uses 0.8) gives slightly better performance.

Table 3.7: **Semantic segmentation on ADE20K test with 150 categories.** MaskFormer outperforms previous state-of-the-art methods on all three metrics: pixel accuracy (P.A.), mIoU, as well as the final test score (average of P.A. and mIoU). We train our model on the union of ADE20K **train** and **val** set with ImageNet-22K pre-trained checkpoint following [96] and use multi-scale inference.

method	backbone	P.A.	mIoU	score
SETR [109]	ViT-L	78.35	45.03	61.69
Swin-UperNet [96, 123]	Swin-L	78.42	47.07	62.75
MaskFormer (ours)	Swin-L	79.36	49.67	64.51

3.7.2 Semantic Segmentation Results

ADE20K test. Table 3.7 compares MaskFormer with previous state-of-the-art methods on the ADE20K **test** set. Following [96], we train MaskFormer on the union of ADE20K **train** and **val** set with ImageNet-22K pre-trained checkpoint and use multi-scale inference. MaskFormer outperforms previous state-of-the-art methods on all three metrics with a large margin.

Table 3.8: **Semantic segmentation on COCO-Stuff-10K test with 171 categories and ADE20K-Full val with 847 categories.**

Table 3.8a: MaskFormer is competitive on COCO-Stuff-10K, showing the generality of mask-classification. Table 3.8b: MaskFormer results on the harder large-vocabulary semantic segmentation. MaskFormer performs better than per-pixel classification and requires less memory during training, thanks to decoupling the number of masks from the number of classes. mIoU (s.s.) and mIoU (m.s.) are the mIoU of single-scale and multi-scale inference with $\pm std$.

(a) COCO-Stuff-10K.				(b) ADE20K-Full.	
method	backbone	mIoU (s.s.)	mIoU (m.s.)	mIoU (s.s.)	training memory
OCRNet [117]	R101c	-	39.5	-	-
PerPixelBaseline	R50	32.4 $\pm$ 0.2	34.4 $\pm$ 0.4	12.4 $\pm$ 0.2	8030M
PerPixelBaseline+	R50	34.2 $\pm$ 0.2	35.8 $\pm$ 0.4	13.9 $\pm$ 0.1	26698M
MaskFormer (ours)	R50	37.1 $\pm$ 0.4	38.9 $\pm$ 0.2	16.0 $\pm$ 0.3	6529M
	R101	38.1 $\pm$ 0.3	39.8 $\pm$ 0.6	16.8 $\pm$ 0.2	6894M
	R101c	38.0 $\pm$ 0.3	39.3 $\pm$ 0.4	17.4 $\pm$ 0.4	6904M

Table 3.9: **Semantic segmentation on Cityscapes val with 19 categories.** 3.9a: MaskFormer is on-par with state-of-the-art methods on Cityscapes which has fewer categories than other considered datasets. We report multi-scale (m.s.) inference results with $\pm std$ for a fair comparison across methods. 3.9b: We analyze MaskFormer with a complimentary PQ^{St} metric, by treating all categories as “stuff.” The breakdown of PQ^{St} suggests mask classification-based MaskFormer is better at recognizing regions (RQ^{St}) while slightly lagging in generation of high-quality masks (SQ^{St}).

(a) Cityscapes standard mIoU metric.			(b) Cityscapes analysis with PQ^{St} metric suit.		
method	backbone	mIoU (m.s.)	PQ^{St} (m.s.)	SQ^{St} (m.s.)	RQ^{St} (m.s.)
Panoptic-DeepLab [6]	X71 [70]	81.5	66.6	82.9	79.4
OCRNet [117]	R101c	82.0	66.1	82.6	79.1
MaskFormer (ours)	R101	80.3 $\pm$ 0.1	65.9	81.5	79.7
	R101c	81.4 $\pm$ 0.2	66.9	82.0	80.5

COCO-Stuff-10K. Table 3.8a compares MaskFormer with our baselines as well as the state-of-the-art OCRNet model [117] on the COCO-Stuff-10K [95] dataset. MaskFormer outperforms our per-pixel classification baselines by a large margin and achieves competitive performances compared to OCRNet. These results demonstrate the generality of the MaskFormer model.

ADE20K-Full. We further demonstrate the benefits in large-vocabulary

Table 3.10: **Semantic segmentation on Mapillary Vistas val with 65 categories.** MaskFormer outperforms per-pixel classification methods on high-resolution images without the need of multi-scale inference, thanks to global context captured by the Transformer decoder. mIoU (s.s.) and mIoU (m.s.) are the mIoU of single-scale and multi-scale inference.

method	backbone	mIoU (s.s.)	mIoU (m.s.)
DeepLabV3+ [21]	R50	47.7	49.4
HMSANet [125]	R50	-	52.2
MaskFormer (ours)	R50	53.1	55.4

Table 3.11: **Panoptic segmentation on COCO panoptic test-dev with 133 categories.** MaskFormer outperforms previous state-of-the-art Max-DeepLab [97] on the **test-dev** set as well. We only train our model on the COCO **train2017** set with ImageNet-22K pre-trained checkpoint.

method	backbone	PQ	PQ Th	PQ St	SQ	RQ
Max-DeepLab [97]	Max-L	51.3	57.2	42.4	82.5	61.3
MaskFormer (ours)	Swin-L	53.3	59.1	44.5	82.0	64.1

semantic segmentation in Table 3.8b. Since we are the first to report performance on this dataset, we only compare MaskFormer with our per-pixel classification baselines. MaskFormer not only achieves better performance, but is also more memory efficient on the ADE20K-Full dataset with 847 categories, thanks to decoupling the number of masks from the number of classes. These results show that our MaskFormer has the potential to deal with real-world segmentation problems with thousands of categories.

Cityscapes. In Table 3.9a, we report MaskFormer performance on Cityscapes, the standard testbed for modern semantic segmentation methods. The dataset has only 19 categories, and therefore the recognition aspect of the dataset is less challenging than in other considered datasets. We observe that MaskFormer performs on par with the best per-pixel classification methods. To better analyze MaskFormer, in Table 3.9b, we further report PQSt. We find MaskFormer performs better in terms of recognition quality (RQSt) while lagging in per-pixel segmentation quality (SQSt). This suggests that on datasets, where recognition is relatively easy to solve, the main challenge for mask classification-based approaches is pixel-level accuracy.

Mapillary Vistas. Table 3.10 compares MaskFormer with state-of-the-art per-pixel classification models on the high-resolution Mapillary Vistas dataset which contains images up to 4000×6000 resolution. We observe: (1)

Table 3.12: **Panoptic segmentation on ADE20K panoptic val with 150 categories.** Following DETR [91], we add 6 additional Transformer encoders when using ResNet [77] (R50 + 6 Enc and R101 + 6 Enc) backbones. MaskFormer achieves competitive results on ADE20K panoptic, showing the generality of our model for panoptic segmentation.

method	backbone	PQ	PQ Th	PQ St	SQ	RQ
BGRNet [126]	R50	31.8	-	-	-	-
Auto-Panoptic [127]	ShuffleNetV2 [128]	32.4	-	-	-	-
MaskFormer (ours)	R50 + 6 Enc	34.7	32.2	39.7	76.7	42.8
	R101 + 6 Enc	35.7	34.5	38.0	77.4	43.8

MaskFormer is able to handle high-resolution images, and (2) MaskFormer outperforms multi-scale per-pixel classification models even without the need of multi-scale inference. We believe the Transformer decoder in MaskFormer is able to capture global context even for high-resolution images.

3.7.3 Panoptic Segmentation Results

COCO panoptic test-dev. Table 3.11 compares MaskFormer with previous state-of-the-art methods on the COCO panoptic **test-dev** set. We only train our model on the COCO **train2017** set with ImageNet-22K pre-trained checkpoint and outperforms previous state-of-the-art by 2 PQ.

ADE20K panoptic. We demonstrate the generality of our model for panoptic segmentation on the ADE20K panoptic dataset in Table 3.12, where MaskFormer is competitive with the state-of-the-art methods.

3.7.4 Additional Ablation Studies

We perform additional ablation studies of MaskFormer for semantic segmentation using the same setting: a single ResNet-50 backbone [77], and we report both the mIoU and the PQSt. The default setting of our MaskFormer is: 100 queries and 6 Transformer decoder layers.

Inference strategies. In Table 3.13, we ablate inference strategies for mask classification-based models performing semantic segmentation (discussed in Section 3.3.4). We compare our default semantic inference strategy and the general inference strategy which first filters out low-confidence masks (a threshold of 0.3 is used) and assigns the class labels to the remaining

Table 3.13: **Inference strategies for semantic segmentation.** *General*: general inference (Section 3.3.4) which first filters low-confidence masks (using a threshold of 0.3) and assigns labels to the remaining ones. *Semantic*: the default semantic inference (Section 3.3.4) for semantic segmentation.

inference	ADE20K (150 classes)				COCO-Stuff (171 classes)				ADE20K-Full (847 classes)			
	mIoU	PQ St	SQ St	RQ St	mIoU	PQ St	SQ St	RQ St	mIoU	PQ St	SQ St	RQ St
PerPixelBaseline+	41.9	28.3	71.9	36.2	34.2	24.6	62.6	31.2	13.9	9.0	24.5	12.0
general	42.4	34.2	74.4	43.5	35.5	29.7	66.3	37.0	15.1	11.6	28.3	15.3
semantic	44.5	33.4	75.4	42.4	37.1	28.9	66.3	35.9	16.0	11.9	28.6	15.7

Table 3.14: **Ablation on number of Transformer decoder layers in MaskFormer.** We find that MaskFormer with only one Transformer decoder layer is already able to achieve reasonable semantic segmentation performance. Stacking more decoder layers mainly improves the recognition quality.

# of decoder layers	ADE20K-Semantic				ADE20K-Panoptic				
	mIoU	PQ St	SQ St	RQ St	PQ	PQ Th	PQ St	SQ	RQ
6 (PerPixelBaseline+)	41.9	28.3	71.9	36.2	-	-	-	-	-
1	43.0	31.1	74.3	39.7	31.9	29.6	36.6	76.6	39.6
6	44.5	33.4	75.4	42.4	34.7	32.2	39.7	76.7	42.8
6 (no self-attention)	44.6	32.8	74.5	41.5	32.6	29.9	38.2	75.6	40.4

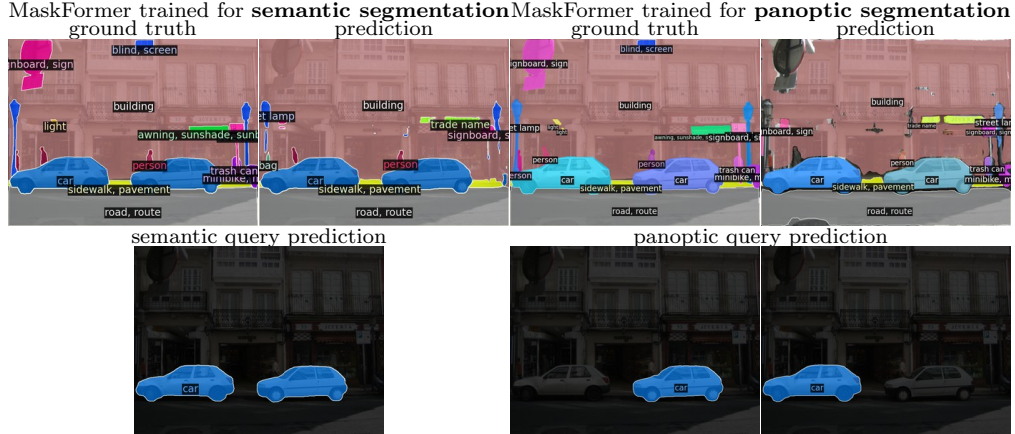


Figure 3.5: **Visualization of “semantic” queries and “panoptic” queries.** Unlike the behavior in a MaskFormer model trained for panoptic segmentation (right), a single query is used to capture multiple instances in a MaskFormer model trained for semantic segmentation (left). Our model has the capacity to adapt to different types of tasks given different ground truth annotations.

masks. We observe (1) general inference is only slightly better than the Per-PixelBaseline+ in terms of the mIoU metric, and (2) on multiple datasets the

general inference strategy performs worse in terms of the mIoU metric than the default semantic inference. However, the general inference has higher PQ^{St} , due to better recognition quality (RQ^{St}). We hypothesize that the filtering step removes false positives which increases the RQ^{St} . In contrast, the semantic inference aggregates mask predictions from multiple queries and thus has better mask quality (SQ^{St}). This observation suggests that semantic and instance-level segmentation can be unified with a single inference strategy (*i.e.*, our general inference) and *the choice of inference strategy largely depends on the evaluation metric instead of the task.*

Number of Transformer decoder layers. In Table 3.14, we ablate the effect of the number of Transformer decoder layers on ADE20K [94] for both semantic and panoptic segmentation. Surprisingly, we find a MaskFormer with even a single Transformer decoder layer already performs reasonably well for semantic segmentation and achieves better performance than our 6-layer-decoder per-pixel classification baseline PerPixelBaseline+. In contrast, for panoptic segmentation the number of decoder layers is more important. We hypothesize that stacking more decoder layers is helpful to de-duplicate predictions which is required by the panoptic segmentation task.

To verify this hypothesis, we train MaskFormer models *without* self-attention in all 6 Transformer decoder layers. On semantic segmentation, we observe MaskFormer without self-attention performs similarly well in terms of the mIoU metric, however the per-mask metric PQ^{St} is slightly worse. On panoptic segmentation, MaskFormer models without self-attention performs worse across all metrics.

“Semantic” queries vs. “panoptic” queries. In Figure 3.5 we visualize predictions for the “car” category from MaskFormer trained with semantic-level and instance-level ground truth data. In the case of semantic-level data, the matching cost and loss used for mask prediction force a single query to predict one mask that combines all cars together. In contrast, with instance-level ground truth, MaskFormer uses different queries to make mask predictions for each car. This observation suggests that our model has the capacity to adapt to different types of tasks given different ground truth annotations.

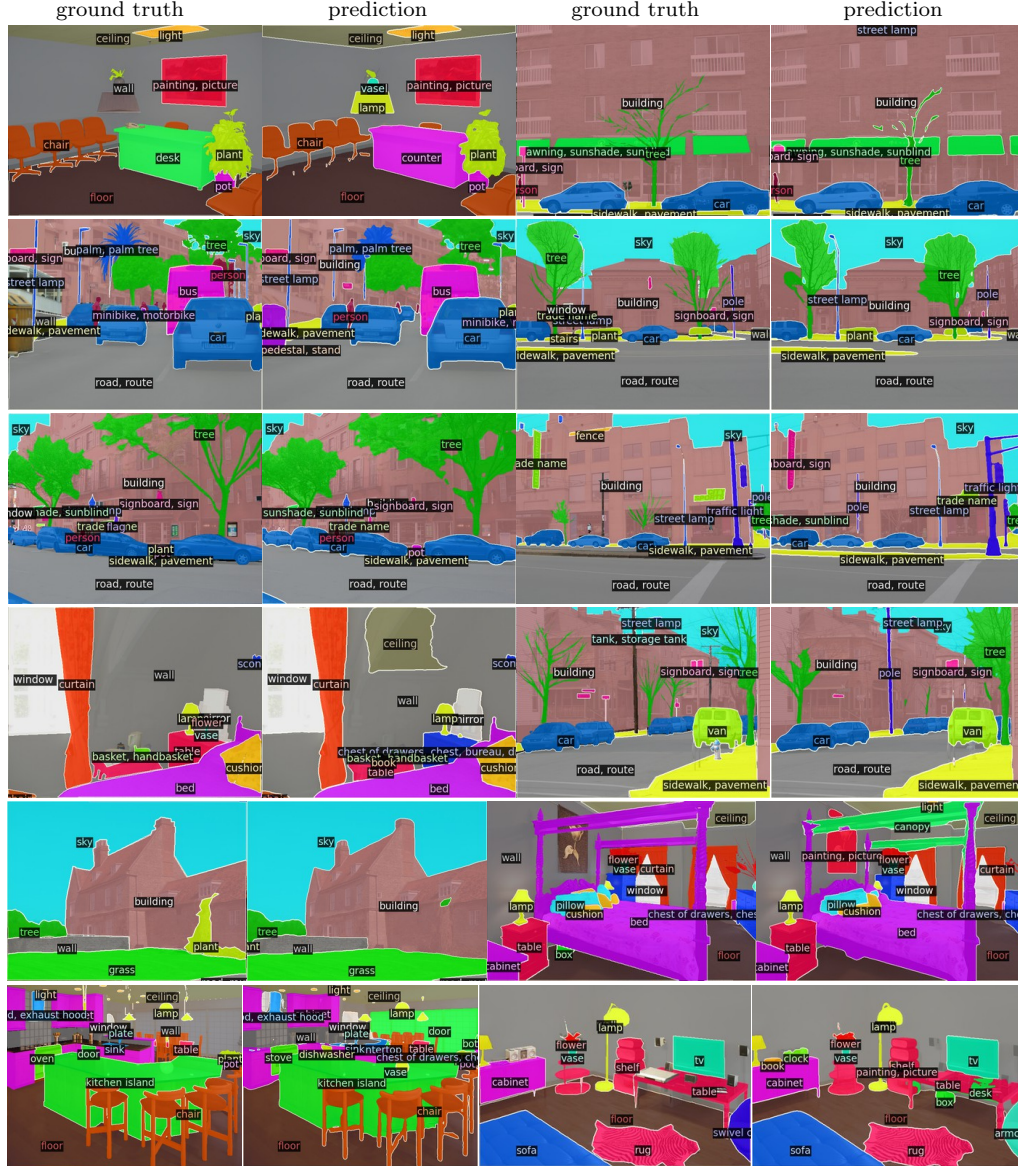


Figure 3.6: **Visualization of MaskFormer** semantic segmentation predictions on the ADE20K dataset. We visualize the MaskFormer with Swin-L backbone which achieves 55.6 mIoU (multi-scale) on the validation set. First and third columns: ground truth. Second and fourth columns: prediction.

3.7.5 Visualization

We visualize sample semantic segmentation predictions of the MaskFormer model with Swin-L [96] backbone (55.6 mIoU) on the ADE20K validation set in Figure 3.6.

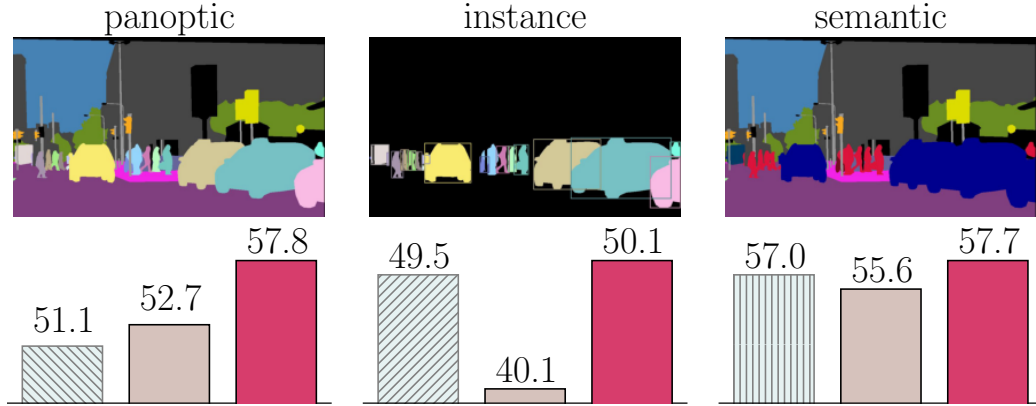
CHAPTER 4

UNIVERSAL IMAGE SEGMENTATION

4.1 Introduction

Image segmentation studies the problem of grouping pixels. Different semantics for grouping pixels, *e.g.*, category or instance membership, have led to different types of segmentation tasks, such as panoptic, instance or semantic segmentation. While these tasks differ only in semantics, current methods develop specialized architectures for each task. Per-pixel classification architectures based on fully convolutional networks (FCNs) [89] are used for semantic segmentation, while mask classification architectures [5, 91] that predict a set of binary masks, each associated with a single category, dominate instance-level segmentation. Although such *specialized* architectures [5, 21, 89, 129] have advanced each individual task, they lack the flexibility to generalize to the other tasks. For example, FCN-based architectures struggle at instance segmentation, leading to the evolution of different architectures for instance segmentation compared to semantic segmentation. Thus, duplicate research and (hardware) optimization effort is spent on each specialized architecture for every task.

To address this fragmentation, recent work [8, 130] has attempted to design *universal architectures* that are capable of addressing all segmentation tasks with the same architecture (*i.e.*, universal image segmentation). These architectures are typically based on an end-to-end set prediction objective (*e.g.*, DETR [91]), and successfully tackle multiple tasks without modifying the architecture, loss, or the training procedure. Note that universal architectures are still trained separately for different tasks and datasets, albeit having the same architecture. In addition to being flexible, universal architectures have recently shown state-of-the-art results on semantic and panoptic segmentation [8]. However, recent work still focuses on advancing



Universal architectures:

Mask2Former (ours) MaskFormer

SOTA specialized architectures:

Max-DeepLab Swin-HTC++ BEiT

Figure 4.1: State-of-the-art segmentation architectures are typically specialized for each image segmentation task. Although recent work has proposed universal architectures that attempt all tasks and are competitive on semantic and panoptic segmentation, they struggle with segmenting instances. We propose **Mask2Former**, which, for the first time, outperforms the best specialized architectures on three studied segmentation tasks on multiple datasets.

specialized architectures [110, 131, 132], which raises the question: Why have universal architectures not replaced specialized ones?

Although existing universal architectures are flexible enough to tackle any segmentation task, as shown in Figure 4.1, in practice their performance lags behind the best specialized architectures. For instance, the best reported performance of universal architectures [8, 130] is currently lower (> 9 AP) than the SOTA specialized architecture for instance segmentation [129]. Beyond the inferior performance, universal architectures are also harder to train. They typically require more advanced hardware and a much longer training schedule. For example, training MaskFormer [8] takes 300 epochs to reach 40.1 AP and it can only fit a single image in a GPU with 32G memory. In contrast, the specialized Swin-HTC++ [129] obtains better performance in only 72 epochs. Both the performance and training efficiency issues hamper the deployment of universal architectures.

In this work, we propose a universal image segmentation architecture named Masked-attention Mask Transformer (**Mask2Former**) that outperforms specialized architectures across different segmentation tasks, while still being easy to train on every task. We build upon a simple meta architecture [8] consisting of a backbone feature extractor [77,96], a pixel decoder [25] and a Transformer decoder [93]. We propose key improvements that enable better results and efficient training. First, we use *masked attention* in the Transformer decoder which restricts the attention to localized features centered around predicted segments, which can be either objects or regions depending on the specific semantic for grouping. Compared to the cross-attention used in a standard Transformer decoder which attends to all locations in an image, our masked attention leads to faster convergence and improved performance. Second, we use *multi-scale high-resolution features* which help the model to segment small objects/regions. Third, we propose *optimization improvements* such as switching the order of self- and cross-attention, making query features learnable, and removing dropout, all of which improve performance without additional compute. Finally, we save $3\times$ training memory without affecting the performance by *calculating mask loss on few randomly sampled points*. These improvements not only boost the model performance, but also make training significantly easier, making universal architectures more accessible to users with limited compute.

We evaluate Mask2Former on three image segmentation tasks (panoptic, instance and semantic segmentation) using four popular datasets (COCO [11], Cityscapes [12], ADE20K [94] and Mapillary Vistas [13]). For the first time, on all these benchmarks, our single architecture performs on par with or better than specialized architectures. Mask2Former sets the new state-of-the-art of **57.8 PQ** on COCO panoptic segmentation [4], **50.1 AP** on COCO instance segmentation [11] and **57.7 mIoU** on ADE20K semantic segmentation [94] using the exact same architecture.

