

**ON MODELING LONG-RANGE DEPENDENCIES  
FOR VISUAL PERCEPTION**

by  
Huiyu Wang

A dissertation submitted to Johns Hopkins University in conformity  
with the requirements for the degree of Doctor of Philosophy

Baltimore, Maryland  
December, 2021

© 2021 Huiyu Wang  
All rights reserved

# Abstract

One of the ultimate goals of computer vision is to extract useful information from visual inputs. An example is to recognize and segment objects from natural images. Recently, deep networks enable us to do a wide range of these tasks better than ever. These are mostly achieved with convolutional neural networks that model pixel relations within a small convolution kernel. Despite such success of convolution, the local window approximation makes it challenging to capture long-range relations. This limitation results in problems, such as unsatisfactory generalization and robustness to out-of-distribution examples.

In this dissertation, I aim to model long-range dependencies in the context of natural image perception. The first part of the dissertation is focused on designing neural architectures that are flexible enough to capture long-range relations. We start by improving convolutional networks with dynamic scaling policies. Then, we explore an alternative solution that completely replaces convolution with global self-attention to capture more context. The attention mechanism is further extended to modeling relations between the pixels and the objects with a transformer, enabling panoptic segmentation in an end-to-end manner. These flexible long-range models usually require a large amount of labeled data to train. In order to address this issue, we discuss self-supervised techniques that learn representation effectively without human annotation in the second part of the dissertation. We regularize the contrastive learning framework with a consistency term that refines self-supervision signals. We also study a more general pretext task, masked image modeling, and train transformers to learn better representations with an online semantic tokenizer.

## **Thesis Readers**

Dr. Alan L. Yuille (Primary Advisor)  
Bloomberg Distinguished Professor  
Department of Cognitive Science & Computer Science  
Johns Hopkins University

Dr. Wei Shen  
Associate Professor  
Artificial Intelligence Institute  
Shanghai Jiao Tong University

Dr. Liang-Chieh Chen  
Research Scientist  
Google Research

*Dedicated to my family  
for their unconditional love and unending support.*

# Acknowledgements

First and foremost, I would like to thank my advisor Alan Yuille. I worked with him as a research assistant before starting the doctoral program at Johns Hopkins University. At that time, I was deeply impressed by his kindness, his probabilistic mind and his thoughts on the limitations of neural networks. I appreciate more of such patient guidance on both high-level research direction and technical derivation during the period of Ph.D. training. I am also grateful for his constant support on my research interest in the topic of neural architectures and a unique two-year part-time internship with Google. During this doctoral journey, I also thank Alan for his enthusiasm and dedication to ambitious research goals; for the inspiration from neuroscience and cognitive science perspectives; for the open and collaborative atmosphere in the lab.

I would also like to thank my thesis committee, Dr. Alan Yuille, Dr. Wei Shen, and Dr. Liang-Chieh Chen from three different time zones around the world. I thank Wei for teaching me the fundamentals of computer vision; for demonstrating how to prepare and give high-quality lectures to graduate students when I was a teaching assistant for the course; for guiding my research on self-supervised learning and micro-batch training. I thank Liang-Chieh for teaching me great research practices including experiment design, paper writing, presentation, *etc.*; for all the command lines you shared and all the brainstorms, discussions, and meetings we had; for advising two of my favorite research projects presented in this dissertation. I am also grateful to Yinzhi Cao, Rama Chellappa, Vishal Patel, Mathias Unberath, and Rene Vidal for serving on my GBO committee.

I feel extremely fortunate to work with amazing people during two internships. I thank Mohammad Rastegari, Aniruddha Kembhavi, and Ali Farhadi from Allen Institute of Artificial Intelligence (AI2) for getting me started with the first computer vision publication which turns out to be an oral presentation at CVPR; for the great non-profit open research atmosphere at AI2; for all the hands-on guidance that one will need for the first publication. For the other internship with Google Research, I thank Liang-Chieh Chen, Yukun Zhu, Bradley Green, and Hartwig Adam for getting me not only impressed by but also prepared for the Google standard research quality and code quality. I also thank both of the internship mentors for the recent support on my job searching.