4.2 Related Work

Specialized semantic segmentation architectures typically treat the task as a per-pixel classification problem. FCN-based architectures [89] independently predict a category label for every pixel. Follow-up methods

find context to play an important role for consistent and precise per-pixel classification and focus on designing customized context modules [7, 26, 90] or self-attention variants [105–110]. **Specialized instance segmentation architectures** are typically based upon “mask classification.” They predict a set of binary masks each associated with a single class label. The pioneer work, Mask R-CNN [5], generates masks from detected bounding boxes. Follow-up methods either focus on detecting more precise bounding boxes [129, 133], or finding new ways to generate a dynamic number of masks, *e.g.*, using dynamic kernels [42, 112, 113] or clustering algorithms [6, 35]. Although the performance has been advanced in each task, these specialized innovations lack the flexibility to generalize from one to the other, leading to duplicated research effort. For instance, although multiple approaches have been proposed for building feature pyramid representations [25], as we show in our experiments, BiFPN [134] performs better for instance segmentation while FaPN [131] performs better for semantic segmentation.

Panoptic segmentation has been proposed to unify both semantic and instance segmentation tasks [4]. Architectures for panoptic segmentation either combine the best of specialized semantic and instance segmentation architectures into a single framework [6, 14, 15, 135] or design novel objectives that equally treat semantic regions and instance objects [91, 97]. Despite those new architectures, researchers continue to develop specialized architectures for different image segmentation tasks [110, 132]. We find that panoptic architectures usually only report performance on a single panoptic segmentation task [97], which does not guarantee good performance on other tasks (Figure 2.1). For example, panoptic segmentation does not measure architectures’ abilities to rank predictions as instance segmentations. Thus, we refrain from referring to architectures that are only evaluated for panoptic segmentation as universal architectures. Instead, here, we evaluate our Mask2Former on all studied tasks to guarantee generalizability.

Universal architectures have emerged with DETR [91] and show that mask classification architectures with an end-to-end set prediction objective are general enough for any image segmentation task. MaskFormer [8] shows that mask classification based on DETR not only performs well on panoptic segmentation but also achieves state-of-the-art on semantic segmentation. K-Net [130] further extends set prediction to instance segmentation. Unfortunately, these architectures fail to replace specialized models as their

performance on particular tasks or datasets is still worse than the best specialized architecture (*e.g.*, MaskFormer [8] cannot segment instances well). To our knowledge, Mask2Former is the first architecture that outperforms state-of-the-art specialized architectures on all considered tasks and datasets.

4.3 Masked-attention Mask Transformer

We now present Mask2Former. We first review a meta architecture for mask classification that Mask2Former is built upon. Then, we introduce our new Transformer decoder with *masked attention* which is the key to better convergence and results. Lastly, we propose training improvements that make Mask2Former efficient and accessible.

4.3.1 Mask Classification Preliminaries

Mask classification architectures group pixels into N segments by predicting N binary masks, along with N corresponding category labels. Mask classification is sufficiently general to address any segmentation task by assigning different semantics, *e.g.*, categories or instances, to different segments. However, the challenge is to find good representations for each segment. For example, Mask R-CNN [5] uses bounding boxes as the representation which limits its application to semantic segmentation. Inspired by DETR [91], each segment in an image can be represented as a C -dimensional feature vector (“object query”) and can be processed by a Transformer decoder, trained with a set prediction objective. A simple meta architecture would consist of three components. A *backbone* that extracts low-resolution features from an image. A *pixel decoder* that gradually upsamples low-resolution features from the output of the backbone to generate high-resolution per-pixel embeddings. And finally a *Transformer decoder* that operates on image features to process object queries. The final binary mask predictions are decoded from per-pixel embeddings with object queries. One successful instantiation of such a meta architecture is MaskFormer [8], and we refer readers to [8] for more details.

4.3.2 Transformer Decoder with Masked Attention

Mask2Former adopts the aforementioned meta architecture, with our proposed Transformer decoder (Figure 4.2 right) replacing the standard one. The key components of our Transformer decoder include a *masked attention* operator, which extracts localized features by constraining cross-attention to within the foreground region of the predicted mask for each query, instead of attending to the full feature map. To handle small objects, we propose an efficient multi-scale strategy to utilize high-resolution features. It feeds successive feature maps from the pixel decoder’s feature pyramid into successive Transformer decoder layers in a round robin fashion. Finally, we incorporate optimization improvements that boost model performance without introducing additional computation. We now discuss these improvements in detail.

4.3.2.1 Masked attention

Context features have been shown to be important for image segmentation [7, 26, 90]. However, recent studies [136, 137] suggest that the slow convergence of Transformer-based models is due to global context in the cross-attention layer, as it takes many training epochs for cross-attention to learn to attend to localized object regions [136]. We hypothesize that local features are enough to update query features and context information can be gathered through self-attention. For this we propose *masked attention*, a variant of cross-attention that only attends within the foreground region of the predicted mask for each query.

Standard cross-attention (with residual path) computes

$$\mathbf{X}_l = \text{softmax}(\mathbf{Q}_l \mathbf{K}_l^T) \mathbf{V}_l + \mathbf{X}_{l-1}. \quad (4.1)$$

Here, l is the layer index, $\mathbf{X}_l \in \mathbb{R}^{N \times C}$ refers to N C -dimensional query features at the l^{th} layer and $\mathbf{Q}_l = f_Q(\mathbf{X}_{l-1}) \in \mathbb{R}^{N \times C}$. $\mathbf{X}_0$ denotes input query features to the Transformer decoder. $\mathbf{K}_l, \mathbf{V}_l \in \mathbb{R}^{H_l W_l \times C}$ are the image features under transformation $f_K(\cdot)$ and $f_V(\cdot)$ respectively, and H_l and W_l are the spatial resolution of image features that we will introduce next in Section 4.3.2.2. f_Q , f_K and f_V are linear transformations.

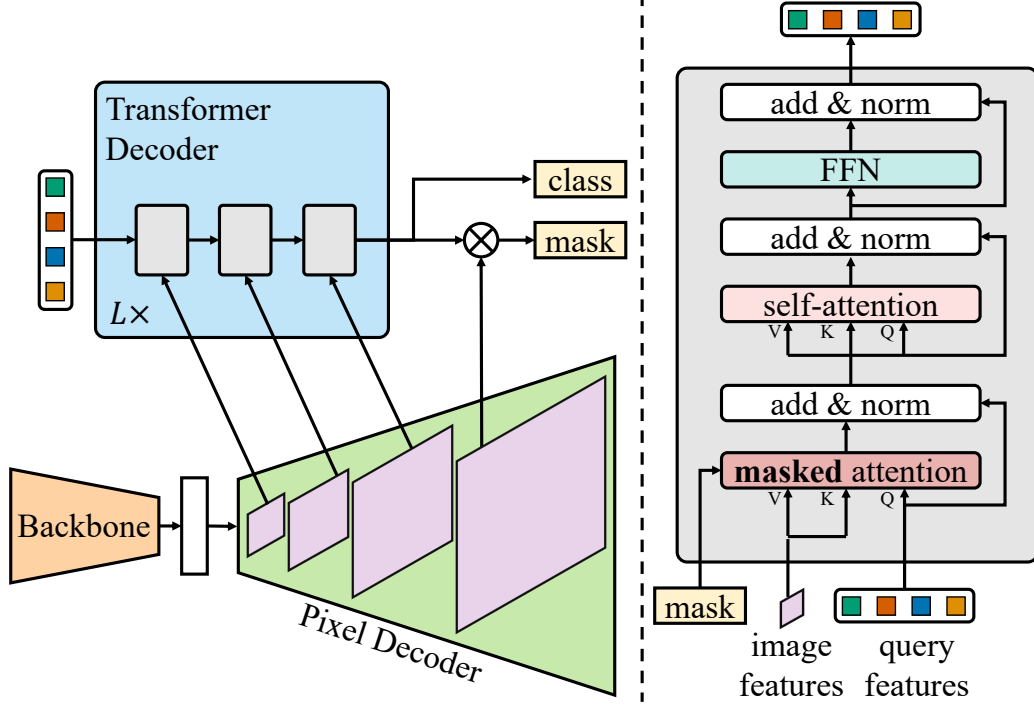


Figure 4.2: **Mask2Former overview.** Mask2Former adopts the same meta architecture as MaskFormer [8] with a backbone, a pixel decoder and a Transformer decoder. We propose a new Transformer decoder with *masked attention* instead of the standard cross-attention (Section 4.3.2.1). To deal with small objects, we propose an efficient way of utilizing high-resolution features from a pixel decoder by feeding one scale of the multi-scale feature to one Transformer decoder layer at a time (Section 4.3.2.2). In addition, we switch the order of self and cross-attention (*i.e.*, our masked attention), make query features learnable, and remove dropout to make computation more effective (Section 4.3.2.3). Note that positional embeddings and predictions from intermediate Transformer decoder layers are omitted in this figure for readability.

Our masked attention modulates the attention matrix via

$$\mathbf{X}_l = \text{softmax}(\mathcal{M}_{l-1} + \mathbf{Q}_l \mathbf{K}_l^T) \mathbf{V}_l + \mathbf{X}_{l-1}. \quad (4.2)$$

Moreover, the attention mask $\mathcal{M}_{l-1}$ at feature location (x, y) is

$$\mathcal{M}_{l-1}(x, y) = \begin{cases} 0 & \text{if } \mathbf{M}_{l-1}(x, y) = 1 \\ -\infty & \text{otherwise} \end{cases}. \quad (4.3)$$

Here, $\mathbf{M}_{l-1} \in \{0, 1\}^{N \times H_l W_l}$ is the binarized output (thresholded at 0.5) of the

resized mask prediction of the previous $(l - 1)$ -th Transformer decoder layer. It is resized to the same resolution of $\mathbf{K}_l$. $\mathbf{M}_0$ is the binary mask prediction obtained from $\mathbf{X}_0$, *i.e.*, before feeding query features into the Transformer decoder.

4.3.2.2 High-resolution features

High-resolution features improve model performance, especially for small objects [91]. However, this is computationally demanding. Thus, we propose an efficient multi-scale strategy to introduce high-resolution features while controlling the increase in computation. Instead of always using the high-resolution feature map, we utilize a feature pyramid which consists of both low- and high-resolution features and feed one resolution of the multi-scale feature to one Transformer decoder layer at a time.

Specifically, we use the feature pyramid produced by the *pixel decoder* with resolution $1/32$, $1/16$ and $1/8$ of the original image. For each resolution, we add both a sinusoidal positional embedding $e_{\text{pos}} \in \mathbb{R}^{H_l W_l \times C}$, following [91], and a learnable scale-level embedding $e_{\text{lvl}} \in \mathbb{R}^{1 \times C}$, following [138]. We use those, from lowest-resolution to highest-resolution for the corresponding Transformer decoder layer as shown in Figure 4.2 left. We repeat this 3-layer Transformer decoder L times. Our final Transformer decoder hence has $3L$ layers. More specifically, the first three layers receive a feature map of resolution $H_1 = H/32$, $H_2 = H/16$, $H_3 = H/8$ and $W_1 = W/32$, $W_2 = W/16$, $W_3 = W/8$, where H and W are the original image resolution. This pattern is repeated in a round robin fashion for all following layers.

4.3.2.3 Optimization improvements

A standard Transformer decoder layer [93] consists of three modules to process query features in the following order: a self-attention module, a cross-attention and a feed-forward network (FFN). Moreover, query features ($\mathbf{X}_0$) are *zero initialized* before being fed into the Transformer decoder and are associated with *learnable* positional embeddings. Furthermore, dropout is applied to both residual connections and attention maps.

To optimize Transformer decoder design, we make the following three im-

provements. First, we switch the order of self- and cross-attention (our new “masked attention”) to make computation more effective: query features to the first self-attention layer are not dependent on image features yet, thus applying self-attention does not generate any meaningful features. Second, we make query features ($\mathbf{X}_0$) learnable as well (we still keep the learnable query positional embeddings), and learnable query features are directly supervised before being used in the Transformer decoder to predict masks ($\mathbf{M}_0$). We find these learnable query features function like a region proposal network [84] and have the ability to generate mask proposals. Finally, we find dropout is not necessary and usually decreases performance. We thus completely remove dropout in our decoder.

4.3.3 Improving Training Efficiency

One limitation of training universal architectures is the large memory consumption due to high-resolution mask prediction, making them less accessible than the more memory-friendly specialized architectures [5, 129]. For example, MaskFormer [8] can only fit a single image in a GPU with 32G memory. Motivated by PointRend [139] and Implicit PointRend [140], which show a segmentation model can be trained with its mask loss calculated on K randomly sampled points instead of the whole mask, we calculate the mask loss with sampled points in both the matching and the final loss calculation. More specifically, in the *matching loss* that constructs the cost matrix for bipartite matching, we *uniformly* sample the same set of K points for all prediction and ground truth masks. In the *final loss* between predictions and their matched ground truths, we sample different sets of K points for different pairs of prediction and ground truth using *importance sampling* [139]. We set $K = 12544$, *i.e.*, 112×112 points. This new training strategy effectively reduces training memory by $3\times$, from 18GB to 6GB per image, making Mask2Former more accessible to users with limited computational resources.

4.4 Experiments

We demonstrate Mask2Former is an effective architecture for universal image segmentation through comparisons with specialized state-of-the-art architectures on standard benchmarks. We evaluate our proposed design decisions through ablations on all three tasks. Finally we show Mask2Former generalizes beyond the standard benchmarks, obtaining state-of-the-art results on four datasets.

Datasets. We study Mask2Former using four widely used image segmentation datasets that support semantic, instance and panoptic segmentation: COCO [11] (80 “things” and 53 “stuff” categories), ADE20K [94] (100 “things” and 50 “stuff” categories), Cityscapes [12] (8 “things” and 11 “stuff” categories) and Mapillary Vistas [13] (37 “things” and 28 “stuff” categories). Panoptic and semantic segmentation tasks are evaluated on the union of “things” and “stuff” categories while instance segmentation is only evaluated on the “things” categories.

Evaluation metrics. For *panoptic segmentation*, we use the standard **PQ** (panoptic quality) metric [4]. We further report $\mathbf{AP}_{\text{pan}}^{\text{Th}}$, which is the AP evaluated on the “thing” categories using instance segmentation annotations, and $\mathbf{mIoU}_{\text{pan}}$, which is the mIoU for semantic segmentation by merging instance masks from the same category, of the same model trained *only* with panoptic segmentation annotations. For *instance segmentation*, we use the standard **AP** (average precision) metric [11]. For *semantic segmentation*, we use **mIoU** (mean Intersection-over-Union) [2].

4.4.1 Implementation Details

We adopt settings from [8] with the following differences:

Pixel decoder. Mask2Former is compatible with any existing pixel decoder module. In MaskFormer [8], FPN [25] is chosen as the default for its simplicity. Since our goal is to demonstrate strong performance across different segmentation tasks, we use the more advanced multi-scale deformable attention Transformer (MSDeformAttn) [138] as our default pixel decoder. Specifically, we use 6 MSDeformAttn layers applied to feature maps with resolution 1/8, 1/16 and 1/32, and use a simple upsampling layer with lateral

Table 4.1: **Panoptic segmentation on COCO panoptic val2017 with 133 categories.** Mask2Former consistently outperforms MaskFormer [8] by a large margin with different backbones on all metrics. Our best model outperforms prior state-of-the-art MaskFormer by 5.1 PQ and K-Net [130] by 3.2 PQ. Backbones pre-trained on ImageNet-22K are marked with [†].

method	backbone	query type	epochs	PQ	PQ Th	PQ St	AP Th _{pan}	mIoU _{pan}	#params.	FLOPs	fps
DETR [91]	R50	100 queries	500+25	43.4	48.2	36.3	31.1	-	-	-	-
MaskFormer [8]	R50	100 queries	300	46.5	51.0	39.8	33.0	57.8	45M	181G	17.6
Mask2Former (ours)	R50	100 queries	50	51.9	57.7	43.0	41.7	61.7	44M	226G	8.6
DETR [91]	R101	100 queries	500+25	45.1	50.5	37.0	33.0	-	-	-	-
MaskFormer [8]	R101	100 queries	300	47.6	52.5	40.3	34.1	59.3	64M	248G	14.0
Mask2Former (ours)	R101	100 queries	50	52.6	58.5	43.7	42.6	62.4	63M	293G	7.2
Max-DeepLab [97]	Max-L	128 queries	216	51.1	57.0	42.2	-	-	451M	3692G	-
MaskFormer [8]	Swin-L [†]	100 queries	300	52.7	58.5	44.0	40.1	64.8	212M	792G	5.2
K-Net [130]	Swin-L [†]	100 queries	36	54.6	60.2	46.0	-	-	-	-	-
Mask2Former (ours)	Swin-L [†]	<i>200 queries</i>	<i>100</i>	57.8	64.2	48.1	48.6	67.4	216M	868G	4.0

Table 4.2: **Instance segmentation on COCO val2017 with 80 categories.** Mask2Former outperforms strong Mask R-CNN [5] baselines with $8\times$ fewer epochs under both AP and AP_{boundary} [10]. Our best model is also competitive with the state-of-the-art specialized instance segmentation model on COCO and has higher boundary quality. For a fair comparison, we only consider single-scale inference and models trained using only COCO train2017 set data. Backbones pre-trained on ImageNet-22K are marked with [†].

method	backbone	query type	epochs	AP	AP _S	AP _M	AP _L	AP _{boundary}	#params.	FLOPs	fps
MaskFormer [8]	R50	100 queries	300	34.0	16.4	37.8	54.2	23.0	45M	181G	19.2
Mask R-CNN [5]	R50	dense anchors	36	37.2	18.6	39.5	53.3	23.1	44M	201G	15.2
Mask R-CNN [5, 141, 142]	R50	dense anchors	400	42.5	23.8	45.0	60.0	28.0	46M	358G	10.3
Mask2Former (ours)	R50	100 queries	50	43.7	23.4	47.2	64.8	30.6	44M	226G	9.7
Mask R-CNN [5]	R101	dense anchors	36	38.6	19.5	41.3	55.3	24.5	63M	266G	10.8
Mask R-CNN [5, 141, 142]	R101	dense anchors	400	43.7	24.6	46.4	61.8	29.1	65M	423G	8.6
Mask2Former (ours)	R101	100 queries	50	44.2	23.8	47.7	66.7	31.1	63M	293G	7.8
QueryInst [132]	Swin-L [†]	300 queries	50	48.9	30.8	52.6	68.3	33.5	-	-	3.3
Swin-HTC++ [96, 129]	Swin-L [†]	dense anchors	72	49.5	31.0	52.4	67.2	34.1	284M	1470G	-
Mask2Former (ours)	Swin-L [†]	200 queries	100	50.1	29.9	53.9	72.1	36.2	216M	868G	4.0

connection on the final 1/8 feature map to generate the feature map of resolution 1/4 as the per-pixel embedding. In our ablation study, we show that this pixel decoder provides best results across different segmentation tasks.

Transformer decoder. We use our Transformer decoder proposed in Section 4.3.2 with $L = 3$ (*i.e.*, 9 layers total) and 100 queries by default. An auxiliary loss is added to every intermediate Transformer decoder layer and to the learnable query features before the Transformer decoder.

Loss weights. We use the binary cross-entropy loss (instead of focal loss [114] in [8]) and the dice loss [115] for our mask loss: $\mathcal{L}_{\text{mask}} = \lambda_{\text{ce}}\mathcal{L}_{\text{ce}} + \lambda_{\text{dice}}\mathcal{L}_{\text{dice}}$. We set $\lambda_{\text{ce}} = 5.0$ and $\lambda_{\text{dice}} = 5.0$. The final loss is a combination of mask loss and classification loss, $\mathcal{L}_{\text{mask}} + \lambda_{\text{cls}}\mathcal{L}_{\text{cls}}$, and we set $\lambda_{\text{cls}} = 2.0$ for predictions matched with a ground truth and 0.1 for the “no object,” *i.e.*, predictions that have not been matched with any ground truth.

Post-processing. We use the exact same post-processing as [8] to acquire the expected output format for panoptic and semantic segmentation from pairs of binary masks and class predictions. Instance segmentation requires additional confidence scores for each prediction. We multiply class confidence and mask confidence (*i.e.*, averaged foreground per-pixel binary mask probability) and use it as the final confidence score.

4.4.2 Training Settings

Panoptic and instance segmentation. We use Detectron2 [78] and follow the updated Mask R-CNN [5] baseline settings¹ for the COCO dataset. More specifically, we use AdamW [120] optimizer and the step learning rate schedule. We use an initial learning rate of 0.0001 and a weight decay of 0.05 for all backbones. A learning rate multiplier of 0.1 is applied to the backbone and we decay the learning rate at 0.9 and 0.95 fractions of the total number of training steps by a factor of 10. If not stated otherwise, we train our models for 50 epochs with a batch size of 16. For data augmentation, we use the large-scale jittering (LSJ) augmentation [141, 142] with a random scale sampled from range 0.1 to 2.0 followed by a fixed size crop to 1024×1024 . We use the standard Mask R-CNN inference setting where we resize an image with shorter side to 800 and longer side up-to 1333. We

¹https://github.com/facebookresearch/detectron2/blob/main/MODEL_ZOO.md#new-baselines-using-large-scale-jitter-and-longer-training-schedule

also report FLOPs and fps. FLOPs are averaged over 100 validation images (COCO images have varying sizes). Frames-per-second (fps) is measured on a V100 GPU with a batch size of 1 by taking the average runtime on the entire validation set including post-processing time.

Semantic segmentation. We follow the same settings as [8] to train our models, except: (1) a learning rate multiplier of 0.1 is applied to *both* CNN and Transformer backbones instead of only applying it to CNN backbones in [8], (2) both ResNet and Swin backbones use an initial learning rate of 0.0001 and a weight decay of 0.05, instead of using different learning rates in [8].

4.4.3 Main Results

Panoptic segmentation. We compare Mask2Former with state-of-the-art models for panoptic segmentation on the COCO panoptic [4] dataset in Table 4.1. Mask2Former consistently outperforms MaskFormer by more than 5 PQ across different backbones while converging $6\times$ faster. With Swin-L backbone, our Mask2Former sets a new state-of-the-art of 57.8 PQ, outperforming existing state-of-the-art [8] by 5.1 PQ and concurrent work, K-Net [130], by 3.2 PQ. Mask2Former even outperforms the best ensemble models with extra training data in the COCO challenge (see Section 4.6.1.1 for test set results).

Beyond the PQ metric, our Mask2Former also achieves higher performance on two other metrics compared to DETR [91] and MaskFormer: $AP_{\text{pan}}^{\text{Th}}$, which is the AP evaluated on the 80 “thing” categories using *instance segmentation annotation*, and $mIoU_{\text{pan}}$, which is the mIoU evaluated on the 133 categories for semantic segmentation converted from panoptic segmentation annotation. This shows Mask2Former’s universality: trained *only* with panoptic segmentation annotations, it can be used for instance and semantic segmentation.

Instance segmentation. We compare Mask2Former with state-of-the-art models on the COCO [11] dataset in Table 4.2. With ResNet [77] backbone, Mask2Former outperforms a strong Mask R-CNN [5] baseline using large-scale jittering (LSJ) augmentation [141, 142] while requiring $8\times$ fewer training iterations. With Swin-L backbone, Mask2Former outperforms the state-of-the-art HTC++ [129]. Although we only observe +0.6 AP improvement over HTC++, the Boundary AP [10] improves by 2.1, suggesting that our

Table 4.3: **Semantic segmentation on ADE20K val with 150 categories.** Mask2Former consistently outperforms MaskFormer [8] by a large margin with different backbones (all Mask2Former models use MSDeformAttn [138] as pixel decoder, except Swin-L-FaPN uses FaPN [131]). Our best model outperforms the best specialized model, BEiT [143]. We report both single-scale (s.s.) and multi-scale (m.s.) inference results. Backbones pre-trained on ImageNet-22K are marked with [†].

method	backbone	crop size	mIoU (s.s.)	mIoU (m.s.)
MaskFormer [8]	R50	512	44.5	46.7
Mask2Former (ours)	R50	512	47.2	49.2
Swin-UpperNet [96, 123]	Swin-T	512	-	46.1
MaskFormer [8]	Swin-T	512	46.7	48.8
Mask2Former (ours)	Swin-T	512	47.7	49.6
MaskFormer [8]	Swin-L [†]	640	54.1	55.6
FaPN-MaskFormer [8, 131]	Swin-L-FaPN [†]	640	55.2	56.7
BEiT-UpperNet [123, 143]	BEiT-L [†]	640	-	57.0
Mask2Former (ours)	Swin-L [†]	640	56.1	57.3
	Swin-L-FaPN [†]	640	56.4	57.7

predictions have a better boundary quality thanks to the high-resolution mask predictions. Note that for a fair comparison, we only consider single-scale inference and models trained with only COCO train2017 set data.

With a ResNet-50 backbone Mask2Former improves over MaskFormer on small objects by 7.0 AP^S, while overall the highest gains come from large objects (+10.6 AP^L). The performance on AP^S still lags behind other state-of-the-art models. Hence there still remains room for improvement on small objects, *e.g.*, by using dilated backbones like in DETR [91], which we leave for future work.

Semantic segmentation. We compare Mask2Former with state-of-the-art models for semantic segmentation on the ADE20K [94] dataset in Table 4.3. Mask2Former outperforms MaskFormer [8] across different backbones, suggesting that the proposed improvements even boost semantic segmentation results where [8] was already state-of-the-art. With Swin-L as backbone and FaPN [131] as pixel decoder, Mask2Former sets a new state-of-the-art of 57.7 mIoU. We also report the test set results in Section 4.6.1.3.

Table 4.4: **Mask2Former ablations.** We perform ablations on three tasks: instance (AP on COCO val2017), panoptic (PQ on COCO panoptic val2017) and semantic (mIoU on ADE20K val) segmentation. FLOPs are measured on COCO instance segmentation.

(a) Masked attention and high-resolution features (from efficient multi-scale strategy) lead to the most gains. More detailed ablations are in Table 4.4c and Table 4.4d. We remove one component at a time.

Mask2Former (ours)	AP	PQ	mIoU	FLOPs
– masked attention	43.7	51.9	47.2	226G
– high-resolution features	37.8 (-5.9)	47.1 (-4.8)	45.5 (-1.7)	213G
	41.5 (-2.2)	50.2 (-1.7)	46.1 (-1.1)	218G

(c) **Masked attention.** Our masked attention performs better than other variants of cross-attention across all tasks.

	AP	PQ	mIoU	FLOPs
cross-attention	37.8	47.1	45.5	213G
SMCA [137]	37.9	47.2	46.6	213G
mask pooling [130]	43.1	51.5	46.0	217G
masked attention	43.7	51.9	47.2	226G

(d) **Feature resolution.** High-resolution features (single scale 1/8) are important. Our efficient multi-scale (efficient m.s.) strategy effectively reduces the FLOPs.

	AP	PQ	mIoU	FLOPs
single scale (1/32)	41.5	50.2	46.1	218G
single scale (1/16)	43.0	51.5	46.5	222G
single scale (1/8)	44.0	51.8	47.4	239G
naïve m.s. (3 scales)	44.0	51.9	46.3	247G
efficient m.s. (3 scales)	43.7	51.9	47.2	226G

(b) Optimization improvements increase the performance without introducing extra compute. Following DETR [91], query features are zero-initialized when not learnable. We remove one component at a time.

	AP	PQ	mIoU	FLOPs
Mask2Former (ours)	43.7	51.9	47.2	226G
– learnable query features	42.9 (-0.8)	51.2 (-0.7)	45.4 (-1.8)	226G
– cross-attention first	43.2 (-0.5)	51.6 (-0.3)	46.3 (-0.9)	226G
– remove dropout	43.0 (-0.7)	51.3 (-0.6)	47.2 (-0.0)	226G
– all 3 components above	42.3 (-1.4)	50.8 (-1.1)	46.3 (-0.9)	226G

(e) **Pixel decoder.** MSDeformAttn [138] consistently performs the best across all tasks.

	AP	PQ	mIoU	FLOPs
FPN [25]	41.5	50.7	45.6	195G
Semantic FPN [15]	42.1	51.2	46.2	258G
FaPN [131]	42.4	51.8	46.8	-
BiFPN [134]	43.5	51.8	45.6	204G
MSDeformAttn [138]	43.7	51.9	47.2	226G

4.4.4 Ablation Studies

We now analyze Mask2Former through a series of ablation studies using a ResNet-50 backbone [77]. To test the generality of the proposed components for universal image segmentation, *all ablations are performed on three tasks.*

Transformer decoder. We validate the importance of each component by removing them one at a time. As shown in Table 4.4a, masked attention leads to the biggest improvement across all tasks. The improvement is larger for instance and panoptic segmentation than for semantic segmentation. Moreover, using high-resolution features from the efficient multi-scale strategy is also important. Table 4.4b shows additional optimization improvements further improve the performance without extra computation.

Masked attention. Concurrent work has proposed other variants of cross-attention [137, 144] that aim to improve the convergence and performance of DETR [91] for object detection. Most recently, K-Net [130] replaced cross-attention with a mask pooling operation that averages features within mask regions. We validate the importance of our masked attention in Table 4.4c. While existing cross-attention variants may improve on a specific task, our masked attention performs the best on all three tasks.

Feature resolution. Table 4.4d shows that Mask2Former benefits from using high-resolution features (*e.g.*, a single scale of 1/8) in the Transformer decoder. However, this introduces additional computation. Our efficient multi-scale (efficient m.s.) strategy effectively reduces the FLOPs without affecting the performance. Note that naively concatenating multi-scale features as input to every Transformer decoder layer (naïve m.s.) does not yield additional gains.

Pixel decoder. As shown in Table 4.4e, Mask2Former is compatible with any existing pixel decoder. However, we observe different pixel decoders specialize in different tasks: while BiFPN [134] performs better on instance-level segmentation, FaPN [131] works better for semantic segmentation. Among all studied pixel decoders, the MSDeformAttn [138] consistently performs the best across all tasks and thus is selected as our default. This set of ablations also suggests that designing a module like a pixel decoder for a specific task does not guarantee generalization across segmentation tasks. Mask2Former, as a universal model, could serve as a testbed for a generalizable module design.

Table 4.5: **Calculating loss with points *vs.* masks.** Training with point loss reduces training memory without influencing the performance. Matching with point loss further improves performance.

matching loss	training loss	AP (COCO)	PQ (COCO)	mIoU (ADE20K)	memory (COCO)
mask	mask	41.0	50.3	45.9	18G
	point	41.0	50.8	45.9	6G
point (ours)	mask	43.1	51.4	47.3	18G
	point (ours)	43.7	51.9	47.2	6G

Calculating loss with points *vs.* masks. As discussed in Section 4.3.3, we study the effect of calculating the loss based on either mask or sampled points on performance and training memory in Table 4.5. Calculating the final training loss with sampled points reduces training memory by $3\times$ without affecting the performance. Additionally, calculating the matching loss with sampled points improves performance across all three tasks.

Learnable queries as region proposals. Region proposals [103, 104], either in the form of boxes or masks, are regions that are likely to be “objects.” With learnable queries being supervised by the mask loss, predictions from learnable queries can serve as mask proposals. In Figure 4.3 top, we visualize mask predictions of selected learnable queries *before* feeding them into the Transformer decoder (the proposal generation process is shown in Figure 4.3 bottom right). In Figure 4.3 bottom left, we further perform a quantitative analysis on the quality of these proposals by calculating the class-agnostic average recall with 100 predictions (AR@100) on COCO val2017. We find these learnable queries already achieve good AR@100 compared to the final predictions of Mask2Former after the Transformer decoder layers, *i.e.*, layer 9, and AR@100 consistently improves with more decoder layers.

4.4.5 Generalization to Other Datasets

To show our Mask2Former can generalize beyond the COCO dataset, we further perform experiments on other popular image segmentation datasets. In Table 4.6, we show results on Cityscapes [12]. Please see Section 4.6.2 for detailed training settings on each dataset as well as more results on ADE20K [94] and Mapillary Vistas [13].

We observe that our Mask2Former is competitive with state-of-the-art

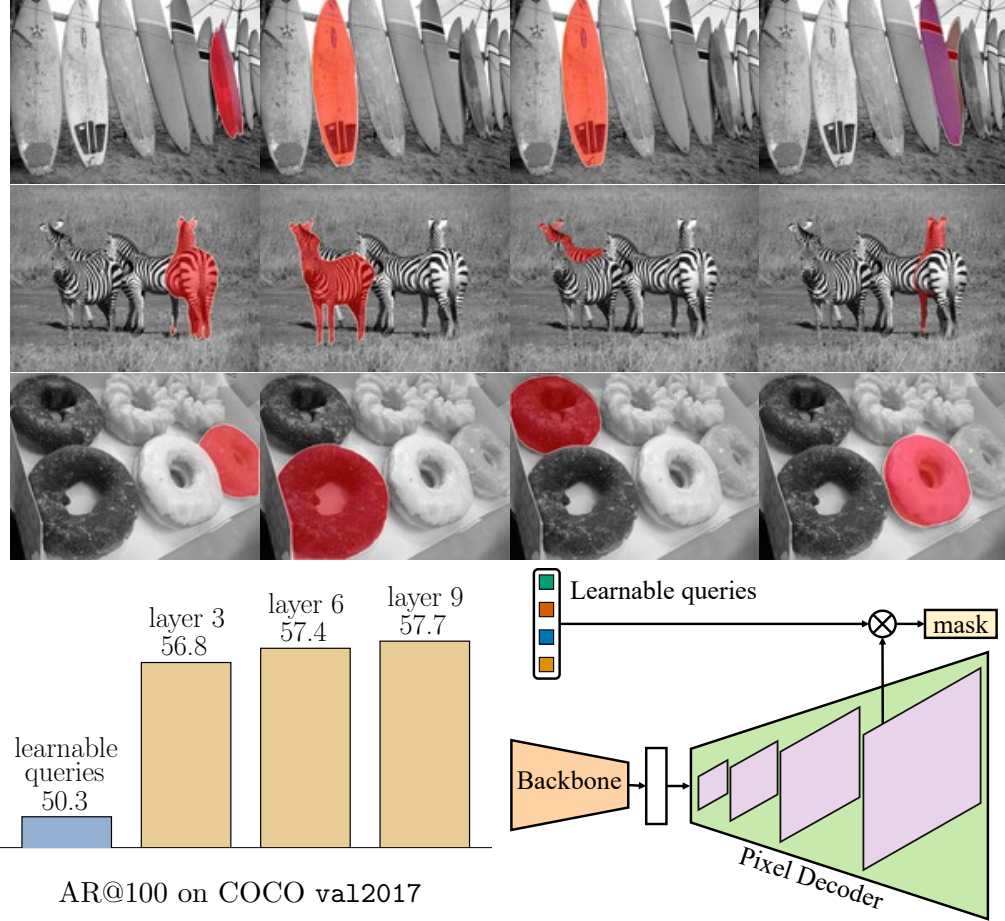


Figure 4.3: **Learnable queries as “region proposals”**. *Top*: We visualize mask predictions of four selected learnable queries *before* feeding them into the Transformer decoder (using R50 backbone). *Bottom left*: We calculate the class-agnostic average recall with 100 proposals (AR@100) and observe that these learnable queries provide good proposals compared to the final predictions of Mask2Former after the Transformer decoder layers (layer 9). *Bottom right*: Illustration of proposal generation process.

methods on these datasets as well. It suggests Mask2Former can serve as a universal image segmentation model and results generalize across datasets.

4.4.6 Limitations

Our ultimate goal is to train a *single* model for *all* image segmentation tasks. In Table 4.7, we find Mask2Former trained on panoptic segmentation only performs slightly worse than the exact same model trained with the corresponding annotations for instance and semantic segmentation tasks across

Table 4.6: **Cityscapes val**. Mask2Former is competitive with specialized models on Cityscapes. Panoptic segmentation models use single-scale inference by default, multi-scale numbers are marked with *. For semantic segmentation, we report both single-scale (s.s.) and multi-scale (m.s.) inference results. Backbones pre-trained on ImageNet-22K are marked with †.

method	backbone	panoptic model			semantic model	
		PQ	AP _{pan} Th	mIoU _{pan}	mIoU (s.s.)	mIoU (m.s.)
Panoptic FCN [135]	Swin-L [†]	65.9	-	-	-	-
Panoptic-DeepLab [6]	SWideRNet [145]	66.4	40.1	82.2	-	-
Panoptic-DeepLab [6]	SWideRNet [145]	67.5*	43.9*	82.9*	-	-
SETR [109]	ViT-L [†] [111]	-	-	-	-	82.2
SegFormer [146]	MiT-B5 [146]	-	-	-	-	84.0
Mask2Former (ours)	R50	62.1	37.3	77.5	79.4	82.2
	Swin-B [†]	66.1	42.8	82.7	83.3	84.5
	Swin-L [†]	66.6	43.6	82.9	83.3	84.3

Table 4.7: **Limitations of Mask2Former**. Although a single Mask2Former can address any segmentation task, we still need to train it on different tasks. Across three datasets we find Mask2Former trained with panoptic annotations performs slightly worse than the exact same model trained specifically for instance and semantic segmentation tasks with the corresponding data.

	PQ	AP	mIoU	PQ	AP	mIoU	PQ	AP	mIoU
panoptic	51.9	41.7	61.7	39.7	26.5	46.1	62.1	37.3	77.5
instance	-	43.7	-	-	26.4	-	-	37.4	-
semantic	-	-	61.5	-	-	47.2	-	-	79.4
(a) COCO			(b) ADE20K			(c) Cityscapes			

three datasets. This suggests that even though Mask2Former can generalize to different tasks, it still needs to be trained for those specific tasks. In the future, we hope to develop a model that can be trained only once for multiple tasks and even for multiple datasets.

Furthermore, as seen in Tables 4.2 and 4.4d, even though it improves over baselines, Mask2Former struggles with segmenting small objects and is unable to fully leverage multi-scale features. We believe better utilization of the feature pyramid and designing losses for small objects are critical.

4.5 Conclusion

We present Mask2Former for universal image segmentation. Built upon a simple meta framework [8] with a new Transformer decoder using the proposed masked attention, Mask2Former obtains top results in all three major image segmentation tasks (panoptic, instance and semantic) on four popular datasets, outperforming even the best specialized models designed for each benchmark while remaining easy to train. Mask2Former saves $3\times$ research effort compared to designing specialized models for each task, and it is accessible to users with limited computational resources. We hope this study attracts more interest in designing universal models in the future.

4.6 Supplementary Materials

We first provide more results for Mask2Former with different backbones as well as test-set performance on standard benchmarks (Section 4.6.1): We use COCO panoptic [4] for panoptic, COCO [11] for instance, and ADE20K [94] for semantic segmentation. Then, we provide more detailed results on additional datasets (Section 4.6.2). Finally, we provide additional ablation studies (Section 4.6.3) and visualization of Mask2Former predictions for all three segmentation tasks (Section 4.6.4).

4.6.1 Additional Results

Here, we provide more results of Mask2Former with different backbones on COCO panoptic [4] for panoptic segmentation, COCO [11] for instance segmentation and ADE20K [94] for semantic segmentation. More specifically, for each benchmark, we evaluate Mask2Former with ResNet [77] with 50 and 101 layers, as well as Swin [96] Tiny, Small, Base and Large variants as backbones. We use ImageNet [121] pre-trained checkpoints to initialize backbones.

Table 4.8: **Panoptic segmentation on COCO panoptic val2017 with 133 categories.** Mask2Former outperforms *all* existing panoptic segmentation models by a large margin with different backbones on all metrics. Our best model sets a new state-of-the-art of 57.8 PQ. Besides PQ for panoptic segmentation, we also report $AP_{\text{pan}}^{\text{Th}}$ (the AP evaluated on the 80 “thing” categories using *instance segmentation annotation*) and $mIoU_{\text{pan}}$ (the $mIoU$ evaluated on the 133 categories for semantic segmentation converted from panoptic segmentation annotation) of the same model trained for panoptic segmentation (**note: we train all our models with panoptic segmentation annotation only**). Backbones pre-trained on ImageNet-22K are marked with $\dagger$.

method	backbone	search space	epochs	PQ	PQ Th	PQ St	$AP_{\text{pan}}^{\text{Th}}$	$mIoU_{\text{pan}}$	#params.	FLOPs
CNN backbones	DETR [91]	R50	100 queries	500+25	43.4	48.2	36.3	31.1	-	-
		R101	100 queries	500+25	45.1	50.5	37.0	33.0	-	-
	K-Net [130]	R50	100 queries	36	47.1	51.7	40.3	-	-	-
	Panoptic SegFormer [147]	R50	400 queries	50	50.0	56.1	40.8	-	47M	246G
		R50	100 queries	300	46.5	51.0	39.8	33.0	57.8	45M
	MaskFormer [8]	R101	100 queries	300	47.6	52.5	40.3	34.1	59.3	64M
		R50	100 queries	50	51.9	57.7	43.0	41.7	61.7	44M
	Mask2Former (ours)	R101	100 queries	50	52.6	58.5	43.7	42.6	62.4	63M
										293G
Transformer backbones	Max-DeepLab [97]	Max-S	128 queries	216	48.4	53.0	41.5	-	-	62M
		Max-L	128 queries	216	51.1	57.0	42.2	-	-	451M
	Panoptic SegFormer [147]	PVTv2-B5 [148]	400 queries	50	54.1	60.4	44.6	-	-	101M
	K-Net [130]	Swin-L $\dagger$	100 queries	36	54.6	60.2	46.0	-	-	-
		Swin-T	100 queries	300	47.7	51.7	41.7	33.6	60.4	42M
		Swin-S	100 queries	300	49.7	54.4	42.6	36.1	61.3	63M
		Swin-B	100 queries	300	51.1	56.3	43.2	37.8	62.6	102M
	MaskFormer [8]	Swin-B $\dagger$	100 queries	300	51.8	56.9	44.1	38.5	63.6	102M
		Swin-L $\dagger$	100 queries	300	52.7	58.5	44.0	40.1	64.8	212M
		Swin-T	100 queries	50	53.2	59.3	44.0	43.3	63.2	47M
		Swin-S	100 queries	50	54.6	60.6	45.7	44.7	64.2	69M
		Swin-B	100 queries	50	55.1	61.0	46.1	45.2	65.1	107M
	Mask2Former (ours)	Swin-B $\dagger$	100 queries	50	56.4	62.4	47.3	46.3	67.1	107M
		Swin-L $\dagger$	200 queries	100	57.8	64.2	48.1	48.6	67.4	216M
										868G

Table 4.9: **Panoptic segmentation on COCO panoptic test-dev with 133 categories.** Mask2Former, without any bells-and-whistles, outperforms the challenge winner (which uses extra training data, model ensemble, *etc.*) on the **test-dev** set. We only train our model on the COCO **train2017** set with ImageNet-22K pre-trained checkpoint.

method	backbone	PQ	PQ Th	PQ St	SQ	RQ
Max-DeepLab [97]	Max-L	51.3	57.2	42.4	82.5	61.3
Panoptic FCN [135]	Swin-L	52.7	59.4	42.5	-	-
MaskFormer [8]	Swin-L	53.3	59.1	44.5	82.0	64.1
Panoptic SegFormer [147]	PVTv2-B5 [148]	54.4	61.1	44.3	83.3	64.6
K-Net [130]	Swin-L	55.2	61.2	46.2	-	-
Megvii (challenge winner)	-	54.7	64.6	39.8	83.6	64.3
Mask2Former (ours)	Swin-L	58.3	65.1	48.1	84.1	68.6

4.6.1.1 Panoptic segmentation

In Table 4.8, we report Mask2Former with various backbones on COCO panoptic **val2017**. Mask2Former outperforms *all* existing panoptic segmentation models with various backbones. Our best model sets a new state-of-the-art of 57.8 PQ.

In Table 4.9, we further report the best Mask2Former model on the **test-dev** set. Note that Mask2Former, **trained only with the standard train2017 data**, achieves the *absolute* new state-of-the-art performance on both validation and test set. Mask2Former even outperforms the best COCO competition entry which uses extra training data and test-time augmentation.

4.6.1.2 Instance segmentation

In Table 4.10, we report Mask2Former with various backbones on COCO **val2017**. Mask2Former outperforms the best single-scale model, HTC++ [96, 129]. Note that it is non-trivial to do multi-scale inference for instance-level segmentation tasks without introducing complex post-processing like non-maximum suppression. Thus, we only compare Mask2Former with other models with single-scale inference as well. We believe multi-scale inference can further improve Mask2Former performance and it remains an interesting future work.

In Table 4.11, we further report the best Mask2Former model on the **test-dev** set. Mask2Former achieves the *absolute* new state-of-the-art performance on both validation and test set. On the one hand, Mask2Former

Table 4.10: **Instance segmentation on COCO val2017 with 80 categories.** Mask2Former outperforms strong Mask R-CNN [5] baselines with $8\times$ fewer epochs in both AP and AP_{boundary} [10] metrics. Our best model is also competitive with the state-of-the-art specialized instance segmentation model on COCO and has higher boundary quality. For a fair comparison, we only consider single-scale inference and models trained using only COCO train2017 set data. Backbones pre-trained on ImageNet-22K are marked with [†].

	method	backbone	search space	epochs	AP	AP _S	AP _M	AP _L	AP _{boundary}	#params.	FLOPs
CNN backbones	Mask R-CNN [5]	R50	dense anchors	36	37.2	18.6	39.5	53.3	23.1	44M	201G
		R50	dense anchors	400	42.5	23.8	45.0	60.0	28.0	46M	358G
		R101	dense anchors	36	38.6	19.5	41.3	55.3	24.5	63M	266G
		R101	dense anchors	400	43.7	24.6	46.4	61.8	29.1	65M	423G
	Mask2Former (ours)	R50	100 queries	50	43.7	23.4	47.2	64.8	30.6	44M	226G
		R101	100 queries	50	44.2	23.8	47.7	66.7	31.1	63M	293G
Transformer backbones	QueryInst [132]	Swin-L [†]	300 queries	50	48.9	30.8	52.6	68.3	33.5	-	-
	Swin-HTC++ [96, 129]	Swin-B [†]	dense anchors	36	49.1	-	-	-	-	160M	1043G
		Swin-L [†]	dense anchors	72	49.5	31.0	52.4	67.2	34.1	284M	1470G
		Swin-T	100 queries	50	45.0	24.5	48.3	67.4	31.8	47M	232G
	Mask2Former (ours)	Swin-S	100 queries	50	46.3	25.3	50.3	68.4	32.9	69M	313G
		Swin-B	100 queries	50	46.7	26.1	50.5	68.8	33.2	107M	466G
		Swin-B [†]	100 queries	50	48.1	27.8	52.0	71.1	34.4	107M	466G
		Swin-L [†]	<i>200 queries</i>	<i>100</i>	50.1	29.9	53.9	72.1	36.2	216M	868G

Table 4.11: **Instance segmentation on COCO test-dev with 80 categories.** Mask2Former is extremely good at segmenting large objects: We can even outperform the challenge winner (which uses extra training data, model ensemble, *etc.*) on AP^L by a large margin without any bells-and-whistles. We only train our model on the COCO `train2017` set with ImageNet-22K pre-trained checkpoint.

method	backbone	AP	AP50	AP75	AP^S	AP^M	AP^L
QueryInst [132]	Swin-L	49.1	74.2	53.8	31.5	51.8	63.2
Swin-HTC++ [96, 129]	Swin-L	50.2	-	-	-	-	-
Swin-HTC++ [96, 129] (multi-scale)	Swin-L	51.1	-	-	-	-	-
Megvii (challenge winner)	-	53.1	76.8	58.6	36.6	56.5	67.7
Mask2Former (ours)	Swin-L	50.5	74.9	54.9	29.1	53.8	71.2

is extremely good at segmenting large objects: We can even outperform the challenge winner (which uses extra training data, model ensemble, *etc.*) on AP^L by a large margin without any bells-and-whistles. On the other hand, the poor performance on small objects leaves room for further improvement in the future.