I am thankful to all members of CCVL during my doctoral journey for all the discussions and parties: Wei Shen, Lingxi Xie, Vittal Premachandran, Ehsan Jahangiri, Yan Wang, Adam Kortylewski, Yongyi Lu, Zongwei Zhou, Jianyu Wang, Weichao Qiu, Chenxi Liu, Zhuotun Zhu, Zhishuai Zhang, Siyuan Qiao, Cihang Xie, Yuyin Zhou, Chenxu Luo, Qing Liu, Qi Chen, Fengze Liu, Yi Zhang, Hongru Zhu, Yingda Xia, Jieru Mei, Qihang Yu, Yingwei Li, Yixiao Zhang, Zhuowan Li, Zihao Xiao, Chenglin Yang, Yutong Bai, Angtian Wang, Chen Wei, Jieneng Chen, Ju He, Prakhar Kaushik, Qihao Liu, Xiaoding Yuan, Kate Sanders, Runtao Liu, and Bowen Li. In addition, I am super grateful to CCVL summer interns advised by me, including Jiteng Mu, Huajin Pi, Jinghao Zhou, and Feng Wang for directly contributing to their amazing summer projects.

Finally, I would like to thank my parents for their unconditional love and unending support. I am grateful to my girlfriend Chen Wei, not only for her company, support, fruitful discussions, but also for contributing substantially to two of the publications presented in this dissertation. In addition, thank you to my cat, Novelty, as well for the encouragement (and for being present in all my research papers).

# Contents

<b>Abstract</b>	ii
<b>Dedication</b>	iv
<b>Acknowledgements</b>	v
<b>Contents</b>	vii
<b>List of Tables</b>	xiii
<b>List of Figures</b>	xix
<b>1 Introduction</b>	1
1.1 Neural Architectures of Long-Range Models . . . . .	3
1.2 Self-Supervised Pre-Training of Long-Range Models . . . . .	4
1.3 Relevant Publications . . . . .	5
<b>I Neural Architectures of Long-Range Models</b>	7
<b>2 ELASTIC: Improving CNNs with Dynamic Scaling Policies</b>	8
2.1 Introduction . . . . .	8
2.2 Related Work . . . . .	11
2.3 Model . . . . .	12
2.3.1 Scale Policy in CNN Blocks . . . . .	13

2.3.2	The ELASTIC Structure . . . . .	14
2.3.3	Augmenting Models with Elastic . . . . .	16
2.4	Experiments . . . . .	17
2.4.1	ImageNet Classification . . . . .	19
2.4.2	Scale Policy Analysis . . . . .	21
2.4.3	MS COCO Multi-Label Classification . . . . .	23
2.4.4	PASCAL VOC Semantic Segmentation . . . . .	26
2.4.5	Ablation Study . . . . .	27
2.5	Conclusion . . . . .	29
<b>3</b>	<b>Axial-DeepLab: Stand-Alone Axial-Attention for Panoptic Segmentation</b>	<b>30</b>
3.1	Introduction . . . . .	30
3.2	Related Work . . . . .	32
3.3	Method . . . . .	34
3.3.1	Position-Sensitive Self-Attention . . . . .	35
3.3.2	Axial-Attention . . . . .	37
3.4	Experimental Results . . . . .	40
3.4.1	ImageNet . . . . .	41
3.4.2	COCO . . . . .	42
3.4.3	Mapillary Vistas . . . . .	44
3.4.4	Cityscapes . . . . .	46
3.4.5	Ablation Studies . . . . .	49
3.5	Conclusion and Discussion . . . . .	50
<b>4</b>	<b>MaX-DeepLab: End-to-End Panoptic Segmentation with Mask Transformers</b>	<b>52</b>
4.1	Introduction . . . . .	52
4.2	Related Work . . . . .	57
4.3	Method . . . . .	58