4.6.1.3 Semantic segmentation

In Table 4.12, we report Mask2Former with various backbones on ADE20K `val`. Mask2Former outperforms *all* existing semantic segmentation models with various backbones. Our best model sets a new state-of-the-art of 57.7 mIoU.

In Table 4.13, we further report the best Mask2Former model on the `test` set. Following [8], we train Mask2Former on the union of ADE20K `train` and `val` set with ImageNet-22K pre-trained checkpoint and use multi-scale inference. Mask2Former is able to outperform previous state-of-the-art methods on all metrics.

4.6.2 Additional Datasets

We study Mask2Former on three image segmentation tasks (panoptic, instance and semantic segmentation) using four datasets. Here we report additional results on Cityscapes [12], ADE20K [94] and Mapillary Vistas [13] as well as more detailed training settings.

Table 4.12: **Semantic segmentation on ADE20K val with 150 categories.** Mask2Former consistently outperforms MaskFormer [8] by a large margin with different backbones (all Mask2Former models use MSDeformAttn [138] as pixel decoder, except Swin-L-FaPN uses FaPN [131]). Our best model outperforms the best specialized model, BEiT [143], with less than half of the parameters. We report both single-scale (s.s.) and multi-scale (m.s.) inference results. Backbones pre-trained on ImageNet-22K are marked with [†].

	method	backbone	crop size	mIoU (s.s.)	(m.s.)	#params	FLOPs
CNN	MaskFormer [8]	R50	512×512	44.5	46.7	41M	53G
		R101	512×512	45.5	47.2	60M	73G
	Mask2Former (ours)	R50	512×512	47.2	49.2	44M	71G
		R101	512×512	47.8	50.1	63M	90G
Transformer backbones	Swin-UperNet [96, 123]	Swin-L [†]	640×640	-	53.5	234M	647G
	FaPN-MaskFormer [8, 131]	Swin-L [†]	640×640	55.2	56.7	-	-
	BEiT-UperNet [123, 143]	BEiT-L [†]	640×640	-	57.0	502M	-
	MaskFormer [8]	Swin-T	512×512	46.7	48.8	42M	55G
		Swin-S	512×512	49.8	51.0	63M	79G
		Swin-B	640×640	51.1	52.3	102M	195G
		Swin-B [†]	640×640	52.7	53.9	102M	195G
		Swin-L [†]	640×640	54.1	55.6	212M	375G
		Swin-L-FaPN [†]	640×640	56.4	57.7	217M	-
	Mask2Former (ours)	Swin-T	512×512	47.7	49.6	47M	74G
		Swin-S	512×512	51.3	52.4	69M	98G
		Swin-B	640×640	52.4	53.7	107M	223G
		Swin-B [†]	640×640	53.9	55.1	107M	223G
		Swin-L [†]	640×640	56.1	57.3	215M	403G
		Swin-L-FaPN [†]	640×640	56.4	57.7	217M	-

Table 4.13: **Semantic segmentation on ADE20K test with 150 categories.** Mask2Former outperforms previous state-of-the-art methods on all three metrics: pixel accuracy (P.A.), mIoU, as well as the final test score (average of P.A. and mIoU). We train our model on the union of ADE20K train and val set with ImageNet-22K pre-trained checkpoint following [8] and use multi-scale inference.

method	backbone	P.A.	mIoU	score
SETR [109]	ViT-L	78.35	45.03	61.69
Swin-UperNet [96, 123]	Swin-L	78.42	47.07	62.75
MaskFormer [8]	Swin-L	79.36	49.67	64.51
Mask2Former (ours)	Swin-L-FaPN	79.80	49.72	64.76

4.6.2.1 Cityscapes

Cityscapes is an urban egocentric street-view dataset with high-resolution images (1024×2048 pixels). It contains 2975 images for training, 500 images for validation and 1525 images for testing with a total of 19 classes.

Training settings. For all three segmentation tasks we use a crop size of

Table 4.14: **Image segmentation results on Cityscapes val.** We report both single-scale (s.s.) and multi-scale (m.s.) inference results for PQ and mIoU. All other metrics are evaluated with *single-scale* inference. Since Mask2Former is an end-to-end model, we only use single-scale inference for instance-level segmentation tasks to avoid the need for further post-processing (*e.g.*, NMS).

method	backbone	panoptic model			instance model		semantic model	
		PQ (s.s.)	PQ (m.s.)	AP Th _{pan}	AP	AP50	mIoU (s.s.)	mIoU (m.s.)
Panoptic-DeepLab [6]	R50	60.3	-	32.1	-	-	-	-
	X71 [70]	63.0	64.1	35.3	-	-	-	-
	SWideRNet [145]	66.4	67.5	40.1	-	-	-	-
Panoptic FCN [135]	Swin-L [†]	65.9	-	-	-	-	-	-
Segmenter [110]	ViT-L [†]	-	-	-	-	-	-	81.3
SETR [109]	ViT-L [†]	-	-	-	-	-	-	82.2
SegFormer [146]	MiT-B5	-	-	-	-	-	-	84.0
Mask2Former (ours)	R50	62.1	-	37.3	37.4	61.9	79.4	82.2
	R101	62.4	-	37.7	38.5	63.9	80.1	81.9
	Swin-T	63.9	-	39.1	39.7	66.9	82.1	83.0
	Swin-S	64.8	-	40.7	41.8	70.4	82.6	83.6
	Swin-B [†]	66.1	-	42.8	42.0	68.8	83.3	84.5
	Swin-L [†]	66.6	-	43.6	43.7	71.4	83.3	84.3

Table 4.15: **Image segmentation results on ADE20K val.** Mask2Former is competitive with specialized models on ADE20K. Panoptic segmentation models use single-scale inference by default, multi-scale numbers are marked with *. For semantic segmentation, we report both single-scale (s.s.) and multi-scale (m.s.) inference results.

method	backbone	panoptic model			instance model			semantic model	
		PQ	AP Th _{pan}	mIoU _{pan}	AP	AP ^S	AP ^M	mIoU (s.s.)	mIoU (m.s.)
MaskFormer [8]	R50	34.7	-	-	-	-	-	-	-
Panoptic-DeepLab [6]	SWideRNet [145]	37.9*	-	50.0*	-	-	-	-	-
Swin-UpperNet [96, 123]	Swin-L [†]	-	-	-	-	-	-	-	53.5
MaskFormer [8]	Swin-L [†]	-	-	-	-	-	-	54.1	55.6
FaPN-MaskFormer [8, 131]	Swin-L [†]	-	-	-	-	-	-	55.2	56.7
BEiT-UpperNet [123, 143]	BEiT-L [†]	-	-	-	-	-	-	-	57.0
Mask2Former (ours)		39.7	26.5	46.1	26.4	10.4	28.9	47.2	49.2
	Swin-L [†]	48.1	34.2	54.5	34.9	16.3	40.0	56.1	57.3
	Swin-L-FaPN [†]	46.2	33.2	55.4	33.4	14.6	37.6	56.4	57.7

Table 4.16: **Image segmentation results on Mapillary Vistas val.** Mask2Former is competitive with specialized models on Mapillary Vistas. Panoptic segmentation models use single-scale inference by default, multi-scale numbers are marked with *. For semantic segmentation, we report both single-scale (s.s.) and multi-scale (m.s.) inference results.

method	backbone	panoptic model		semantic model	
		PQ	mIoU _{pan}	mIoU (s.s.)	mIoU (m.s.)
Panoptic-DeepLab [6]	ensemble	42.2*	58.7*	-	-
	SWideRNet [145]	43.7	59.4	-	-
	SWideRNet [145]	44.8*	60.0*	-	-
Panoptic FCN [135]	Swin-L [†]	45.7	-	-	-
MaskFormer [8]	R50	-	-	53.1	55.4
HMSANet [125]	HRNet [72]	-	-	-	61.1
Mask2Former (ours)	R50	36.3	50.7	57.4	59.0
	Swin-L [†]	45.5	60.8	63.2	64.7

512×1024 , a batch size of 16 and train all models for 90k iterations. During inference, we operate on the whole image (1024×2048). Other implementation details largely follow Section 4.4.1 (panoptic and instance segmentation follow semantic segmentation training settings), except that we use 200 queries for panoptic and instance segmentation models with Swin-L backbone. All other backbones or semantic segmentation models use 100 queries. **Results.** In Table 4.14, we report Mask2Former with various backbones on Cityscapes for three segmentation tasks and compare it with other state-of-the-art methods *without using extra data*. For panoptic segmentation, Mask2Former with Swin-L backbone outperforms the state-of-the-art Panoptic-DeepLab [6] with SWideRnet [145] using single-scale inference. For semantic segmentation, Mask2Former with Swin-B backbone outperforms the state-of-the-art SegFormer [146].

4.6.2.2 ADE20K

Training settings. For panoptic and instance segmentation, we use the exact same training parameters as we used for semantic segmentation, except that we always use a crop size of 640×640 for all backbones. Other implementation details largely follow Section 4.4.1, except that we use 200 queries for panoptic and instance segmentation models with Swin-L backbone. All other backbones or semantic segmentation models use 100 queries.

Results. In Table 4.15, we report the results of Mask2Former with various

backbones on ADE20K for three segmentation tasks and compare it with other state-of-the-art methods. Mask2Former with Swin-L backbone sets a new state-of-the-art performance on ADE20K for panoptic segmentation. As there are few papers reporting results on ADE20K, we hope this experiment can serve as a useful benchmark for future research.

4.6.2.3 Mapillary Vistas

Mapillary Vistas is a large-scale urban street-view dataset with 18k, 2k and 5k images for training, validation and testing. It contains images with a variety of resolutions, ranging from 1024×768 to 4000×6000 . We only report panoptic and semantic segmentation results for this dataset.

Training settings. For both panoptic and semantic segmentation, we follow the same data augmentation of [8]: standard random scale jittering between 0.5 and 2.0, random horizontal flipping, random cropping with a crop size of 1024×1024 as well as random color jittering. We train our model for 300k iterations with a batch size of 16 using the “poly” learning rate schedule [26]. During inference, we resize the longer side to 2048 pixels. Our panoptic segmentation model with a Swin-L backbone uses 200 queries. All other backbones or semantic segmentation models use 100 queries.

Results. In Table 4.16, we report Mask2Former with various backbones on Mapillary Vistas for panoptic and semantic segmentation tasks and compare it with other state-of-the-art methods. Our Mask2Former is very competitive compared to state-of-the-art specialized models even if it is not designed for Mapillary Vistas.

4.6.3 Additional Ablation Studies

We perform additional ablation studies of Mask2Former using the same settings: a single ResNet-50 backbone [77].

4.6.3.1 Convergence analysis

We train Mask2Former with 12, 25, 50 and 100 epochs with either standard scale augmentation (Standard Aug.) [78] or the more recent large-

scale jittering augmentation (LSJ Aug.) [141, 142]. As shown in Figure 4.4, Mask2Former converges in 25 epochs using standard augmentation and almost converges in 50 epochs using large-scale jittering augmentation. This shows that Mask2Former with our proposed Transformer decoder converges faster than models using the standard Transformer decoder: *e.g.*, DETR [91] and MaskFormer [8] require 500 epochs and 300 epochs respectively.

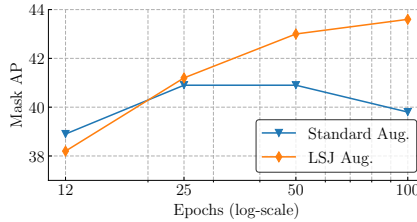


Figure 4.4: **Convergence analysis.** We train Mask2Former with different epochs using either standard scale augmentation (Standard Aug.) [78] or the more recent large-scale jittering augmentation (LSJ Aug.) [141, 142]. Mask2Former converges in 25 epochs using standard augmentation and almost converges in 50 epochs using large-scale jittering augmentation. Using LSJ also improves performance with longer training epochs (*i.e.*, with more than 25 epochs).

4.6.3.2 Object queries analysis

Object queries play an important role in Mask2Former. We ablate different design choices of object queries including the number of queries and making queries learnable.

Number of queries. We study the effect of different numbers of queries for three image segmentation tasks in Table 4.17a. For instance and semantic segmentation, using 100 queries achieves the best performance, while using 200 queries can further improve panoptic segmentation results. As panoptic segmentation is a combination of instance and semantic segmentation, it has more segments per image than the other two tasks. This ablation suggests that picking the number of queries for Mask2Former may depend on the number of segments per image for a particular task or dataset.

Learnable queries. An object query consists of two parts: object query features and object query positional embeddings. Object query features are only used as the initial input to the Transformer decoder and are updated through decoder layers, whereas query positional embeddings are added to

query features in every Transformer decoder layer when computing the attention weights. In DETR [91], query features are zero-initialized and query positional embeddings are learnable; furthermore, there is no direct supervision on these query features before feeding them into the Transformer (since they are zero vectors). In our Mask2Former, we still make query positional embeddings learnable. In addition, we make query features learnable as well and directly apply losses on these learnable query features before feeding them into the Transformer decoder.

In Table 4.17b, we compare our learnable query features with zero-initialized query features in DETR. We find it is important to directly supervise object queries even before feeding them into the Transformer decoder. Learnable queries *without* supervision perform similarly well as zero-initialized queries in DETR.

4.6.3.3 MaskFormer *vs.* Mask2Former

Mask2Former builds upon the same meta architecture as MaskFormer [8] with two major differences: (1) We use more advanced training parameters summarized in Table 4.18a, and (2) We propose a new Transformer decoder with masked attention, instead of using the standard Transformer decoder, as well as some optimization improvements summarized in Table 4.18b. To better understand Mask2Former’s improvements over MaskFormer, we perform ablation studies on training parameter improvements and Transformer decoder improvements in isolation.

In Table 4.18c, we study our new training parameters. We train the MaskFormer model with either its original training parameters in [8] or our new training parameters. We observe significant improvements of using our new training parameters for MaskFormer as well. This shows the new training parameters are also generally applicable to other models.

In Table 4.18d, we study our new Transformer decoder. We train a MaskFormer model and a Mask2Former model with the exact same backbone, *i.e.*, a ResNet-50; pixel decoder, *i.e.*, a FPN; and training parameters. That is, the only difference is in the Transformer decoder, summarized in Table 4.18b. We observe improvements for all three tasks, suggesting that the new Transformer decoder itself is indeed better than the standard Transformer decoder.

Table 4.17: **Analysis of object queries.** Table 4.17a: ablation on number of queries. Table 4.17b: ablation on using learnable queries.

(a) **Number of queries ablation.** For instance and semantic segmentation, using 100 queries achieves the best performance while using 200 queries can further improve panoptic segmentation results.

	AP (COCO)	PQ (COCO)	mIoU (ADE20K)	FLOPs (COCO)
50	42.4	50.5	46.2	217G
100	43.7	51.9	47.2	226G
200	43.5	52.2	47.0	246G
300	43.5	52.1	46.5	265G
1000	40.3	50.7	44.8	405G

(b) **Learnable queries ablation.** It is important to supervise object queries before feeding them into the Transformer decoder. Learnable queries *without* supervision perform similarly well as zero-initialized queries in DETR.

	AP (COCO)	PQ (COCO)	mIoU (ADE20K)	FLOPs (COCO)
zero-initialized (DETR [91])	42.9	51.2	45.5	226G
learnable <i>w/o</i> supervision	42.9	51.2	47.0	226G
learnable <i>w/</i> supervision	43.7	51.9	47.2	226G

4.6.4 Visualization

We visualize sample predictions of the Mask2Former model with Swin-L [96] backbone on three tasks: COCO panoptic **val2017** set for panoptic segmentation (57.8 PQ) in Figure 4.5, COCO **val2017** set for instance segmentation (50.1 AP) in Figure 4.6 and ADE20K validation set for semantic segmentation (57.7 mIoU, multi-scale inference) in Figure 4.7.

Table 4.18: **MaskFormer vs. Mask2Former.** Table 4.18a and Table 4.18b provide an in-depth comparison between MaskFormer and our Mask2Former settings. Table 4.18c: MaskFormer benefits from our new training parameters as well. Table 4.18d: Comparison between MaskFormer and our Mask2Former with the exact same backbone, pixel decoder and training parameters. The improvements solely come from a better Transformer decoder.

(a) Comparison of training parameters for MaskFormer [8] and our Mask2Former on the COCO dataset. *: in the original MaskFormer implementation, the model is trained with a batch size of 64 for 300 epochs. We find MaskFormer achieves similar performance when trained with a batch size of 16 for 75 epochs, *i.e.*, the same number of iterations with a smaller batch size.

training parameters	MaskFormer	Mask2Former (ours)
learning rate	0.0001	0.0001
weight decay	0.0001	0.05
batch size	16*	16
epochs	75*	50
data augmentation	standard scale aug. w/ crop	LSJ aug.
λ_{cls}	1.0	2.0
$\lambda_{\text{focal}} / \lambda_{\text{ce}}$	20.0 / -	- / 5.0
λ_{dice}	1.0	5.0
mask loss	mask	12544 sampled points

(b) Comparison of Transformer decoder in MaskFormer [8] and our Mask2Former. SA: self-attention, CA: cross-attention, FFN: feed-forward network, MA: masked attention, p.e.: positional embedding.

Transformer decoder	MaskFormer	Mask2Former (ours)
# of layers	6	9
single layer	SA-CA-FFN	MA-SA-FFN
dropout	0.1	0.0
feature resolution	$\{1/32\} \times 6$	$\{1/32, 1/16, 1/8\} \times 3$
input query features	zero init.	learnable
query p.e.	learnable	learnable

(c) Improvements from better **training parameters.**

model	training params.	AP (COCO)	PQ (COCO)	mIoU (ADE20K)
MaskFormer	MaskFormer	34.0	46.5	44.5
MaskFormer	Mask2Former	37.8 (+3.8)	48.2 (+1.7)	45.3 (+0.8)

(d) Improvements from better **Transformer decoder.**

Transformer decoder	pixel decoder	AP (COCO)	PQ (COCO)	mIoU (ADE20K)
MaskFormer	FPN	37.8	48.2	45.3
Mask2Former	FPN	41.5 (+3.7)	50.7 (+2.5)	45.6 (+0.3)

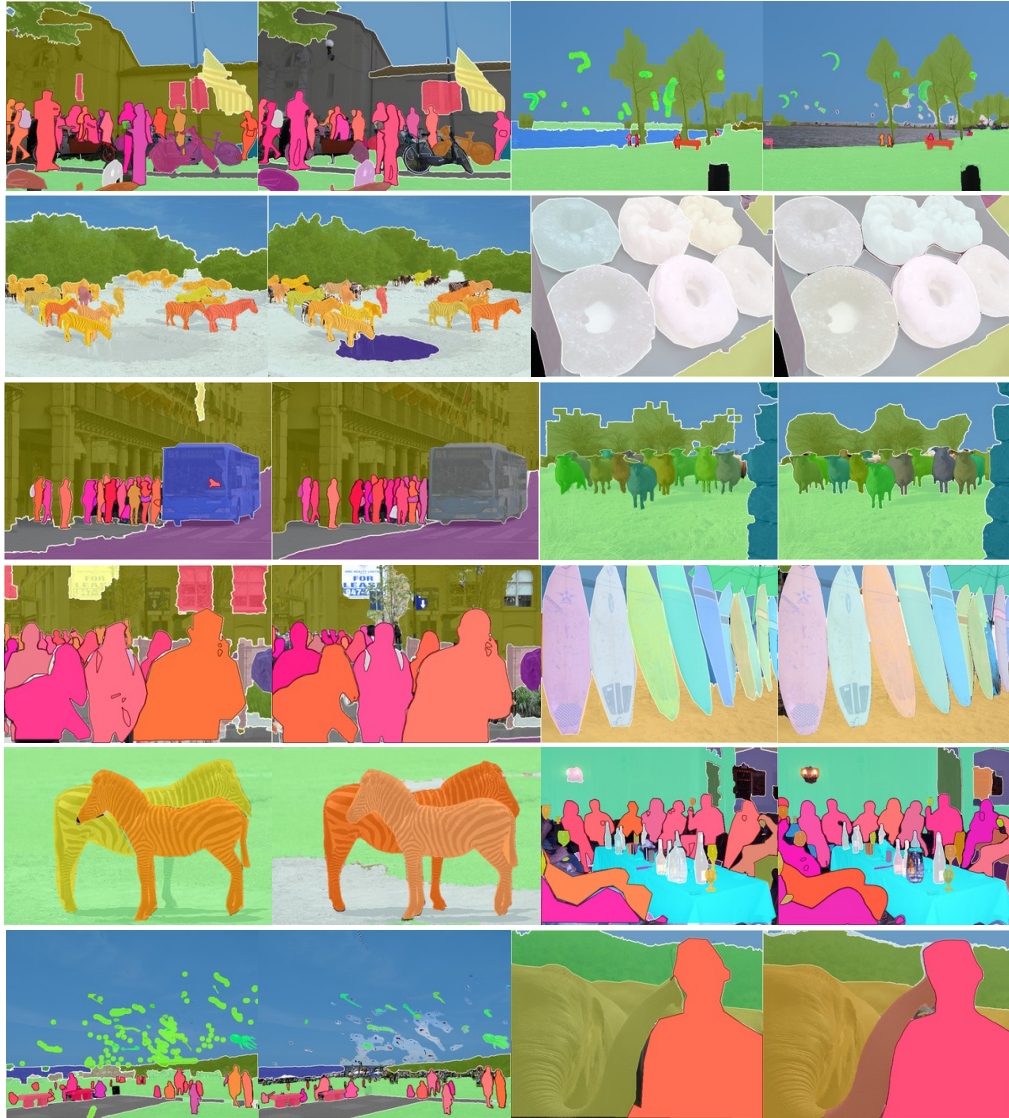


Figure 4.5: Visualization of **panoptic segmentation** predictions on the COCO panoptic dataset: Mask2Former with Swin-L backbone which achieves 57.8 PQ on the validation set. First and third columns: ground truth. Second and fourth columns: prediction. **Last row shows failure cases.**



Figure 4.6: Visualization of **instance segmentation** predictions on the COCO dataset: Mask2Former with Swin-L backbone which achieves 50.1 AP on the validation set. First and third columns: ground truth. Second and fourth columns: prediction. **Last row shows failure cases.** We show predictions with confidence scores greater than 0.5.

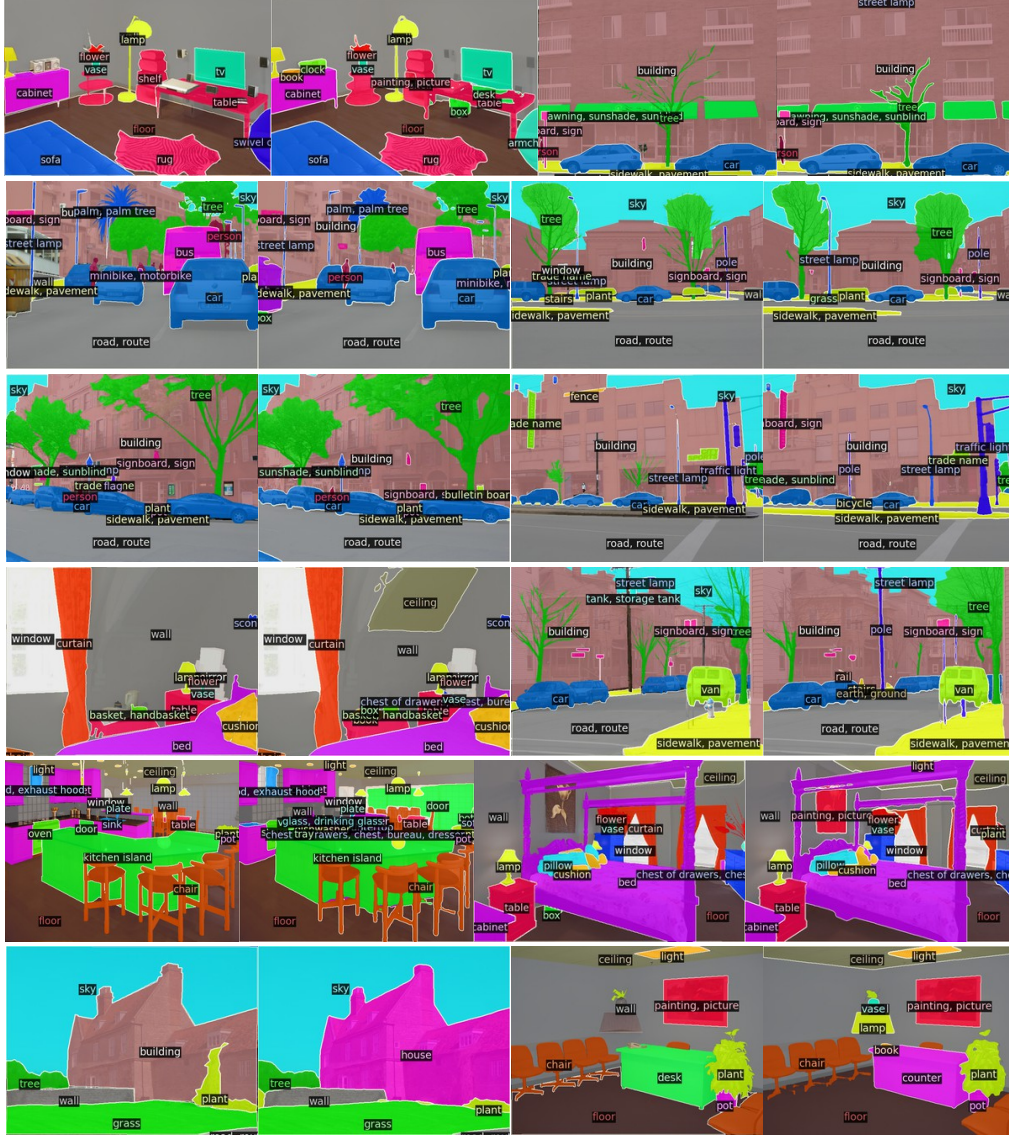


Figure 4.7: Visualization of **semantic segmentation** predictions on the ADE20K dataset: Mask2Former with Swin-L backbone which achieves 57.7 mIoU (multi-scale) on the validation set. First and third columns: ground truth. Second and fourth columns: prediction. **Last row shows failure cases.**

CHAPTER 5

IMAGE SEGMENTATION EVALUATION

5.1 Introduction

The Common Task Framework [149], in which standardized tasks, datasets, and evaluation metrics are used to track research progress, yields impressive results. For example, researchers working on the instance segmentation task, which requires an algorithm to delineate objects with pixel-level binary masks, have improved the standard Average Precision (AP) metric on COCO [11] by an astonishing 86% (relative) from 2015 [150] to 2019 [151].

However, this progress is not equal across all error modes, because different evaluation metrics are sensitive (or insensitive) to different types of errors. If a metric is used for a prolonged time, as in the Common Task Framework, then the corresponding sub-field most rapidly resolves the types of errors to which this metric is sensitive. Research directions that improve other error types typically advance more slowly, as such progress is harder to quantify.

This phenomenon is at play in instance segmentation, where, among the multitude of papers contributing to the impressive 86% relative improvement in AP (*e.g.*, [133, 152–155]), only a few address mask boundary quality.

Note that mask boundary quality is an essential aspect of image segmentation, as various downstream applications directly benefit from more precise object segmentations [156–158]. However, the dominant family of Mask R-CNN-based methods [5] are well-known to predict low-fidelity, blobby masks (see Figure 5.1). This observation suggests that the current evaluation metrics may have limited sensitivity to mask prediction errors near object boundaries.

To understand why, we start by analyzing Mask Intersection-over-Union (Mask IoU), the underlying measure used in AP to compare predicted and ground truth masks. Mask IoU divides the intersection area of two masks

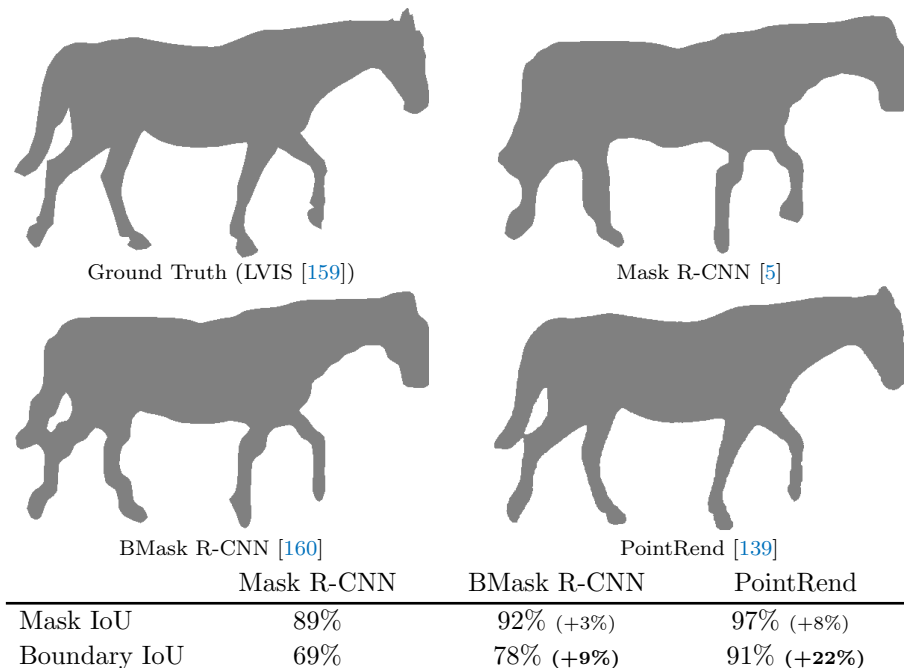


Figure 5.1: Given the bounding box for a horse, the mask predicted by Mask R-CNN scores a high Mask IoU value (89%) relative to the ground truth despite having low-fidelity, blobby boundaries. The recently proposed BMask R-CNN [160] and PointRend [139] methods predict masks with higher fidelity boundaries, yet these clear visual improvements only marginally improve Mask IoU (+3% and +8%, respectively). In contrast, our proposed **Boundary IoU** measure demonstrates greater sensitivity to boundary errors, and thus provides a clear, quantitative gradient that rewards improvements to boundary segmentation quality.

by the area of their union. This measure values all pixels equally and is therefore less sensitive to boundary quality in *larger objects*: The number of interior pixels grows quadratically in object size and can far exceed the number of boundary pixels, which only grows linearly. In this chapter we aim to identify a measure for image segmentation that is sensitive to boundary quality across all scales.

Towards this goal we start by studying standard segmentation measures like Mask IoU and boundary-focused measures such as Trimap IoU [26, 161] and F-measure [162–164]. We study error-sensitivity characteristics of each measure by generating a variety of error types on top of the high-quality ground truth masks from the LVIS dataset [159]. Our analysis confirms that Mask IoU is less sensitive to errors in larger objects. In addition, the analysis reveals limitations of existing boundary-focused measures, such as

asymmetries and instability to small changes in mask quality.

Based on these insights we propose a new **Boundary IoU** measure. Boundary IoU is simple and easy to compute. Instead of considering all pixels, it calculates the intersection-over-union for mask pixels within a certain distance from the corresponding ground truth or prediction boundary contours. Our analysis demonstrates that Boundary IoU measures boundary quality of large objects well, unlike Mask IoU, and it does not over-penalize errors on small objects. An illustrative examples compares Boundary IoU to Mask IoU in Figure 2.1.

Boundary IoU enables new task-level evaluation metrics. For the task of instance segmentation [11], we propose **Boundary Average Precision** (Boundary AP), and for panoptic segmentation [4], we propose **Boundary Panoptic Quality** (Boundary PQ).

Boundary AP assesses all relevant aspects of instance segmentation, simultaneously taking into account categorization, localization, and segmentation quality, unlike prior boundary-focused metrics for instance segmentation like AF [165] that ignore false positive rates. We test Boundary AP on three common datasets: COCO [11], LVIS [159], and Cityscapes [12]. With real predictions from recent instance segmentation methods that directly aim to improve boundary quality [139, 160], we verify that Boundary AP tracks improvements better than Mask AP. With synthetic predictions, we show that Boundary AP is significantly more sensitive to large-object boundary quality than Mask AP.

For panoptic segmentation, we apply Boundary PQ to the COCO [4] and Cityscapes [12] panoptic datasets. We test the new metric with synthetic predictions and show that it is more sensitive than the previous metric based on Mask IoU. Finally, we evaluate the performance of various recent instance and panoptic segmentation models with the new evaluation metrics to ease comparison for future research.

These new metrics reveal improvements in boundary quality that are generally ignored by Mask IoU-based evaluation metrics. We hope that the adoption of these new boundary-sensitive evaluations can enable faster progress towards segmentation models with better boundary quality.

5.2 Preliminaries

Image segmentation tasks like semantic, instance, or panoptic segmentation are evaluated by comparing segmentation masks predicted by a system to ground truth masks provided by annotators. Modern evaluation metrics for these tasks are based on segmentation quality measures that evaluate consistency between ground truth object shape G and prediction shape P represented by binary masks of a fixed resolution. We define the most common segmentation quality measures and the new Boundary IoU measure in Table 5.1 using the unified notation presented in Table 5.2. We split the measures into mask- and boundary-based types and discuss their differences next.

Mask-based segmentation measures take into account all pixels of an object mask. The first PASCAL VOC semantic segmentation track in 2007 [166] used Pixel Accuracy measure to evaluate predictions. For each class it calculates the ratio of correctly labeled ground truth pixels (see Table 5.1). Pixel accuracy is not symmetric and biased toward prediction masks that are larger than ground truth masks. Subsequently, PASCAL VOC [2] switched its evaluation to the Mask Intersection-over-Union (Mask IoU) measure.

Mask IoU segmentation consistency measure divides the number of pixels in the intersection of the prediction and ground truth masks by the number of pixels in their union (see Table 5.1). The measure is widely used in the evaluation metrics for most popular semantic, instance, and panoptic segmentation tasks [2, 4, 11] and datasets [11, 12, 94, 95]. Unlike Pixel Accuracy, Mask IoU is symmetric; however, as we will show in this chapter, it demonstrates unbalanced responsiveness to the boundary quality across object sizes.

Boundary-based segmentation measures evaluate segmentation quality by estimating contour alignment between predicted and ground truth masks. Unlike mask-based measures, these measures only evaluate the pixels that lie directly on the masks’ contours or in their close proximity.

Trimap IoU [26, 161] is a boundary-based segmentation measure that calculates IoU in a narrow band of pixels within a pixel distance d from the contour of the ground truth mask (see Table 5.1). In contrast to Mask IoU, Trimap IoU reacts similarly to comparable pixel errors across object scales because it calculates IoU only for pixels around the contour. However, un-

Table 5.1: Existing segmentation measures and the new Boundary IoU defined with the unified notation from Table 5.2. For each measure we detail several properties. **Symmetric**: whether the swap of ground truth and prediction masks changes measure’s value. **Preference**: whether the measure biases better scores to a certain type of prediction. **Insensitivity**: types of errors the measure is less sensitive to.

Name	Type	Definition	Symmetric	Preference	Insensitivity
Pixel accuracy	mask-based	$\frac{ G \cap P }{ G }$	✗	larger prediction	–
Mask IoU	mask-based	$\frac{ G \cap P }{ G \cup P }$	✓	–	boundary errors
Trimap IoU	boundary-based	$\frac{ G_d \cap (G \cap P) }{ G_d \cap (G \cup P) } = \frac{ (G_d \cap G) \cap P }{ (G_d \cap G) \cup (G_d \cap P) }$	✗	larger prediction	errors far from ground truth boundary
F-measure	boundary-based	$\frac{2 \cdot \tilde{p} \cdot \tilde{r}}{\tilde{p} + \tilde{r}}, \tilde{p} = \frac{ P_1 \cap G_d }{ P_1 }, \tilde{r} = \frac{ G_1 \cap P_d }{ G_1 }$	✓	–	errors on small objects
Boundary IoU	boundary-based	$\frac{ (G_d \cap G) \cap (P_d \cap P) }{ (G_d \cap G) \cup (P_d \cap P) }$	✓	–	errors far from predicted and ground truth boundaries

Table 5.2: Notation used in this chapter. We define a contour as the 1d line comprised of the set of mask pixels that touches the background. The boundary is a 2D region consisting of pixels within pixel distance d from the contour pixels. A boundary region can be constructed by dilating the contour line by d pixels.

Notation	Definition
G	ground truth binary mask
P	prediction binary mask
G_1, P_1	set of pixels on the contour line of the binary mask
G_d, P_d	set of pixels in the boundary region of the binary mask
d	pixel width of the boundary region

like Mask IoU, the measure is not symmetric and favors predictions whose masks are larger than the corresponding ground truth masks. Moreover, the measure ignores prediction errors that appear outside the band around the ground truth contour.

F-measure was initially proposed for edge detection [167], but it is also used to evaluate segmentation quality [163, 164]. $\text{F-measure} = 2 \cdot p \cdot r / (p + r)$, where p and r denote precision and recall. In the original definition, p and r are calculated by matching prediction and ground truth contour pixels within the pixel distance threshold d via bipartite matching. However, the matching process is computationally expensive for high resolution masks and large datasets; therefore, [163, 164] proposed an approximation procedure to compute the precision and recall by allowing duplicate matches, which we denote by $\tilde{p}$ and $\tilde{r}$. In this case, $\tilde{p}$ computes the ratio of pixels in the prediction contour that lie within a distance d from the ground truth contours, whereas $\tilde{r}$ computes a similar ratio for the pixels of the ground truth contour, see Table 5.1. In the rest of the chapter we use the approximate formulation of F-measure. The measure is symmetric and tolerates small contour misalignments that can be attributed to ambiguity, however it ignores significant errors when the object size is comparable to d . For example, this occurs with reasonable choices of d and small objects commonly found in datasets (*e.g.*, COCO [11]).

Trimap and F-measure are often used to evaluate boundary quality for semantic segmentation tasks in an ad-hoc fashion. For example, Trimap IoU is used as an extra evaluation to show boundary quality improvement [26,

168], but it is not reported by most segmentation methods. In the next section we will study both measures in detail and analyze their behavior across different error types and object sizes.

5.3 Sensitivity Analysis

In Section 5.4 and Section 5.5 we will compare several mask consistency measures by observing how a measure’s value changes in response to errors of different magnitudes. We will observe and interpret these curves to draw conclusions about the behavior of these measures, a methodology that we refer to as *sensitivity analysis*.

To enable a systematic comparison, we *simulate* a set of common segmentation errors across different mask sizes by generating pseudo-predictions from ground truth annotations. This approach allows us explicitly control the type and severity of the errors used in the analysis. Moreover, the use of pseudo-predictions avoids any bias toward specific segmentation models which makes the analysis more robust and general. A potential limitation of this approach is that simulated errors may not fully represent errors created by real models. We aim to counteract this limitation by using a diverse set of error types. Figure 5.2 depicts an example of each error type we consider:

- **Scale error.** Dilation/erosion are applied to the ground truth masks. The error severity is controlled by the kernel radius of the morphological operations.
- **Boundary localization error.** Random Gaussian noise is added to the coordinate of each vertex in polygons that represent ground truth masks. The error severity is controlled by the standard deviation (std) of the Gaussian noise.
- **Object localization error.** Ground truth masks are shifted with random offsets. The error severity is controlled by the pixel length of the shift.
- **Boundary approximation error.** The `simplify` function from Shapely [169] removes vertices from the polygons that represent ground truth masks while keeping the simplified polygons as close to the original ones as possible. The error severity is controlled by the error tolerance parameter of the `simplify` function.
- **Inner mask error.** Holes of random shape are added to ground truth

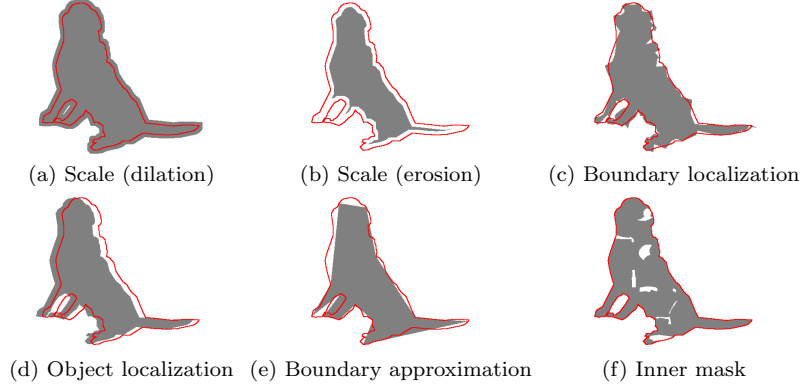


Figure 5.2: **Examples of error types** generated from a single ground truth mask. The red contour represents the ground truth contour. *Scale errors*: (a) dilation and (b) erosion of the mask. *Boundary localization error*: (c) adding random Gaussian noise to each vertex in polygons. *Object localization error*: (d) shifting masks. *Boundary approximation error*: (e) simplifying polygons. *Inner mask error*: (f) adding holes to masks.

masks. The error severity is controlled by the number of holes added. While this error type is not common for modern segmentation approaches, we include it to assess the effect of interior mask errors.

Implementation details. For the analysis, we randomly sample instance masks from the LVIS v0.5 [159] validation set. The dataset is selected due to its high-quality annotations. Using these masks, for each segmentation error type we create multiple sets of pseudo-predictions by varying the severity of the error. To analyze a segmentation measure, we report its mean and standard deviation across a set of pseudo-predictions that represent a given error type of a fixed severity. We will also compare segmentation measures across different object sizes by generating a separate set of pseudo-predictions using ground truth objects within a specific mask area range. For all boundary-based measures that use pixel distance parameter, d , we set it to 2% of the image diagonal for fair comparison.

5.4 Analysis of Existing Segmentation Measures

First, we analyze the standard Mask IoU segmentation consistency measure from both theoretical and empirical perspectives. Then, we study two exist-

ing alternatives – Trimap IoU and F-measure boundary-based measures.

5.4.1 Mask IoU

Theoretical analysis. Mask IoU is scale-invariant w.r.t. object area. For a fixed Mask IoU value, a larger object will have more incorrect pixels and the change in incorrect pixel count grows in proportional to the change in object area (as Mask IoU is a ratio of areas). However, when scaling up a typical object, the number of interior pixels grows quadratically, whereas the number of contour pixels only grows linearly. These different asymptotic growth rates cause Mask IoU to tolerate a larger number of misclassified pixels *per each unit of contour length* for a larger object.

Empirical analysis. This property corresponds to an assumption that boundary localization error in ground truth annotations (*i.e.*, intrinsic annotation ambiguity) also grows with the object size. However, a classic study on multi-region segmentation [162] shows that the pixel distance between two contours of the same object labeled by different annotators seldomly exceeds 1% of the image diagonal, irrespective of the object size. We confirm this observation by exploring double annotations that are provided for a subset of images in LVIS [159]. In Figure 5.3 we present a random pair of objects with significant size difference. While one of the objects is 100 times larger, the boundary discrepancy within the cropped part, which has the same resolution, is similar between the two objects. Observed results suggest that boundary ambiguity is fixed and independent of objects area. This is likely a consequence of the annotation tool, which includes the ability to zoom while drawing contours.

Using simulated scale errors (described in §5.3) we confirm Mask IoU’s bias in favor of large objects. The dilation/erosion of the ground truth mask by a fixed number of pixels significantly decreases Mask IoU for small objects while Mask IoU grows as object area increases (see Figure 5.4a). Note that Mask IoU’s insensitivity to boundary errors on large objects cannot be addressed by simply increasing the lowest Mask IoU threshold in evaluation metrics like AP or PQ. Such a change does not remedy the bias and will lead to relative over-penalization for smaller objects.



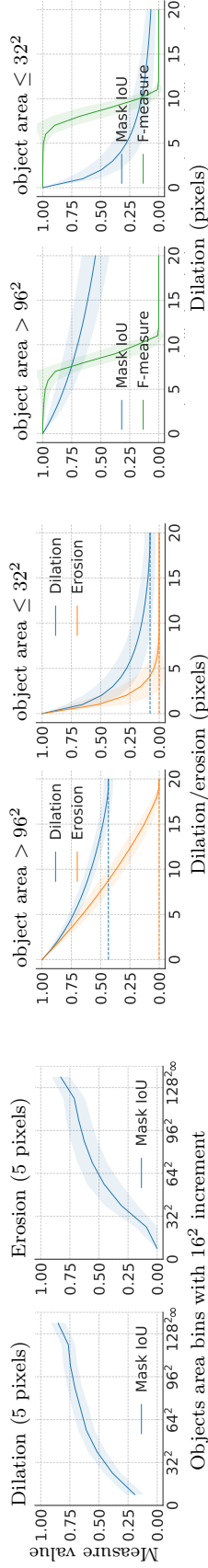
Figure 5.3: The two objects from LVIS [159] dataset annotated independently twice. While the fridge (left) has more than 100 times larger area than the wing (right), on the crop of the same resolution the discrepancy between annotations is visually similar. This simple example supports the observation that the boundary quality in the ground truth is independent of the object size. Mask IoU counts all pixels equally and gives a higher score to fridge, 0.97, *vs.* wing, 0.81, due to the larger internal region with consistent labeling. The bias toward larger objects render Mask IoU inadequate for boundary quality evaluation across object sizes. In contrast, the new Boundary IoU yields much closer scores (0.87 *vs.* 0.81).

5.4.2 Boundary-Based Measures

Next, we will analyze the boundary-based measures Trimap IoU and F-measure. These measures focus on pixels within a distance d from object contours. The parameter d is usually fixed on the dataset [26] or image level [164] which results in these measures treating boundary localization errors independently of the size of the object. By matching the natural characteristic of the ground truth segmentation data, these boundary-based measures are better suited to evaluate improvements in boundary quality across object sizes.

Trimap IoU computes IoU for a region around the ground truth boundary only (*i.e.*, the region is independent of the prediction), and therefore is not symmetric: swapping the prediction and ground truth masks will give a different score. In Figure 5.4b we show that this asymmetry favors predictions that are larger than ground truth masks. For larger (dilated) pseudo-predictions Trimap IoU does not drop below some positive value irrespective of the error severity, whereas for smaller (eroded) pseudo-predictions it drops to 0. Moreover, the measure ignores any errors outside of the ground truth boundary region, penalizing inner mask errors less than Mask IoU (see the Section 5.8 for details).

F-Measure matches the pixels of the predicted and ground truth contours if they are within the pixels distance threshold d . Hence, it ignores small



(a) Mask IoU is biased toward large objects. The pseudo-predictions for larger objects receive higher score under a fixed error severity.

(b) Trimap IoU is not symmetric. It favors predictions larger than ground truth masks (*e.g.* dilated pseudo-predictions).

(c) F-measure completely ignores small contour misalignments and rapidly drops to zero within a short range of severities.

Figure 5.4: Sensitivity analysis across object scales for Mask IoU (a), Trimap IoU (b), and F-measure (c) with scaling error type.