4.3.1	MaX-DeepLab Formulation . . . . .	58
4.3.2	PQ-Style Loss . . . . .	59
4.3.3	MaX-DeepLab Architecture . . . . .	62
4.3.4	Auxiliary Losses . . . . .	66
4.4	Experiments . . . . .	67
4.4.1	Main Results . . . . .	68
4.4.2	Ablation Study . . . . .	70
4.4.3	Analysis . . . . .	73
4.5	Conclusion . . . . .	75
<b>II</b>	<b>Self-Supervised Pre-Training of Long-Range Models</b>	<b>76</b>
<b>5</b>	<b>CO2: Consistent Contrast for Unsupervised Visual Representation Learning</b>	<b>77</b>
5.1	Introduction . . . . .	77
5.2	Method . . . . .	80
5.2.1	Contrastive Learning . . . . .	80
5.2.2	Consistent Contrast . . . . .	83
5.3	Experiments . . . . .	85
5.3.1	Linear Classification . . . . .	85
5.3.2	Linear Classification . . . . .	86
5.3.3	Semi-Supervised Learning . . . . .	87
5.3.4	Transfer Learning . . . . .	87
5.3.5	Analysis . . . . .	89
5.4	Related Work . . . . .	92
5.5	Discussion . . . . .	94
<b>6</b>	<b>iBOT: Image BERT Pre-Training with Online Tokenizer</b>	<b>95</b>
6.1	Introduction . . . . .	95

6.2	Preliminaries . . . . .	98
6.2.1	Masked Image Modeling as Knowledge Distillation . . . . .	98
6.2.2	Self-Distillation . . . . .	99
6.3	iBOT . . . . .	100
6.3.1	Framework . . . . .	100
6.3.2	Implementation . . . . .	102
6.4	Experiment . . . . .	102
6.4.1	Classification on ImageNet-1K . . . . .	103
6.4.2	Downstream Tasks . . . . .	105
6.4.3	Properties of ViT Trained with MIM . . . . .	107
6.4.4	Robustness . . . . .	109
6.4.5	Ablation Study on Tokenizer . . . . .	110
6.5	Related Work . . . . .	111
6.6	Conclusion . . . . .	112
<b>7</b>	<b>Conclusion</b>	<b>113</b>
7.1	Summary . . . . .	113
7.2	Future Directions . . . . .	114
7.2.1	Dynamic Computation . . . . .	114
7.2.2	Long-Range Modeling for Videos . . . . .	114
7.2.3	Relation Between Vision and Language . . . . .	115
<b>A</b>	<b>ELASTIC: Improving CNNs with Dynamic Scaling Policies</b>	<b>116</b>
A.1	sElastic (simple Elastic) . . . . .	116
A.2	Elastic Architecture Details . . . . .	116
A.3	Semantic Segmentation Results . . . . .	120
<b>B</b>	<b>Axial-DeepLab: Stand-Alone Axial-Attention for Panoptic Segmentation</b>	<b>121</b>
B.1	Runtime . . . . .	121