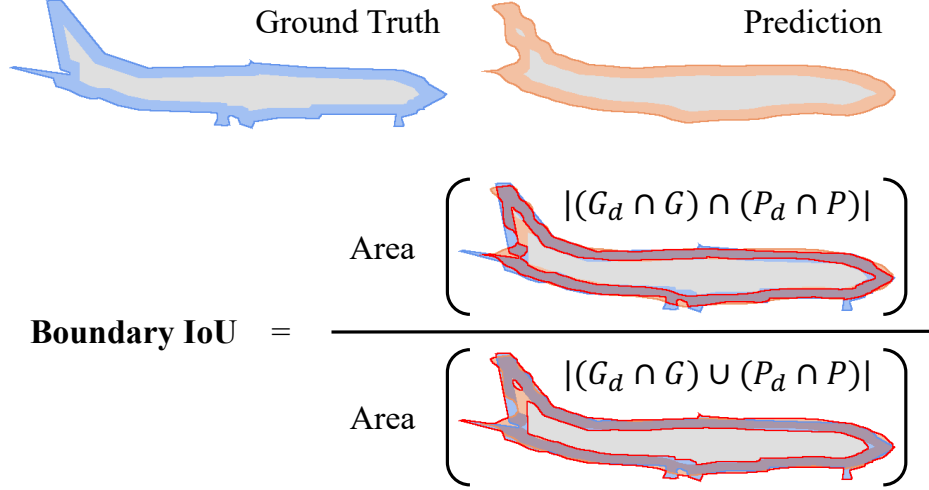


Figure 5.5: **Boundary IoU computation illustration.** (top) Ground truth and prediction masks. For both, we highlight *mask* pixels that are within distance d from the contours. (bottom) Boundary IoU segmentation consistency measures computes the intersection-over-union between the highlighted regions. In this example, Boundary IoU is 0.73, whereas Mask IoU is 0.91.

contour misalignments that can be attributed to ambiguity. While robustness to ambiguity is good in principle, in Figure 5.4c we observe that F-measure can be nearly discontinuous, rapidly stepping from 1 to 0 when the error severity changes by a small amount. Sharp response curves can lead to task metrics with high variance. In comparison, Mask IoU is more continuous. Further, d may be large relative to small objects, causing F-measure to award significant errors a perfect score.

Discussion. Given the limitations presented above, we conclude that neither Trimap IoU nor F-measure can replace Mask IoU as the main segmentation consistency measure for a broad range of evaluation metrics. At the same time, Mask IoU is biased towards large objects in a way that discourages improvements to boundary segmentations. Next, we propose *Boundary IoU* as a new measure to evaluate segmentation boundary-quality that does not have any of the previously mentioning limitations.

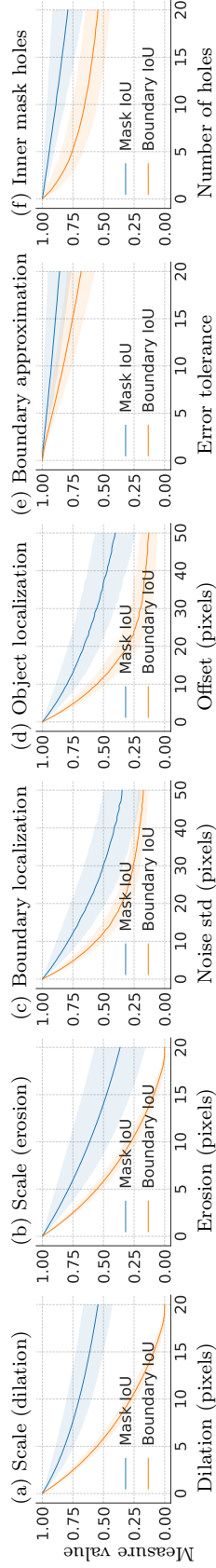


Figure 5.6: Boundary IoU sensitivity curves *across error severities*. We use pseudo-predictions for objects with area $> 96^2$. For each error type, Boundary IoU makes better use of the 0-1 value range demonstrating an improved ability to differentiate between error severity.

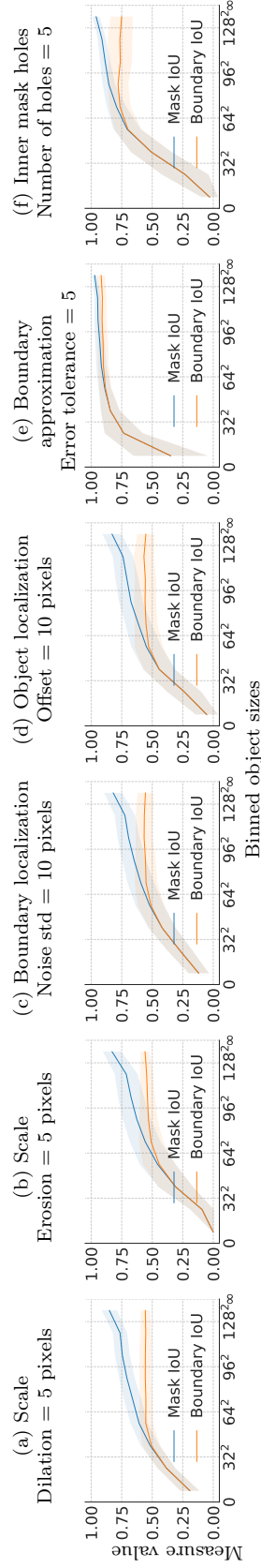


Figure 5.7: Boundary IoU sensitivity curves *across object sizes*. We use pseudo-predictions for objects of different sizes with *fixed* error severities. Objects are binned by their area with 16^2 increment. For larger objects, Boundary IoU remains flat given the fixed error severity, while Mask IoU demonstrates a clear preferential bias for large objects. For small objects, both metrics have similar curves indicating that neither over-penalizes small objects w.r.t. to the other.

5.5 Boundary IoU

In this section we introduce a new segmentation measure and compare it with existing consistency measures using simulated errors. The new measure should have a weaker bias toward large objects than Mask IoU. Furthermore, we aim for a measure that neither over-penalizes nor ignores errors in small objects similarly to Mask IoU.

Guided by these principles, we propose the *Boundary IoU* segmentation consistency measure. The new measure is simultaneously simple and satisfies the principles charted above. Given two masks G and P , Boundary IoU first computes the set of the original masks' pixels that are within distance d from each contour, and then computes the intersection-over-union of these two sets (see Figure 5.5). Using the notation from Table 5.2:

$$\text{Boundary IoU}(G, P) = \frac{|(G_d \cap G) \cap (P_d \cap P)|}{|(G_d \cap G) \cup (P_d \cap P)|}, \quad (5.1)$$

where boundary regions G_d and P_d are the sets of all pixels within d pixels distance from the ground truth and prediction contours respectively. Note, that the new measure evaluates only *mask* pixels that are within pixel distance d from the contours. A simpler version with IoU calculated directly for boundary regions G_d and P_d loses information about sharp contour corners that are smoothed by considering all pixels within distance d from the contours.

The distance to the contour parameter d in Boundary IoU controls the sensitivity of the measure. With d large enough to include all pixels inside the prediction and ground truth masks, Boundary IoU becomes equivalent to Mask IoU. With a smaller parameter d Boundary IoU ignores interior mask pixels, which makes it more sensitive to the boundary quality than Mask IoU for larger objects. For smaller objects, Boundary IoU is very close or even equivalent to Mask IoU depending on the parameter d which prevents it from over-penalizing errors in smaller objects where the number of inner pixels is comparable with the number of pixels close to the contours.

In Figure 5.6 and Figure 5.7 we present the results of our analysis for Boundary IoU. The analysis shows that Boundary IoU is less biased than Mask IoU towards large object sizes across all considered error types (Figure 5.6). Varying object sizes while keeping error severities constant (Fig-

ure 5.7), Boundary IoU behaves identically to Mask IoU for smaller objects avoiding over-penalization for any error type. For larger objects Boundary IoU reduces the bias that Mask IoU exhibits and its value grows more slowly with the object area given fixed error severities.

Comparison with Trimap IoU. We note that the new measure appears quite similar to Trimap IoU (see Table 5.1). However, unlike Trimap IoU, Boundary IoU considers pixels close to the contours of both prediction and ground truth together. This simple change remedies two main limitations of Trimap IoU. The new measure is symmetric and penalizes the errors that appear away from the ground truth boundary (see Figure 5.6 (a), (b), and (f)).

Comparison with F-measure. F-measure uses hard matching between the contours of the predicted and ground truth masks. If the distance between contour pixels is within d pixels, then both precision and recall are perfect, but once the distance goes beyond d the matching does not happen at all. In contrast, Boundary IoU evaluates consistency in a soft manner. Intersection over union is 1.0 if two contours are perfectly aligned and as the contours diverge Boundary IoU gradually decreases. In Section 5.8, we also compare Boundary IoU with a soft generalization of F-measure that averages multiple scores across different parameters d . Our analysis shows that it under-penalizes errors in small objects in comparison with Mask IoU and Boundary IoU.

The pixel distance parameter d . If d is large enough Boundary IoU is equivalent to Mask IoU. On the other hand, if d is too small, Boundary IoU severely penalizes even the smallest misalignment ignoring possible ambiguity of the contours. To select d that does not over-penalize possible ambiguity of the contours, we use Boundary IoU to measure the consistency of two expert annotators who independently delineated the same objects. The creators of LVIS [159] have collected such expert annotations for images in COCO [11] and ADE20k [94] datasets. Both datasets have images of similar resolution and we find that median Boundary IoU between the annotations of the two experts exceeds 0.9 for both datasets when d equals 2% of an image diagonal (15 pixels distance on average). For Cityscapes [12] that has higher resolution images and excellent annotation quality we suggest to use the same distance in pixels which results in d set to 0.5% of an image diagonal for the dataset.

For other datasets, we suggest two considerations for selecting the pixel

distance d : (1) the annotation consistency sets the lower bound on d , and (2) d should be selected based on the performance of current methods and decreased as performance improves.

Limitations of Boundary IoU. The new measure does not evaluate mask pixels that are further than d pixels away from the corresponding ground truth or prediction boundary. Therefore, it can award a perfect score to non-identical masks. For example, Boundary IoU is perfect for a disc-shaped mask and a ring-shaped mask that has the same center and outer radius as the disk, plus the inner radius that is exactly d pixels smaller than the outer one (we show this example in Section 5.8). For these two masks, all non-matching pixels of the disc-shaped mask are further than d pixels away from its boundary. To penalize such cases, we suggest a simple combination of Mask IoU and Boundary IoU by taking their minimum. In our experiments with both real and simulated predictions, we found that Boundary IoU is smaller than or equal to Mask IoU in the absolute majority of cases (99.9%) and the inequality is violated when a prediction with accurate boundaries misses interior part of an object (similar to the toy example above).

5.6 Applications

The most common evaluation metrics for instance and panoptic segmentation tasks are Average Precision (AP or Mask AP) [11] and Panoptic Quality (PQ or Mask PQ) [4] respectively. Both metrics use Mask IoU and inherit its bias toward large objects and, subsequently, the insensitivity to the boundary quality observed before in [139, 170].

We update the evaluation metrics for these tasks by replacing Mask IoU with $\min(\text{Mask IoU}, \text{Boundary IoU})$ as suggested in the previous section. We name the new evaluation metrics Boundary AP and Boundary PQ. The change is simple to implement and we demonstrate that the new metrics are more sensitive to the boundary quality while able to track other types of improvements in predictions as well.

We hope that adoption of the new evaluation metrics will allow rapid progress of boundary quality in segmentation methods. We present our results for instance segmentation in the main text, and we refer the reader to Section 5.8 for our analysis of Boundary PQ.

Table 5.3: Boundary AP and Mask AP on COCO val set for synthetic 28×28 predictions generated from the ground truth. Unlike Mask AP_L , Boundary AP_L successfully captures the lack of fidelity in the synthetic prediction for large objects (area $> 96^2$).

Evaluation metric	AP	AP_S	AP_M	AP_L
Mask AP	96.5	98.9	95.7	95.0
Boundary AP	85.9	98.9	93.0	73.0

Boundary AP for instance segmentation. The goal of the instance segmentation task is to delineate each object with a pixel-level mask. An evaluation metric for the task is simultaneously assessing multiple aspects such as categorization, localization, and segmentation quality. To compare different evaluation metrics, we conduct our experiments with both synthetic predictions and real instance segmentation models. Synthetic predictions allow us to assess the segmentation quality aspect in isolation, whereas real predictions provide insights into the ability of Boundary AP to track all aspects of the instance segmentation task.

We compare Mask AP and Boundary AP on the COCO instance segmentation dataset [11] in the main text. In addition, our findings are supported by similar experiments on Cityscapes [12] and LVIS [159] presented in Section 5.8. Detailed description of all datasets can be found in Section 5.8, along with Boundary AP evaluation for various recent and classic models on all three datasets. These results can be used as a reference to simplify the comparison for future methods.

5.6.1 Evaluation on Synthetic Predictions

Using synthetic predictions we evaluate the segmentation quality aspect of instance segmentation in isolation without a bias that any particular model can have. We simulate predictions by capping the effective resolution of each mask. First, we downscale cropped ground truth masks to a 28×28 resolution¹ mask with continuous values, we then upscale it back using bilinear interpolation, and finally binarize it. Such synthetic masks are close to the ground truth masks for smaller objects, however the discrepancy grows with object size. In Table 5.3 we report overall AP and AP_S , AP_M , and AP_L

¹This is a popular prediction resolution used in practice [5].

Table 5.4: Comparison of Mask AP and Boundary AP on COCO val set for different instance segmentation models **fed with ground truth boxes unless specified otherwise**. Using ground truth boxes disentangles segmentation errors from localization and classification errors.

(a) AP of Mask R-CNN **with ground truth boxes**. Mask R-CNN makes blobby predictions with large defects around boundaries for large objects (see Figure 2.1). Boundary AP successfully captures these errors with much lower AP_L than Mask AP.

Method	Evaluation metric	AP	AP_S	AP_M	AP_L
Mask R-CNN	Mask AP	52.5	44.9	55.0	66.0
Mask R-CNN	Boundary AP	36.1	44.8	46.5	25.9

(c) PointRend (with either 28×28 or 224×224 output resolution) is designed to give higher-quality output masks than Mask R-CNN (which has 28×28 output resolution). Boundary AP captures these improvements well, especially for the higher-resolution output variant of PointRend and for AP_L .

Method	AP^{mask}	AP^{boundary}	AP_S^{boundary}	AP_M^{boundary}	AP_L^{boundary}
Mask R-CNN	52.5	36.1	44.8	46.5	25.9
PointRend-28	57.0 (+4.5)	41.6 (+5.5)	48.2 (+3.4)	52.3 (+5.8)	33.3 (+7.4)
PointRend-224	57.2 (+4.7)	42.1 (+6.0)	48.3 (+3.5)	52.5 (+6.0)	34.4 (+8.5)

(b) Larger backbones do not improve segmentation quality significantly (*i.e.*, Boundary AP stays roughly the same when ground truth boxes are used). With real box predictions, Boundary AP tracks the categorization and localization improvements similarly to Mask AP.

Method	Backbone	Ground truth boxes		Predicted boxes	
		AP^{mask}	AP^{boundary}	AP^{mask}	AP^{boundary}
Mask R-CNN	R50	53.6	37.7	37.2	23.1
Mask R-CNN	R101	53.8 (+0.2)	38.2 (+0.5)	38.6 (+1.4)	24.5 (+1.4)
Mask R-CNN	X101	53.4 (-0.2)	38.1 (+0.4)	39.5 (+2.3)	25.4 (+2.3)

(d) Boundary-preserving Mask R-CNN (BMask R-CNN) which uses 28×28 output *vs.* PointRend which uses 224×224 output. The Boundary AP metric reveals that BMask R-CNN outperforms PointRend for small objects but trails it for large objects where the high output resolution of PointRend improves boundary quality.

Method	AP^{mask}	AP^{boundary}	AP_S^{boundary}	AP_M^{boundary}	AP_L^{boundary}
Mask R-CNN	52.5	36.1	44.8	46.5	25.9
PointRend	57.2 (+4.7)	42.1 (+6.0)	48.3 (+3.5)	52.5 (+6.0)	34.4 (+8.5)
BMask R-CNN	57.4 (+4.9)	42.3 (+6.2)	48.7 (+4.1)	52.7 (+6.2)	33.9 (+8.0)

for object size splits defined in COCO [11]. Mask AP follows the behavior of Mask IoU, showing little sensitivity to the error growth between AP_S and AP_L . In contrast, Boundary AP successfully captures the difference with significantly lower score for larger objects. In Section 5.8, we provide an example of the synthetic predictions and more results using different effective resolutions.

5.6.2 Evaluation on Real Predictions

We use outputs of existing segmentation models to further study Boundary AP. Unless specified, to isolate the segmentation quality from categorization and localization errors for purposes of analysis, we supply ground truth boxes to these methods and assign a random confidence score to each box. We use Detectron2 [78] with a ResNet-50 [77] backbone unless otherwise specified.

Mask AP vs. Boundary AP. Table 5.4a shows both Mask AP and Boundary AP for the standard Mask R-CNN model [5]. Mask R-CNN is well-known to predict blobby masks with significant visual defects for larger objects (see Figure 2.1). Nevertheless, Mask AP_L is larger than Mask AP_S . In contrast, we observe that Boundary AP_L is smaller than Boundary AP_S for Mask R-CNN suggesting that the new measure is more sensitive to the boundary quality of the large objects. Note that in this experiment the use of ground truth boxes removes any categorization and localization errors that are usually larger for small objects.

Segmentation vs. categorization and localization. A general evaluation metric for instance segmentation should track the improvements in all aspects of the task including segmentation, categorization, and localizations. In Table 5.4b we first evaluate Mask R-CNN with several backbones (ResNet-50, ResNet-101, and ResNeXt-101-32 $\times$ 8d [75]), again supplying ground truth boxes. Note that both Mask AP and Boundary AP do not change significantly with different backbones, suggesting that more powerful backbones do not directly influence the segmentation quality. Next, we evaluate Boundary AP requiring each model to predict its own boxes as is standard. We observe that Boundary AP is able to track improvements from better localization and categorization similarly to Mask AP.

Mask quality improvements. We explore Boundary AP’s ability to cap-

ture the improvements in segmentation quality by the methods designed for this purpose in Tables 5.4c and 5.4d. To compare the segmentation quality aspect across models we again supply ground truth boxes to each model.

PointRend [139] was developed to improve pixel-level prediction quality of models like Mask R-CNN and can produce predictions of varying resolution. PointRend significantly improves mask quality; while this can be measured via mask AP, it is more pronounced in Boundary AP, especially for large objects and for a higher resolution PointRend variant. See Table 5.4c for details.

Boundary-preserving Mask R-CNN [160] (BMask R-CNN) improves boundary quality by adding a direct boundary supervision loss and increasing the resolution of feature maps used in its mask head. In Table 5.4d Boundary AP shows that BMask R-CNN with its 28×28 output resolution outperforms PointRend for small objects, whereas for larger objects the 224×224 resolution output of PointRend is preferable, which matches a subjective visual quality assessment (see an example in Figure 2.1). We hope that the improved sensitivity of the new Boundary AP metric will lead to a rapid progress in the methods that improve boundary quality for instance segmentation.

5.7 Conclusion

Unlike the standard Mask IoU, Boundary IoU segmentation quality measure provides a clear, quantitative gradient that rewards improvements to boundary segmentation quality. We hope that the new measure will challenge our community to develop new methods with high-fidelity mask predictions. In addition, Boundary IoU allows a more granular analysis of segmentation-related errors for the complex multifaceted tasks like instance and panoptic segmentation. Incorporation of the measure in the performance analysis tools like TIDE [171] can provide better insights into specific error types of instance segmentation models.

5.8 Supplementary Materials

5.8.1 Additional Measure Analysis

Trimap IoU computes IoU for a band around the ground truth boundary and therefore ignores errors away from the ground truth boundary (*e.g.* inner mask prediction errors). We generate pseudo-predictions with such errors by adding holes of random shapes to ground truth masks. In Figure 5.8 we show that Trimap IoU penalizes inner mask prediction errors less than Mask IoU.

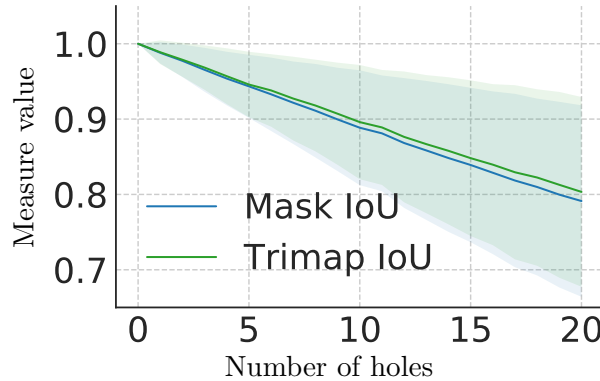


Figure 5.8: **Sensitivity analysis for Trimap IoU.** The measure penalizes inner mask errors less than Mask IoU.

F-measure matches the pixels of the predicted and ground truth contours if they are within the pixels distance threshold d . In the experiments presented in the main text we observe that this strategy makes F-measure ignore scale type errors for smaller objects. The Mean F-measure (mF-measure) modification ameliorates this limitation by averaging several F-measures with different threshold parameters d . Figure 5.9 demonstrates the sensitivity curves of this measure for the scale (dilation) error type. For mF-measure we use d from 0.1% to 2.1% image diagonal with 0.4% increment (from 1 pixel to 17 pixels on average) to compare it with Boundary IoU that uses single d set to 2% image diagonal. We observe that mF-measure behaves similarly to Boundary IoU for large objects, however it under-penalizes errors in small objects where Boundary IoU matches Mask IoU behavior. Furthermore, mF-measure is substantially slower than Boundary IoU as it requires to perform the matching of prediction/ground truth pairs several times for different thresholds d .

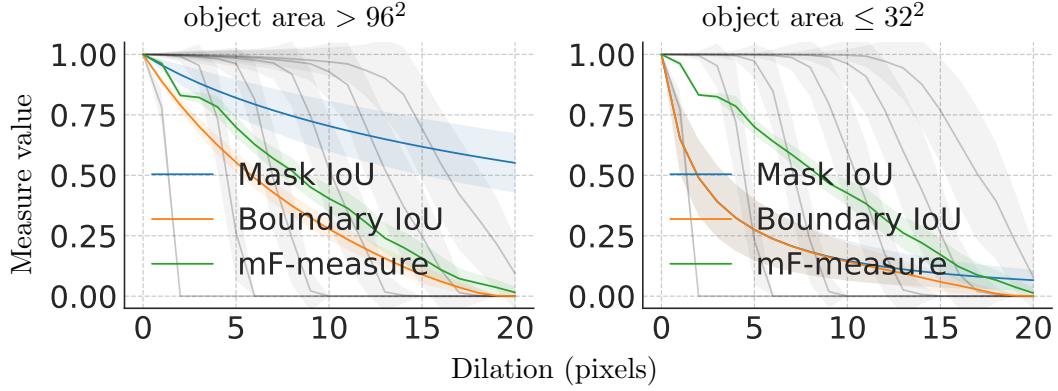


Figure 5.9: **Sensitivity analysis for mean F-measure (mF-measure).** The measure under-penalizes scale type (dilation) errors in small objects in comparison with Boundary IoU that matches Mask IoU behavior for such objects. F-measure curves for different threshold parameters d are shown in gray.

Boundary IoU can award a perfect score for two non-identical masks (see Figure 5.10). As discussed in the main text, we observe that Boundary IoU is smaller or equal to Mask IoU in the absolute majority of cases and the inequality is violated when prediction misses interior part of an object (similar to the toy example in Figure 5.10). To mitigate this limitation, we propose a simple combination of Mask IoU and Boundary IoU by taking their minimum for real-world segmentation evaluation metrics.