B.2	Axial-Decoder . . . . .	122
B.3	COCO Visualization . . . . .	123
B.4	Raw Data . . . . .	125
<b>C</b>	<b>MaX-DeepLab: End-to-End Panoptic Segmentation with Mask Transformers</b>	<b>130</b>
C.1	Panoptic Segmentation Results . . . . .	130
C.2	Runtime . . . . .	130
C.3	Mask Output Slot Analysis . . . . .	132
C.4	Mask Head Visualization . . . . .	133
C.5	Transformer Attention Visualization . . . . .	133
C.6	More Technical Details . . . . .	134
<b>D</b>	<b>CO2: Consistent Contrast for Unsupervised Visual Representation Learning</b>	<b>142</b>
D.1	Implementation Details of Contrastive Pre-Training . . . . .	142
D.2	Implemetation Details of Downstream Tasks . . . . .	143
<b>E</b>	<b>iBOT: Image BERT Pre-Training with Online Tokenizer</b>	<b>145</b>
E.1	Pseudocode . . . . .	145
E.2	Multi-Crop . . . . .	145
E.3	Additional Implementations . . . . .	150
E.4	Additional Results . . . . .	153
E.5	Additional Ablations . . . . .	157
E.6	Alternative Tokenizers . . . . .	162
E.7	Visualization . . . . .	163
E.7.1	Pattern Layout . . . . .	163
E.7.2	Self-Attention Visualizations . . . . .	164
E.7.3	Sparse Correspondence. . . . .	164
<b>Bibliography</b>		<b>173</b>



# List of Tables

2.1	<b>Computation in multi-scaling models.</b> This table compares the FLOPs and number of parameters between Elastic and feature/filter pyramid for a single convolutional operation, where the input tensor is $n \times n \times c$ and the filter size is $k \times k$ . $q$ denotes the number of branches in the layer, where $\sum_1^q \frac{1}{b_i} = 1$ and $b_i > 1$ and $r_i > 1$ denote the branching and scaling ratio respectively. Note that the FLOPs and parameters in Elastic is always (under any branching $q$ and scaling ratio $r$ ) lower than or equal to the original model whereas in feature/filter pyramid is higher or equal. . . . .	18
2.2	<b>State-of-the-art model comparisons on ImageNet validation set.</b> Base models (DenseNet, ResNeXt, and DLA) are augmented by Elastic (indicated by '+Elastic'). * indicates our implementation of these models. Note that augmenting with Elastic always improves accuracy across the board. . . . .	20
2.3	<b>MSCOCO multi-class classification.</b> This table shows the generality of our Elastic model by finetuning pre-trained ImageNet models on MSCOCO multi-class images with binary cross entropy loss. Elastic improves F1 scores all across the board. . . . .	25
2.4	F1 scores on small, medium, and large objects respectively. C means per-class F1 and O means overall F1. ResNeXt50 + Elastic improves the most on small objects. . . . .	26

2.5	<b>PASCAL VOC semantic segmentation.</b> This table compares the accuracy of semantic image segmentation (mIOU%) using Elastic models vs. the original model. Elastic models outperform original models by a large margin. This supports that Elastic learns a scale policy that allows processing high-level semantic information and low-level boundary information together. . . . .	27
2.6	<b>Ablation study of up(down)sampling methods.</b> In this table, we show the accuracy of ImageNet classification using Elastic by different choices of up(down)sampling methods. <b>w/ AP</b> indicates average pooling. Our experiment shows Elastic with bilinear up(down)sampling is the best choice with reduced FLOPs. . . . .	28
2.7	<b>Ablation study of high(low) resolution branching rates.</b> In this table, we evaluate different branching rate across high and low-resolutions at each block. We observe that the best trade-off is when we equally divide the branches into high and low-resolutions. Independent of the ratio, all variations of branching are better than the base model. . . . .	28
3.1	ImageNet validation set results. <b>BN:</b> Use batch normalizations in attention layers. <b>PS:</b> Our position-sensitive self-attention. <b>Full:</b> Stand-alone self-attention models without spatial convolutions . . . . .	42
3.2	COCO val set. <b>MS:</b> Multi-scale inputs . . . . .	44
3.3	COCO test-dev set. <b>MS:</b> Multi-scale inputs . . . . .	45
3.4	Mapillary Vistas validation set. <b>MS:</b> Multi-scale inputs . . . . .	46
3.5	Cityscapes val set. <b>MS:</b> Multi-scale inputs. <b>MV:</b> Mapillary Vistas . . . . .	47
3.6	Cityscapes test set. <b>C:</b> Cityscapes coarse annotation. <b>V:</b> Cityscapes video. <b>MV:</b> Mapillary Vistas . . . . .	48
3.7	Ablating self-attention variants on Cityscapes val set. <b>ASPP:</b> Atrous spatial pyramid pooling. <b>PS:</b> Our position-sensitive self-attention . . . . .	49
3.8	Varying axial-attention span on Cityscapes val set . . . . .	50