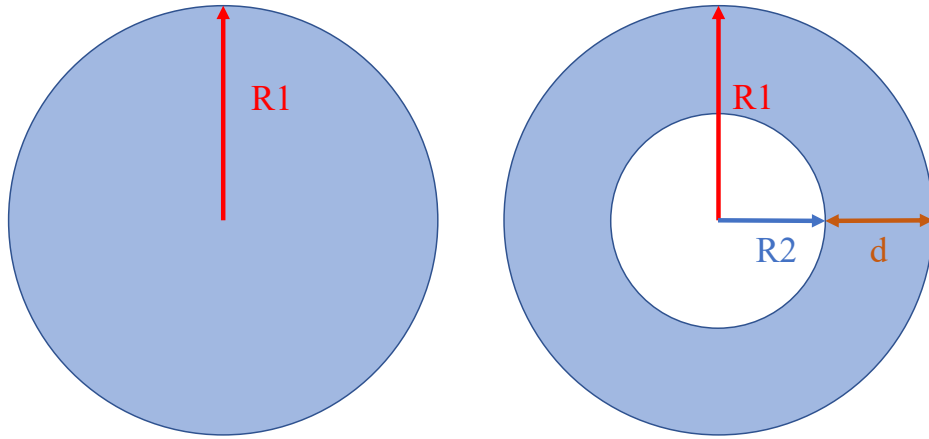


Figure 5.10: Boundary IoU gives a perfect score for two non-identical masks: a disc mask and a ring mask that has the same center and outer radius as the disk, plus the inner radius that is exactly d pixels smaller than the outer one.

Table 5.5: Boundary AP and Mask AP on COCO val set, LVIS*v0.5 val set and Cityscapes val set for synthetic 28×28 , 56×56 , and 112×112 predictions generated from the ground truth. Unlike Mask AP, Boundary AP_L successfully captures the lack of fidelity in the synthetic prediction with lower effective resolution for large objects that have area $> 96^2$.

Mask resolution	Evaluation metric	COCO [11]				LVIS*v0.5 [159]				Cityscapes [12]			
		AP	AP _S	AP _M	AP _L	AP	AP _S	AP _M	AP _L	AP	AP _S	AP _M	AP _L
28×28	Mask AP	96.5	98.9	95.7	95.1	94.3	96.7	93.7	93.9	93.5	98.4	91.6	91.4
	Boundary AP	85.9	98.9	93.0	73.0	85.5	96.7	91.4	73.1	75.9	98.4	86.4	55.9
56×56	Mask AP	99.5	99.8	99.4	99.3	98.2	98.4	99.0	98.9	98.9	99.7	99.0	98.7
	Boundary AP	95.2	99.8	99.3	89.5	94.6	98.4	98.8	89.9	90.9	99.7	97.7	80.5
112×112	Mask AP	99.9	100.0	99.9	99.9	98.8	98.5	99.7	99.9	99.8	99.9	99.9	99.9
	Boundary AP	99.0	100.0	99.9	97.9	98.1	98.5	99.7	98.3	97.2	99.9	99.9	94.9

5.8.2 Application

5.8.2.1 Instance segmentation

Datasets. We evaluate instance segmentation on three datasets: COCO [11], LVIS [159] and Cityscapes [12].

COCO [11] is the most popular instance segmentation benchmark for common objects. It contains 80 categories. There are 118k images for training, 5k images for validation and 20k images for testing.

LVIS [159] is a federated dataset with more than 1000 categories. It shares the same set of images as COCO but the dataset has higher quality ground truth masks. We use LVISv0.5 version of the dataset. Following [139], we construct the LVIS*v0.5 dataset which keeps only the 80 COCO categories from LVISv0.5. LVIS*v0.5 allows us to compare models trained on COCO using higher quality mask annotations from LVIS (*i.e.* AP* in [139]).

Cityscapes [12] is a street-scene high-resolution dataset. There are 5k images annotated with high quality pixel-level annotations and 8 classes with instance-level segmentation.

Evaluation on synthetic predictions. We simulate predictions by capping the effective resolution of each mask. First, we downscale cropped ground truth masks to a fixed resolution mask with continuous values, we then upscale it back using bilinear interpolation, and finally binarize it. Figure 5.11 show visualizations of the synthetic predictions with different effective resolutions. In Table 5.5 we compare Mask AP and Boundary AP for the synthetic predictions with different synthetic scales across different datasets.

Evaluation on real predictions. In addition to the experiments with COCO in the main text, we evaluate Mask R-CNN [5], PointRend [139], and Boundary-preserving Mask R-CNN (BMask R-CNN) [160] on LVIS*v0.5 and Cityscapes in Table 5.6. For each method we feed ground truth boxes to isolate the segmentation quality aspect of the instance segmentation task. On all datasets we observe that Boundary AP better captures improvements in the mask quality.

Reference Boundary AP evaluation. We provide Boundary AP evaluation for various recent and classic models on COCO (Table 5.7), LVIS (Table 5.8), and Cityscapes (Table 5.9) datasets. We do not train any models ourselves and use the Detectron2 framework [78] or official implementations

instead. These results can be used as a reference to simplify the comparison for future methods.

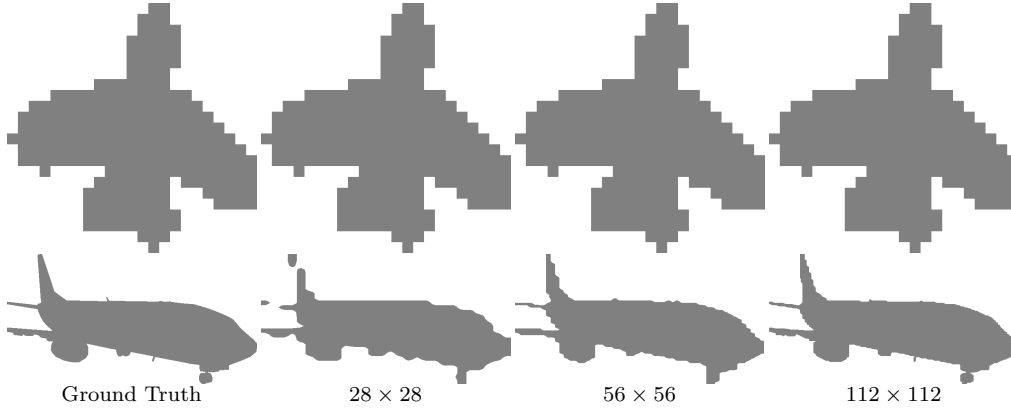


Figure 5.11: **Synthetic predictions visualization.** First, we downscale cropped ground truth masks to a fixed resolution (from 28×28 to 112×112) mask with continuous values, we then upscale it back using bilinear interpolation, and finally binarize it. Synthetic prediction with low effective resolution are close to the ground truth masks for smaller objects (top row), however the discrepancy grows with object size (bottom row).

Table 5.6: Mask R-CNN comparison with the methods designed to improve the mask quality. All models are fed with ground truth boxes. Boundary AP better captures improvements in the mask quality. BMask R-CNN, which outputs 28×28 resolution predictions, outperforms PointRend for smaller objects but trails it for large objects where 224×224 output resolution of PointRend improves boundary quality.

(a) The models are trained on COCO and evaluated on LVIS*v0.5 val set, which has higher annotation quality.

Method	AP^{mask}	AP^{boundary}	AP_S^{boundary}	AP_M^{boundary}	AP_L^{boundary}
Mask R-CNN	51.5	38.3	45.4	48.7	29.2
PointRend	56.8 (+5.3)	45.9 (+7.6)	49.6 (+4.2)	56.0 (+7.3)	42.2 (+13.0)
BMask R-CNN	57.8 (+6.3)	46.1 (+7.8)	50.8 (+5.4)	56.3 (+7.6)	40.7 (+11.5)

(b) The models are trained and evaluated on Cityscapes.

Method	AP^{mask}	AP^{boundary}	AP_S^{boundary}	AP_M^{boundary}	AP_L^{boundary}
Mask R-CNN	35.5	16.4	22.3	22.0	9.7
PointRend	42.2 (+6.7)	23.6 (+7.2)	29.9 (+7.6)	29.0 (+7.0)	19.8 (+10.1)
BMask R-CNN	43.3 (+7.8)	24.0 (+7.6)	36.0 (+13.7)	29.4 (+7.4)	18.7 (+9.0)

Table 5.7: Boundary AP for recent and classic models on COCO val. All models are based on Detectron2. LRS: learning rate schedule, a $1\times$ learning rate schedule refers to 90,000 iterations and a $3\times$ learning rate schedule refers to 270,000 iterations, with batch size 16.

Name	Backbone	LRS	Mask AP					Boundary AP						
			AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
Mask R-CNN [5]	R50 [77]	1×	35.2	56.3	37.5	17.2	37.7	50.3	21.2	46.4	16.8	17.1	31.6	19.7
	R50 [77]	3×	37.2	58.6	39.9	18.6	39.5	53.3	23.1	49.6	19.0	18.6	33.4	22.2
	R101 [77]	3×	38.6	60.4	41.3	19.5	41.3	55.3	24.5	51.7	20.3	19.4	35.0	23.9
	X101-32×8d [75]	3×	39.5	61.7	42.6	20.7	42.0	56.5	25.4	53.2	21.0	20.6	35.8	24.7
Mask R-CNN with deformable conv. [152]	R50 [77]	1×	37.5	59.4	40.2	18.4	39.7	54.8	22.8	49.6	18.1	18.3	33.4	22.1
	R50 [77]	3×	38.5	60.8	41.1	19.7	40.6	55.7	24.1	51.8	19.4	19.6	34.4	23.4
Mask R-CNN with cascade box head [133]	R50 [77]	1×	36.4	56.9	39.2	17.5	38.7	52.5	22.5	47.9	18.7	17.5	32.6	21.7
	R50 [77]	3×	38.5	59.6	41.5	19.5	41.1	54.5	24.5	51.2	20.8	19.5	34.8	23.6
PointRend [139]	R50 [77]	1×	36.2	56.6	38.6	17.1	38.8	52.5	23.5	48.4	20.2	17.1	33.0	24.1
	R50 [77]	3×	38.3	59.1	41.1	19.1	40.7	55.8	25.4	51.3	22.3	19.1	34.8	26.4
	R101 [77]	3×	40.1	61.1	43.0	20.0	42.9	58.6	27.0	54.1	24.2	19.9	37.0	28.7
	X101-32×8d [75]	3×	41.1	62.8	44.2	21.5	43.8	59.1	28.0	55.6	25.3	21.5	37.8	29.1
BMask R-CNN [160]	R50 [77]	1×	36.6	56.7	39.4	17.3	38.8	53.8	23.5	48.4	20.2	17.2	33.0	24.5
	R50 [77]	3×	38.6	59.2	41.7	19.6	41.1	55.7	25.4	51.4	22.3	19.5	35.2	26.3

Table 5.8: Boundary AP of Mask R-CNN baselines on LVISv0.5 val. All models are from the Detectron2 model zoo.

Backbone	Mask AP									Boundary AP								
	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L	AP _r	AP _c	AP _f	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L	AP _r	AP _c	AP _f
R50 [77]	24.4	37.7	26.0	16.7	31.2	41.2	16.0	24.0	28.3	18.8	33.9	18.0	16.7	28.0	19.3	11.9	18.2	22.3
R101 [77]	25.8	39.7	27.3	17.6	33.0	43.7	15.5	26.0	29.6	20.1	35.2	19.8	17.6	29.9	20.8	11.8	20.1	23.5
X101-32×8d [75]	27.0	41.4	28.7	19.0	35.1	43.7	15.4	27.3	31.3	21.4	37.6	21.2	19.0	32.0	21.7	11.2	21.6	25.1

Table 5.9: Boundary AP evaluation on Cityscapes `val` set for models implemented in Detectron2 [78]. Note that we set the dilation width to 0.5% image diagonal for Cityscapes.

Method	Backbone	Mask AP		Boundary AP	
		AP	AP ₅₀	AP	AP ₅₀
Mask R-CNN [5]	R50 [77]	33.8	61.5	11.4	37.4
PointRend [139]	R50 [77]	35.9	61.8	16.7	47.2
BMask R-CNN [160]	R50 [77]	36.2	62.6	15.7	46.2
Panoptic-DeepLab [6]	X71 [70]	35.3	57.9	16.5	47.7

5.8.2.2 Panoptic segmentation

The standard evaluation metric for panoptic segmentation is panoptic quality (PQ or Mask PQ) [4], defined as:

$$\text{PQ} = \underbrace{\frac{\sum_{(p,g) \in \text{TP}} \text{IoU}(p, g)}{|\text{TP}|}}_{\text{segmentation quality (SQ)}} \times \underbrace{\frac{|\text{TP}|}{|\text{TP}| + \frac{1}{2}|\text{FP}| + \frac{1}{2}|\text{FN}|}}_{\text{recognition quality (RQ)}}$$

Mask IoU is presented in two places: (1) calculating the average Mask IoU for true positives in Segmentation Quality (SQ) component and (2) matching prediction and ground truth masks to split them into true positives, false positives, and false negatives. Similarly to Boundary AP, we replace Mask IoU with **min(Mask IoU, Boundary IoU)** in both places and refer the new metric as Boundary PQ.

Datasets. We use two popular datasets with panoptic annotation: COCO panoptic [4] and Cityscapes [12].

COCO panoptic [4] combines annotations from COCO instance segmentation [11] and COCO stuff segmentation [95] into a unified panoptic format with no overlaps. COCO panoptic has 80 things and 53 stuff categories.

Cityscapes [12] has 8 thing and 11 stuff categories.

Similar to the instance segmentation task, we set dilation width to 2% image diagonal for COCO panoptic and 0.5% image diagonal for Cityscapes.

Analysis with synthetic predictions. Following our experimental setup for instance segmentation, we evaluate Boundary PQ on low-fidelity synthetic predictions generated from ground truth annotations to avoid any potential bias toward a specific model. The synthetic predictions are generated by downscaling ground truth panoptic segmentation maps for each image and then upscaling it back using nearest neighbor interpolation in both cases.

This image-level generation process ensures a unified treatment of both things and stuff segments following the idea behind the panoptic segmentation task.

In Table 5.10, we report Panoptic Quality and its two components: Segmentation Quality (SQ) and Recognition Quality (RQ) for synthetic predictions with various downscaling ratios across different datasets. Similar to our findings for AP, Boundary PQ better tracks boundary quality improvements than Mask PQ for panoptic segmentation. Furthermore, we find that the difference between Boundary PQ and Mask PQ is mainly caused by the difference in SQ. This observation confirms that Boundary IoU better tracks the mask quality of predictions and does not significantly change other aspects like the matching procedure between prediction and ground truth segments.

Table 5.10: Boundary PQ and Mask PQ evaluated on COCO panoptic **val** and Cityscapes **val** sets for synthetic prediction with 8, 4, and 2 downscaling ratios generated from the ground truth. Boundary PQ is more sensitive than Mask PQ in its Segmentation Quality (SQ) component while the Recognition Quality (RQ) component is comparable.

Downscaling ratio	Evaluation metric	COCO panoptic [4]			Cityscapes [12]		
		PQ	SQ	RQ	PQ	SQ	RQ
8	Mask PQ	62.6	78.5	78.4	66.3	77.8	83.7
	Boundary PQ	52.8	68.1	77.0	47.1	58.6	80.2
4	Mask PQ	81.0	85.9	93.7	84.3	85.7	98.2
	Boundary PQ	76.6	81.4	93.7	75.0	76.3	98.2
2	Mask PQ	92.5	93.4	99.0	94.2	94.2	99.9
	Boundary PQ	90.8	91.6	99.0	90.7	90.7	99.9

References Boundary PQ evaluation. We provide Boundary PQ evaluation for various models on COCO panoptic and Cityscapes datasets in Table 5.11. We do not train any models ourselves and use models trained by their authors. These results can be used as a reference to simplify the comparison for future methods.

Table 5.11: Reference Boundary PQ evaluation for various models on COCO panoptic val and Cityscapes val.

Dataset	Method	Backbone	Mask PQ			Boundary PQ		
			PQ	SQ	RQ	PQ	SQ	RQ
COCO panoptic [4]	Panoptic FPN [15]	R50 [77]	41.5	79.1	50.5	30.8	70.0	41.7
		R101 [77]	43.0	80.0	52.1	32.5	70.9	43.7
		X101-32×8d [75]	44.4	80.4	53.8	33.9	71.4	45.5
Cityscapes [12]	UPSNNet [14] DETR [91]	R50 [77]	42.5	78.2	52.4	31.0	68.7	43.3
		R50 [77]	43.4	79.3	53.8	32.8	71.0	45.2
	UPSNNet [14]	R50 [77]	59.4	79.7	73.1	33.4	63.1	51.9
Cityscapes [12]	Panoptic-DeepLab [6]	R50 [77]	59.8	80.0	73.5	36.3	64.3	55.6
		X71 [70]	63.0	81.7	76.2	41.0	65.5	61.7

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

In summary, in this dissertation, we demonstrate that it is possible to use a universal architecture to address multiple image segmentation tasks, *i.e.*, universal image segmentation. We explore universal image segmentation with both per-pixel classification and mask classification. Although per-pixel classification has a simpler form, we find it to suffer from recognizing regions from a large number classes. In contrast, we find mask classification is the right way to go and show that mask classification can successfully address multiple image segmentation tasks.

Then, we also investigate how to evaluate image segmentation models. We find the popular Mask IoU measure has limited sensitivity to errors along object boundaries, especially for larger objects, and we propose a new Boundary IoU which focuses more on the boundary regions to address this issue. We believe boundary quality is important as we design more accurate models. However, there is no perfect metric and the evaluation should depend on downstream tasks because there exist multiple partitions or segmentations of the same image.

6.2 Future Work

Although our work on universal image segmentation has shown state-of-the-art results on multiple benchmarks, we believe it is not the end of the story and image segmentation is still far from being solved. Here, we list three of the many remaining problems that we think are important and interesting to work on.

How to segment any modality beyond 2D images? 2D images are only one of the modalities in computer vision and there are other modalities like videos and 3D point clouds which are also very important. How can we generalize our image segmentation model to these modalities? We believe segmentation is about grouping and thus different tasks or modalities are not intrinsically different. We think mask classification naturally unifies even more general grouping tasks and our universal architecture can be generalized to address segmentation problems in other modalities. We have actually shown that our Mask2Former can outperform previous state-of-the-art on video instance segmentation without any modification. And we believe Mask2Former can also be used for 3D segmentation.

How to segment everything without naming it? Till now our image segmentation models have only been evaluated with objects from a small set of closed vocabulary. For example, COCO instance segmentation only cares about 80 categories. In real applications, we usually require the model to segment arbitrary objects. Thus, we are also interested in designing a model that can segment everything and we can get the desired output by querying the model with user-input of various types:

- We can query the desired mask with clicks; *e.g.*, a user clicks on the four extreme points of an object and the model will return the mask. This is often referred to as interactive segmentation.
- We can query the model with examples of segmentation masks; *e.g.*, we give an example mask of a bicycle or a horse to the network and the model should segment this specific object in other images as well. This is often referred to as few shot or one shot segmentation.
- We can query the model with natural language; *e.g.*, the model predicts different masks for different input prompts. This is often referred to as the referring expression segmentation.

How to segment everything and name it as well? We can take the direction discussed in the preceding paragraph one step further. If we have a model that can segment everything, can we even ask the model to name everything as well? This recently introduced problem is usually referred to as open-vocabulary segmentation, where the model should partition an image with arbitrary vocabulary.

Once we have a model that is able to address these problems well, it means we have eventually solved segmentation. And such a model can be naturally used for any segmentation problem. To better summarize the capability of our imaginary segmentation system, we will describe it as a segmentation “foundation model”. A segmentation foundation model is a pre-trained model for general grouping or segmentation which can be used to solve any segmentation problems via either zero-shot transfer or fine-tuning with very few examples.

In the longer term, we believe such effort will eventually lead to a unified computer vision system that can address as many problems as possible in a standardized manner.

REFERENCES

- [1] M. Wertheimer, “Untersuchungen zur lehre von der gestalt,” *Psychologische forschung*, 1923. 1
- [2] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The PASCAL visual object classes challenge: A retrospective,” *IJCV*, 2015. 1, 35, 41, 69, 100
- [3] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, “Simultaneous detection and segmentation,” in *ECCV*, 2014. 2, 33, 36
- [4] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, “Panoptic segmentation,” in *CVPR*, 2019. 2, 5, 7, 35, 36, 40, 41, 44, 53, 62, 63, 69, 73, 80, 99, 100, 113, 125, 126, 127
- [5] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” in *ICCV*, 2017. 2, 5, 7, 8, 20, 29, 33, 36, 37, 38, 47, 60, 63, 64, 68, 71, 72, 73, 83, 97, 98, 114, 116, 121, 123, 125
- [6] B. Cheng, M. D. Collins, Y. Zhu, T. Liu, T. S. Huang, H. Adam, and L.-C. Chen, “Panoptic-DeepLab: A simple, strong, and fast baseline for bottom-up panoptic segmentation,” in *CVPR*, 2020. 2, 42, 54, 63, 79, 86, 87, 88, 125, 127
- [7] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, “Rethinking atrous convolution for semantic image segmentation,” *arXiv:1706.05587*, 2017. 3, 8, 35, 42, 45, 63, 65
- [8] B. Cheng, A. G. Schwing, and A. Kirillov, “Per-pixel classification is not all you need for semantic segmentation,” in *NeurIPS*, 2021. 3, 60, 61, 62, 63, 64, 66, 68, 69, 70, 71, 72, 73, 74, 80, 81, 82, 84, 85, 87, 88, 89, 90, 91, 93
- [9] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar, “Masked-attention mask transformer for universal image segmentation,” in *CVPR*, 2022. 4
- [10] B. Cheng, R. Girshick, P. Dollár, A. C. Berg, and A. Kirillov, “Boundary IoU: Improving object-centric image segmentation evaluation,” in *CVPR*, 2021. 4, 71, 73, 83

- [11] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: Common objects in context,” in *ECCV*, 2014. 5, 6, 7, 12, 35, 41, 52, 53, 62, 69, 73, 80, 97, 99, 100, 102, 112, 113, 114, 116, 120, 121, 125
- [12] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The Cityscapes dataset for semantic urban scene understanding,” in *CVPR*, 2016. 5, 6, 7, 12, 34, 41, 52, 62, 69, 77, 84, 99, 100, 112, 114, 120, 121, 125, 126, 127
- [13] G. Neuhold, T. Ollmann, S. Rota Bulò, and P. Kotschieder, “The mapillary vistas dataset for semantic understanding of street scenes,” in *CVPR*, 2017. 5, 6, 7, 12, 34, 41, 52, 62, 69, 77, 84
- [14] Y. Xiong, R. Liao, H. Zhao, R. Hu, M. Bai, E. Yumer, and R. Urtasun, “UPSNet: A unified panoptic segmentation network,” in *CVPR*, 2019. 5, 8, 14, 15, 19, 20, 21, 23, 29, 63, 127
- [15] A. Kirillov, R. Girshick, K. He, and P. Dollár, “Panoptic feature pyramid networks,” in *CVPR*, 2019. 5, 8, 14, 19, 20, 28, 29, 63, 75, 127
- [16] J. Li, A. Raventos, A. Bhargava, T. Tagawa, and A. Gaidon, “Learning to fuse things and stuff,” *arXiv:1812.01192*, 2018. 5, 8, 14, 15, 18, 20
- [17] Y. Li, X. Chen, Z. Zhu, L. Xie, G. Huang, D. Du, and X. Wang, “Attention-guided unified network for panoptic segmentation,” *arXiv:1812.03904*, 2018. 5, 8, 14, 19, 20, 29
- [18] L. Porzi, S. R. Bulò, A. Colovic, and P. Kotschieder, “Seamless scene segmentation,” in *CVPR*, 2019. 5, 8, 14, 15, 18
- [19] T.-J. Yang, M. D. Collins, Y. Zhu, J.-J. Hwang, T. Liu, X. Zhang, V. Sze, G. Papandreou, and L.-C. Chen, “DeeperLab: Single-shot image parser,” *arXiv:1902.05093*, 2019. 5, 6, 8, 10, 11, 14, 15, 18, 19, 20, 21, 23, 29
- [20] N. Gao, Y. Shan, Y. Wang, X. Zhao, Y. Yu, M. Yang, and K. Huang, “SSAP: Single-shot instance segmentation with affinity pyramid,” in *ICCV*, 2019. 5, 8, 12, 14, 15, 19, 20
- [21] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder-decoder with atrous separable convolution for semantic image segmentation,” in *ECCV*, 2018. 7, 10, 14, 15, 17, 25, 28, 33, 42, 44, 45, 55, 60
- [22] D. H. Ballard, “Generalizing the Hough transform to detect arbitrary shapes,” *Pattern Recognition*, 1981. 7, 8, 10