4.1	Our end-to-end MaX-DeepLab dispenses with these common hand-designed components necessary for existing methods. . . . .	54
4.2	COCO val set. <b>TTA</b> : Test-time augmentation . . . . .	69
4.3	COCO test-dev set. <b>TTA</b> : Test-time augmentation . . . . .	70
4.4	Scaling MaX-DeepLab by using a larger input <b>Resolution</b> , replacing convolutional blocks with <b>Axial</b> -attention blocks, stacking decoder $L$ times, and training with more <b>Iterations</b> . . . . .	71
4.5	Varying transformer <b>P2M</b> feedback attention, <b>M2M</b> self-attention, and the <b>Stride</b> where we apply the transformer. . . . .	71
4.6	Varying the similarity metric <b>sim</b> and whether to apply the auxiliary <b>Instance Discrimination loss</b> , <b>Mask-ID cross-entropy loss</b> or the <b>Semantic segmentation loss</b> . . . . .	73
5.1	Linear classification protocol on ImageNet-1K . . . . .	85
5.2	Top-5 accuracy for semi-supervised learning on ImageNet . . . . .	86
5.3	Transfer learning performance on PASCAL VOC datasets . . . . .	88
5.4	Linear classification accuracy (%) using an end-to-end encoder and with different choices of $\mathcal{L}_{con}$ . The results are summarized as mean and standard deviation over three different runs. . . . .	91
6.1	<b>ImageNet-1K <math>k</math>-NN and linear probing accuracy.</b> $^{\dagger}$ denotes using selective kernel. $^{\ddagger}$ denotes pre-training on ImageNet-22K. . . . .	103
6.2	<b>ImageNet-1K fine-tuning.</b> . . . . .	103
6.3	<b>ImageNet-1K fine-tuning.</b> Pre-trained on ImageNet-22K. . . . .	103
6.4	<b>Semi-supervised learning on ImageNet-1K.</b> 1% and 10% denotes label fraction. SD denotes self-distillation. . . . .	104
6.5	<b>Unsupervised learning on ImageNet-1K.</b> $^{\dagger}$ denotes $k$ -means clustering on frozen features. . . . .	104

<b>6.6 Object detection (Det.) &amp; instance segmentation (ISeg.) on COCO and Semantic segmentation (Seg.) on ADE20K.</b> We report the results of ViT-S/16 (left) and ViT-B/16 (right). Seg. <sup>†</sup> denotes using a linear head for semantic segmentation. . . . .	105
<b>6.7 Transfer learning by fine-tuning pre-trained models on different datasets.</b> We report Top-1 accuracy of ViT-S/16 (left) and ViT-B/16 (right). . . . .	106
<b>6.8 Robustness evaluation of pre-trained models against background change, occlusion, and out-of-distribution examples.</b> . . . . .	109
<b>6.9 Effect of design choices of semantically meaningful tokenization.</b> . . . . .	110
<b>A.1 Error rates for sElastic on the ImageNet validation set.</b> sElastic models with reduced FLOPs already perform better than some of the original models. We also provide the Elastic versions from the original chapter as a reference. . . . .	117
<b>A.2 DLA model architectures.</b> Following DLA, we show our DLA classification architectures in the table. Split32 means a ResNeXt bottleneck with 32 paths while Split50 means a ResNeXt bottleneck with 50 paths. Stages 3 to 6 show d-n where d is the aggregation depth and n is the number of channels. . . . .	118
<b>A.3 ResNeXt50 vs. ResNeXt50+sElastic vs. ResNeXt50+Elastic.</b> ResNeXt50+Elastic employs two resolutions in each block, and keeps output resolution high for more blocks, compared with ResNeXt50. . . . .	118
<b>A.4 DenseNet201 vs. DenseNet201+Elastic.</b> DenseNet+Elastic follows a similar modification as ResNeXt+Elastic, i.e. two resolutions in each block and more blocks in high resolutions. . . . .	119
<b>B.1 Runtime of Axial-ResNet-L on a <math>224 \times 224</math> image</b> . . . . .	122

B.2 Ablating output strides and decoder types on Cityscapes val set. <b>ASPP</b> : Atrous spatial pyramid pooling. <b>OS</b> : Output stride ( <i>i.e.</i> , the ratio of image resolution to final feature resolution in backbone). <b>AD</b> : Use axial-decoder in Axial-DeepLab . . . . .	123
B.3 ImageNet validation set results. <b>Width</b> : the width multiplier that scales the models up. <b>Full</b> : Stand-alone self-attention models without spatial convolutions . . . . .	128
C.1 End-to-end runtime. <b>PQ [val]</b> : PQ (%) on COCO val set. <b>PQ [test]</b> : PQ (%) on COCO test-dev set. . . . .	132
E.1 <i>k</i> -NN performance of iBOT variants. . . . .	147
E.2 Varying multi-crop scale $s$ . . . . .	148
E.3 <i>k</i> -NN and linear probing accuracy on ImageNet-1K without multi-crop augmentation (left) and with multi-crop augmentation (right) multi-crop augmentation. We split the table into results without or with multi-crop augmentation. . . . .	149
E.4 Different fine-tuning recipes. LD: layerwise learning rate decay. DS: mixed-precision training with DeepSpeed. . . . .	150
E.5 Evaluation protocols for semi-supervised learning. <i>Proj.</i> : fine-tuning from the middle layer of the projection head. LR: logistic regression. . . . .	150
E.6 Linear probing on ADE20K semantic segmentation with and without the last LayerNorm [LN]. . . . .	152
E.7 Additional object detection, instance segmentation, and semantic segmentation results with small-size models. We pre-train iBOT with ViT-S/16 for 800 epochs. . . . .	153

<b>E.8 Additional object detection, instance segmentation, and semantic segmentation results with base-size models.</b> We pre-train iBOT with ViT-B/16 for 400 epochs. . . . .	153
<b>E.9 <math>k</math>-NN and linear probing on ImageNet-1K with different pre-training datasets.</b> . . . . .	154
<b>E.10 Effectiveness of pre-trained features on nearest neighbor retrieval.</b> We report the results on different downstream tasks whose evaluation is based on nearest neighbor retrieval. . . . .	154
<b>E.11 Different head sharing strategy.</b> . . . . .	157
<b>E.12 Hard label versus soft label.</b> <i>Cen.</i> : centering. $^t$ : smaller temperature for teacher output. . . . .	157
<b>E.13 Varying iBOT projection head design.</b> . . . . .	158
<b>E.14 Comparing MIM with dense self-distillation.</b> . . . . .	158
<b>E.15 Varying the smoothing momentum for online centering <math>m'</math> and sharpening temperature <math>\tau'_t</math> for the patch tokens.</b> . . . . .	159
<b>E.16 Varying the loss ratio between <math>\mathcal{L}_{\text{CLS}}</math> and <math>\mathcal{L}_{\text{MIM}}</math>.</b> . . . . .	160
<b>E.17 Varying the projection head output dimension.</b> . . . . .	160
<b>E.18 Time and Memory Requirements.</b> We detail the actual training time (T) and GPU memory (Mem.) of different methods, together with their respective linear probing (Lin.) and fine-tuning (Fin.) accuracy. All methods are trained on two 8-GPU V100 machines with a batch size of 1024. . . . .	161
<b>E.19 Methodology comparison over different approaches to tokenize the patches.</b> We report ImageNet-1K $k$ -NN, linear and fine-tuning validation accuracy. Models are pre-trained with ViT-S/16 and 300 epochs. . . . .	162