- [23] A. Kendall, Y. Gal, and R. Cipolla, “Multi-task learning using uncertainty to weigh losses for scene geometry and semantics,” in *CVPR*, 2018. 7, 8, 10
- [24] H. Liu¹, C. Peng, C. Yu, J. Wang, X. Liu, G. Yu, and W. Jiang, “An end-to-end network for panoptic segmentation,” in *CVPR*, 2019. 8
- [25] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” in *CVPR*, 2017. 8, 20, 42, 62, 63, 69, 75
- [26] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs,” *TPAMI*, 2018. 8, 35, 42, 43, 63, 65, 89, 98, 100, 102, 106
- [27] K. Sofiiuk, O. Barinova, and A. Konushin, “AdaptIS: Adaptive instance selection network,” in *ICCV*, 2019. 8, 14, 15, 18, 19, 20
- [28] Y. Liu, S. Yang, B. Li, W. Zhou, J. Xu, H. Li, and Y. Lu, “Affinity derivation and graph merge for instance segmentation,” in *ECCV*, 2018. 8
- [29] M. Keuper, E. Levinkov, N. Bonneel, G. Lavoué, T. Brox, and B. Andres, “Efficient decomposition of image and mesh graphs by lifted multicuts,” in *ICCV*, 2015. 8
- [30] Z. Zhang, A. G. Schwing, S. Fidler, and R. Urtasun, “Monocular object instance segmentation and depth ordering with cnns,” in *ICCV*, 2015. 8
- [31] J. Uhrig, M. Cordts, U. Franke, and T. Brox, “Pixel-level encoding and depth layering for instance-level semantic labeling,” in *GCPR*, 2016. 8
- [32] Z. Zhang, S. Fidler, and R. Urtasun, “Instance-level segmentation for autonomous driving with deep densely connected MRFs,” in *CVPR*, 2016. 8
- [33] M. Bai and R. Urtasun, “Deep watershed transform for instance segmentation,” in *CVPR*, 2017. 8, 12
- [34] S. Liu, J. Jia, S. Fidler, and R. Urtasun, “SGN: Sequential grouping networks for instance segmentation,” in *CVPR*, 2017. 8
- [35] A. Kirillov, E. Levinkov, B. Andres, B. Savchynskyy, and C. Rother, “InstanceCut: From edges to instances with multicut,” in *CVPR*, 2017. 8, 63

- [36] A. Newell, Z. Huang, and J. Deng, “Associative embedding: End-to-end learning for joint detection and grouping,” in *NeurIPS*, 2017. 8, 9
- [37] A. Fathi, Z. Wojna, V. Rathod, P. Wang, H. O. Song, S. Guadarrama, and K. P. Murphy, “Semantic instance segmentation via deep metric learning,” *arXiv:1703.10277*, 2017. 8
- [38] B. De Brabandere, D. Neven, and L. Van Gool, “Semantic instance segmentation with a discriminative loss function,” *arXiv:1708.02551*, 2017. 8
- [39] X. Liang, L. Lin, Y. Wei, X. Shen, J. Yang, and S. Yan, “Proposal-free network for instance-level object segmentation,” *TPAMI*, 2018. 8
- [40] V. Kulikov, V. Yurchenko, and V. Lempitsky, “Instance segmentation by deep coloring,” *arXiv:1807.10007*, 2018. 8
- [41] D. Neven, B. D. Brabandere, M. Proesmans, and L. V. Gool, “Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth,” in *CVPR*, 2019. 8, 10
- [42] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, “YOLACT++: Better real-time instance segmentation,” *TPAMI*, 2020. 8, 63
- [43] B. Leibe, A. Leonardis, and B. Schiele, “Combined object categorization and segmentation with an implicit shape model,” in *Workshop on Statistical Learning in Computer Vision, ECCV*, 2004. 8
- [44] J. Gall and V. Lempitsky, “Class-specific Hough forests for object detection,” in *CVPR*, 2009. 8
- [45] O. Barinova, V. Lempitsky, and P. Kholi, “On detection of multiple object instances using Hough transforms,” *TPAMI*, 2012. 8
- [46] J. Uhrig, E. Rehder, B. Fröhlich, U. Franke, and T. Brox, “Box2pix: Single-shot instance segmentation by assigning pixels to object boxes,” in *IEEE Intelligent Vehicles Symposium (IV)*, 2018. 8
- [47] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, “Optics: ordering points to identify the clustering structure,” in *ACM Sigmod Record*, 1999. 8
- [48] I. Kokkinos, “UberNet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory,” in *CVPR*, 2017. 8

- [49] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool, “Fast scene understanding for autonomous driving,” *arXiv:1708.02550*, 2017. 8
- [50] G. Papandreou, T. Zhu, L.-C. Chen, S. Gidaris, J. Tompson, and K. Murphy, “PersonLab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model,” in *ECCV*, 2018. 8, 15
- [51] H. Law and J. Deng, “CornerNet: Detecting objects as paired keypoints,” in *ECCV*, 2018. 9
- [52] X. Zhou, J. Zhuo, and P. Krähenbühl, “Bottom-up object detection by grouping extreme and center points,” in *CVPR*, 2019. 9
- [53] D. P. Papadopoulos, J. R. Uijlings, F. Keller, and V. Ferrari, “Extreme clicking for efficient object annotation,” in *ICCV*, 2017. 9
- [54] X. Zhou, D. Wang, and P. Krähenbühl, “Objects as points,” *arXiv:1904.07850*, 2019. 9
- [55] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, “CenterNet: Keypoint triplets for object detection,” in *ICCV*, 2019. 9
- [56] B. Cheng, Y. Wei, H. Shi, R. Feris, J. Xiong, and T. Huang, “Revisiting RCNN: On awakening the classification power of Faster RCNN,” in *ECCV*, 2018. 10
- [57] B. Cheng, Y. Wei, H. Shi, R. Feris, J. Xiong, and T. Huang, “Decoupled classification refinement: Hard false positive suppression for object detection,” *arXiv:1810.04002*, 2018. 10
- [58] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *arXiv:1704.04861*, 2017. 10
- [59] Z. Wu, C. Shen, and A. van den Hengel, “Bridging category-level and instance-level semantic image segmentation,” *arXiv:1605.06885*, 2016. 10
- [60] S. R. Bulò, G. Neuhold, and P. Kotschieder, “Loss maxpooling for semantic image segmentation,” in *CVPR*, 2017. 10
- [61] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe, “Full-resolution residual networks for semantic segmentation in street scenes,” in *CVPR*, 2017. 10

- [62] J. J. Tompson, A. Jain, Y. LeCun, and C. Bregler, “Joint training of a convolutional network and a graphical model for human pose estimation,” in *NeurIPS*, 2014. 10, 15
- [63] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *CVPR*, 2016. 12
- [64] X. Li, H. Zhao, L. Han, Y. Tong, and K. Yang, “GFF: Gated fully fusion for semantic segmentation,” *arXiv:1904.01803*, 2019. 15
- [65] Y. Zhu, K. Sapra, F. A. Reda, K. J. Shih, S. Newsam, A. Tao, and B. Catanzaro, “Improving semantic segmentation via video propagation and label relaxation,” in *CVPR*, 2019. 15
- [66] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, “Path aggregation network for instance segmentation,” in *CVPR*, 2018. 15
- [67] W. Liu, A. Rabinovich, and A. C. Berg, “ParseNet: Looking wider to see better,” *arXiv:1506.04579*, 2015. 14
- [68] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *ICML*, 2015. 14
- [69] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *ICLR*, 2015. 14
- [70] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in *CVPR*, 2017. 15, 20, 27, 54, 86, 125, 127
- [71] H. Qi, Z. Zhang, B. Xiao, H. Hu, B. Cheng, Y. Wei, and J. Dai, “Deformable convolutional networks – COCO detection and segmentation challenge 2017 entry,” *ICCV COCO Challenge Workshop*, 2017. 15, 20
- [72] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao, “Deep high-resolution representation learning for visual recognition,” *TPAMI*, 2019. 17, 18, 24, 25, 26, 88
- [73] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei, “Auto-DeepLab: Hierarchical neural architecture search for semantic image segmentation,” in *CVPR*, 2019. 17, 27
- [74] B. Cheng, M. D. Collins, Y. Zhu, T. Liu, T. S. Huang, H. Adam, and L.-C. Chen, “Panoptic-DeepLab,” in *ICCV COCO + Mapillary Joint Recognition Challenge Workshop*, 2019. 17, 24
- [75] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” in *CVPR*, 2017. 19, 116, 123, 124, 127

- [76] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, “Deformable convolutional networks,” in *ICCV*, 2017. 19, 20
- [77] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *CVPR*, 2016. 19, 20, 24, 27, 42, 43, 45, 47, 56, 62, 73, 76, 80, 89, 116, 123, 124, 125, 127
- [78] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, “Detectron2,” <https://github.com/facebookresearch/detectron2>, 2019. 19, 43, 72, 89, 90, 116, 121, 125
- [79] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan et al., “Searching for MobileNetV3,” in *ICCV*, 2019. 20, 23
- [80] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted residuals and linear bottlenecks,” in *CVPR*, 2018. 23
- [81] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski, “A database and evaluation methodology for optical flow,” *IJCV*, 2011. 22
- [82] R. Girshick, “Fast R-CNN,” in *ICCV*, 2015. 20
- [83] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *CVPR*, 2014. 20
- [84] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in *NeurIPS*, 2015. 20, 68
- [85] B. Singh and L. S. Davis, “An analysis of scale invariance in object detection-snip,” *CVPR*, 2018. 20
- [86] Y. Zhu, Y. Tian, D. Metaxas, and P. Dollár, “Semantic amodal segmentation,” in *CVPR*, 2017. 23
- [87] K. Li and J. Malik, “Amodal instance segmentation,” in *ECCV*, 2016. 23
- [88] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *CVPR*, 2009. 24
- [89] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *CVPR*, 2015. 33, 35, 60, 62

- [90] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, “Pyramid scene parsing network,” in *CVPR*, 2017. 33, 35, 42, 63, 65
- [91] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-end object detection with transformers,” in *ECCV*, 2020. 33, 34, 35, 36, 37, 38, 39, 40, 43, 44, 46, 50, 56, 60, 63, 64, 67, 70, 73, 74, 75, 76, 81, 90, 91, 92, 127
- [92] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu, “Semantic segmentation with second-order pooling,” in *ECCV*, 2012. 33, 35
- [93] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *NeurIPS*, 2017. 34, 35, 36, 38, 39, 62, 67
- [94] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, “Scene parsing through ADE20K dataset,” in *CVPR*, 2017. 34, 35, 41, 52, 53, 58, 62, 69, 74, 77, 80, 84, 100, 112
- [95] H. Caesar, J. Uijlings, and V. Ferrari, “COCO-Stuff: Thing and stuff classes in context,” in *CVPR*, 2018. 34, 41, 52, 54, 100, 125
- [96] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, “Swin transformer: Hierarchical vision transformer using shifted windows,” *arXiv:2103.14030*, 2021. 35, 39, 42, 43, 44, 45, 53, 59, 62, 71, 74, 80, 82, 83, 84, 85, 87, 92
- [97] H. Wang, Y. Zhu, H. Adam, A. Yuille, and L.-C. Chen, “MaX-DeepLab: End-to-end panoptic segmentation with mask transformers,” in *CVPR*, 2021. 35, 36, 38, 46, 47, 55, 63, 70, 81, 82
- [98] S. Konishi and A. Yuille, “Statistical cues for domain specific image segmentation with performance analysis,” in *CVPR*, 2000. 35
- [99] D. Comaniciu and P. Meer, “Robust analysis of feature spaces: Color image segmentation,” in *CVPR*, 1997. 35
- [100] J. Shi and J. Malik, “Normalized cuts and image segmentation,” *TPAMI*, 2000. 35
- [101] J. Dai, K. He, and J. Sun, “Convolutional feature masking for joint object and stuff segmentation,” in *CVPR*, 2015. 35
- [102] J. Carreira and C. Sminchisescu, “CPMC: Automatic object segmentation using constrained parametric min-cuts,” *TPAMI*, 2011. 35
- [103] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, “Selective search for object recognition,” *IJCV*, 2013. 35, 77

- [104] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik, “Multiscale combinatorial grouping,” in *CVPR*, 2014. 35, 77
- [105] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu, “Dual attention network for scene segmentation,” in *CVPR*, 2019. 35, 63
- [106] Y. Yuan, L. Huang, J. Guo, C. Zhang, X. Chen, and J. Wang, “OCNet: Object context for semantic segmentation,” *IJCV*, 2021. 35, 63
- [107] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu, “CCNet: Criss-cross attention for semantic segmentation,” in *ICCV*, 2019. 35, 42, 63
- [108] X. Wang, R. Girshick, A. Gupta, and K. He, “Non-local neural networks,” in *CVPR*, 2018. 35, 63
- [109] S. Zheng, J. Lu, H. Zhao, X. Zhu, Z. Luo, Y. Wang, Y. Fu, J. Feng, T. Xiang, P. H. Torr et al., “Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers,” in *CVPR*, 2021. 35, 39, 44, 53, 63, 79, 85, 86
- [110] R. Strudel, R. Garcia, I. Laptev, and C. Schmid, “Segmenter: Transformer for semantic segmentation,” in *ICCV*, 2021. 35, 39, 61, 63, 86
- [111] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly et al., “An image is worth 16x16 words: Transformers for image recognition at scale,” in *ICLR*, 2021. 35, 45, 79
- [112] Z. Tian, C. Shen, and H. Chen, “Conditional convolutions for instance segmentation,” in *ECCV*, 2020. 36, 63
- [113] X. Wang, R. Zhang, T. Kong, L. Li, and C. Shen, “SOLOv2: Dynamic and fast instance segmentation,” *NeurIPS*, 2020. 36, 63
- [114] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” in *ICCV*, 2017. 39, 43, 72
- [115] F. Milletari, N. Navab, and S.-A. Ahmadi, “V-Net: Fully convolutional neural networks for volumetric medical image segmentation,” in *3DV*, 2016. 39, 43, 72
- [116] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, “Scene parsing challenge 2016,” http://sceneparsing.csail.mit.edu/index_challenge.html, 2016. 41, 52, 53
- [117] Y. Yuan, X. Chen, and J. Wang, “Object-contextual representations for semantic segmentation,” in *ECCV*, 2020. 42, 44, 54

- [118] B. Cheng, L.-C. Chen, Y. Wei, Y. Zhu, Z. Huang, J. Xiong, T. S. Huang, W.-M. Hwu, and H. Shi, “SPGNet: Semantic prediction guidance for scene parsing,” in *ICCV*, 2019. 42
- [119] Y. Wu and K. He, “Group normalization,” in *ECCV*, 2018. 42
- [120] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” in *ICLR*, 2019. 43, 72
- [121] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet large scale visual recognition challenge,” *IJCV*, 2015. 43, 80
- [122] “MMSegmentation: OpenMMLab semantic segmentation toolbox and benchmark,” <https://github.com/open-mmlab/mms Segmentation>, 2020. 43
- [123] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun, “Unified perceptual parsing for scene understanding,” in *ECCV*, 2018. 44, 53, 74, 85, 87
- [124] “COCO + Places Challenges 2017,” <https://places-coco2017.github.io/>, 2016. 53
- [125] A. Tao, K. Sapra, and B. Catanzaro, “Hierarchical multi-scale attention for semantic segmentation,” *arXiv:2005.10821*, 2020. 55, 88
- [126] Y. Wu, G. Zhang, Y. Gao, X. Deng, K. Gong, X. Liang, and L. Lin, “Bidirectional graph reasoning network for panoptic segmentation,” in *CVPR*, 2020. 56
- [127] Y. Wu, G. Zhang, H. Xu, X. Liang, and L. Lin, “Auto-panoptic: Co-operative multi-component architecture search for panoptic segmentation,” in *NeurIPS*, 2020. 56
- [128] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, “ShuffleNet V2: Practical guidelines for efficient CNN architecture design,” in *ECCV*, 2018. 56
- [129] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang et al., “Hybrid task cascade for instance segmentation,” in *CVPR*, 2019. 60, 61, 63, 68, 71, 73, 82, 83, 84
- [130] W. Zhang, J. Pang, K. Chen, and C. C. Loy, “K-Net: Towards unified image segmentation,” in *NeurIPS*, 2021. 60, 61, 63, 70, 73, 75, 76, 81, 82
- [131] S. H. Z. Lu, R. Cheng, and C. He, “FaPN: Feature-aligned pyramid network for dense image prediction,” *arXiv*, 2021. 61, 63, 74, 75, 76, 85, 87

- [132] Y. Fang, S. Yang, X. Wang, Y. Li, C. Fang, Y. Shan, B. Feng, and W. Liu, “Instances as queries,” in *ICCV*, 2021. 61, 63, 71, 83, 84
- [133] Z. Cai and N. Vasconcelos, “Cascade R-CNN: Delving into high quality object detection,” in *CVPR*, 2018. 63, 97, 123
- [134] M. Tan, R. Pang, and Q. V. Le, “EfficientDet: Scalable and efficient object detection,” in *CVPR*, 2020. 63, 75, 76
- [135] Y. Li, H. Zhao, X. Qi, Y. Chen, L. Qi, L. Wang, Z. Li, J. Sun, and J. Jia, “Fully convolutional networks for panoptic segmentation with point-based supervision,” *arXiv:2108.07682*, 2021. 63, 79, 82, 86, 88
- [136] Z. Sun, S. Cao, Y. Yang, and K. M. Kitani, “Rethinking transformer-based set prediction for object detection,” in *ICCV*, 2021. 65
- [137] P. Gao, M. Zheng, X. Wang, J. Dai, and H. Li, “Fast convergence of DETR with spatially modulated co-attention,” in *ICCV*, 2021. 65, 75, 76
- [138] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, “Deformable DETR: Deformable transformers for end-to-end object detection,” in *ICLR*, 2021. 67, 69, 74, 75, 76, 85
- [139] A. Kirillov, Y. Wu, K. He, and R. Girshick, “PointRend: Image segmentation as rendering,” in *CVPR*, 2020. 68, 98, 99, 113, 117, 121, 123, 125
- [140] B. Cheng, O. Parkhi, and A. Kirillov, “Pointly-supervised instance segmentation,” *arXiv*, 2021. 68
- [141] G. Ghiasi, Y. Cui, A. Srinivas, R. Qian, T.-Y. Lin, E. D. Cubuk, Q. V. Le, and B. Zoph, “Simple copy-paste is a strong data augmentation method for instance segmentation,” in *CVPR*, 2021. 71, 72, 73, 90
- [142] X. Du, B. Zoph, W.-C. Hung, and T.-Y. Lin, “Simple training strategies and model scaling for object detection,” *arXiv:2107.00057*, 2021. 71, 72, 73, 90
- [143] H. Bao, L. Dong, and F. Wei, “BEiT: BERT pre-training of image transformers,” *arXiv*, 2021. 74, 85, 87
- [144] D. Meng, X. Chen, Z. Fan, G. Zeng, H. Li, Y. Yuan, L. Sun, and J. Wang, “Conditional DETR for fast training convergence,” in *ICCV*, 2021. 76
- [145] L.-C. Chen, H. Wang, and S. Qiao, “Scaling wide residual networks for panoptic segmentation,” *arXiv:2011.11675*, 2020. 79, 86, 87, 88

- [146] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, “SegFormer: Simple and efficient design for semantic segmentation with transformers,” in *NeurIPS*, 2021. 79, 86, 88
- [147] Z. Li, W. Wang, E. Xie, Z. Yu, A. Anandkumar, J. M. Alvarez, T. Lu, and P. Luo, “Panoptic segformer,” *arXiv:2109.03814*, 2021. 81, 82
- [148] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, “PVTv2: Improved baselines with pyramid vision transformer,” *arXiv:2106.13797*, 2021. 81, 82
- [149] M. Liberman, “Reproducible research and the common task method,” <https://www.simonsfoundation.org/event/reproducible-research-and-the-common-task-method>, 2015. 97
- [150] J. Dai, K. He, and J. Sun, “Instance-aware semantic segmentation via multi-task network cascades,” in *CVPR*, 2016. 97
- [151] Z. Li, Y. Ma, Y. Chen, X. Zhang, and J. Sun, “Joint COCO and Mapillary workshop at ICCV 2019: COCO instance segmentation challenge track,” *arXiv:2010.02475*, 2020. 97
- [152] X. Zhu, H. Hu, S. Lin, and J. Dai, “Deformable ConvNets v2: More deformable, better results,” in *CVPR*, 2019. 97, 123
- [153] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, “Soft-NMS—improving object detection with one line of code,” in *ICCV*, 2017. 97
- [154] Z. Huang, L. Huang, Y. Gong, C. Huang, and X. Wang, “Mask scoring R-CNN,” in *CVPR*, 2019. 97
- [155] Z. Li, Y. Zhuang, X. Zhang, G. Yu, and J. Sun, “COCO instance segmentation challenges 2018,” <http://presentations.cocodataset.org/ECCV18/COCO18-Detect-Megvii.pdf>, 2018. 97
- [156] J. Y. Zhang, S. Pepose, H. Joo, D. Ramanan, J. Malik, and A. Kanazawa, “Perceiving 3D human-object spatial arrangements from a single image in the wild,” in *ECCV*, 2020. 97
- [157] A. Sengupta, I. Budvytis, and R. Cipolla, “Synthetic training for accurate 3D human pose and shape estimation in the wild,” in *BMVC*, 2020. 97
- [158] A. Sharma, W. Dong, and M. Kaess, “Compositional scalable object SLAM,” *arXiv:2011.02658*, 2020. 97
- [159] A. Gupta, P. Dollar, and R. Girshick, “LVIS: A dataset for large vocabulary instance segmentation,” in *ICCV*, 2019. 98, 99, 104, 105, 106, 112, 114, 120, 121

- [160] T. Cheng, X. Wang, L. Huang, and W. Liu, “Boundary-preserving Mask R-CNN,” in *ECCV*, 2020. 98, 99, 117, 121, 123, 125
- [161] P. Kohli, P. H. Torr et al., “Robust higher order potentials for enforcing label consistency,” *IJCV*, 2009. 98, 100
- [162] D. R. Martin, *An empirical approach to grouping and segmentation*. PhD dissertation, University of California Berkeley, 2003. 98, 105
- [163] G. Csurka, D. Larlus, F. Perronnin, and F. Meylan, “What is a good evaluation measure for semantic segmentation?” in *BMVC*, 2013. 98, 102
- [164] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung, “A benchmark dataset and evaluation methodology for video object segmentation,” in *CVPR*, 2016. 98, 102, 106
- [165] J. Liang, N. Homayounfar, W.-C. Ma, Y. Xiong, R. Hu, and R. Urtasun, “PolyTransform: Deep polygon transformer for instance segmentation,” in *CVPR*, 2020. 99
- [166] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results,” <http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html>, 2007. 100
- [167] D. R. Martin, C. C. Fowlkes, and J. Malik, “Learning to detect natural image boundaries using local brightness, color, and texture cues,” *TPAMI*, 2004. 102
- [168] D. Marin, Z. He, P. Vajda, P. Chatterjee, S. Tsai, F. Yang, and Y. Boykov, “Efficient segmentation: Learning downsampling near semantic boundaries,” in *ICCV*, 2019. 102
- [169] S. Gillies et al., “Shapely: Manipulation and analysis of geometric objects,” toblerity.org, 2007. [Online]. Available: <https://github.com/Toblerity/Shapely> 103
- [170] Z. Wang, D. Acuna, H. Ling, A. Kar, and S. Fidler, “Object instance annotation with deep extreme level set evolution,” in *CVPR*, 2019. 113
- [171] D. Bolya, S. Foley, J. Hays, and J. Hoffman, “TIDE: A general toolbox for identifying object detection errors,” in *ECCV*, 2020. 117