# List of Figures

1.1	A typical CNN performs convolution in a hierarchical manner. The low-level filters detect part-like patterns (the red boxes) independently. The high-level convolution, operating on a small resolution, applies a high-level template matching and combines the part patterns into a possible giant (the blue box). . . . .	2
1.2	The focus of this dissertation in a typical visual learning pipeline. Part I discusses neural architectures proposed for long-range dependency modeling and Part II presents self-supervised learning methods that train the models to capture long-range dependencies without human annotation. . .	2

2.1	<b>Dynamic scale policy.</b> Scaling policies in CNNs are typically integrated into the network architecture manually in a pyramidal fashion. The color bar in this figure (second row) shows the scales at different blocks of the ResNext50 architecture. The early layers receive <b>eXtra-large</b> resolutions and in the following layers resolutions decrease as <b>Large</b> , <b>Medium</b> , and <b>Small</b> . We argue that scaling policies in CNNs should be instance-specific. Our Elastic model (the third row) allows different scaling policies for different input images and it learns from the training data how to pick the best policy. For scale challenging images e.g. images with lots of small(or diverse scale) objects, it is crucial that network can adapt its scale policy based on the input. As it can be seen in this figure, Elastic gives a better prediction for these scale challenging images. (See section 2.4.2 for more details). . . . .	9
2.2	<b>Multi-scaling model structures.</b> This figure illustrates different approaches to multi-scaling in CNN models and our Elastic model. The solid-line rectangles show the input size and the dashed-line rectangles shows the filter size. . . . .	13
2.3	<b>Left:</b> ResNeXt bottleneck vs. Elastic bottleneck. <b>Right:</b> DenseNet block vs. its equivalent form vs. Elastic block. Elastic blocks spend half of the paths processing downsampled inputs in a low resolution, then the processed features are upsampled and added back to features with the original resolution. Elastic blocks have the same number of parameters and less FLOPs than original blocks. . . . .	16
2.4	<b>Imagenet Accuracy vs. FLOPS and Parameters.</b> This figure shows our Elastic model can achieve a lower error without any extra (or with lower) computational cost. . . . .	19

2.5	<b>Scale policy for complex vs. simple image categories.</b> This figure shows the overall block scale policy score on the entire ImageNet categories. It shows that categories with complex image patterns mostly go through the high-resolution branches in the network and categories with simpler image pattern go through the low-resolution branches. . . . .	22
2.6	<b>Scale policy analysis.</b> This figure shows the impact of the scale policy on the accuracy of our Elastic model. (left) shows all the ImageNet validation set clustered using tsne by their scale policy pattern in the ResNeXt50+Elastic as discussed in section 2.4.2. (middle) shows the the scale policy score of all the images at 17 blocks of the network. Most of the images use high-resolution features at early layers and low-resolution features at later layers but some images break this pattern. Images pointed in the <b>green circle</b> use high-resolution features in the 13 <sup>th</sup> block. Images pointed in the <b>purple circle</b> use low-resolution features in the 4 <sup>th</sup> block. These images usually contain a simpler pattern. (right)-bottom shows the density of images in the tsne space and (right)-top shows the density of the images that got correctly classified by Elastic model but miss-classified by the base ResNeXt model. This shows that Elastic can improve prediction when images are challenging in terms of their scale information. Some samples are pointed by the <b>yellow circle</b> . Best viewed in color. . . . .	24
2.7	<b>Scale stress test</b> on MSCOCO multi-label classification. This bar chart shows the relative F1 improvement of DLA-x60 being augmented Elastic over different image resolutions. Although both models are trained on $224 \times 224$ images, Elastic shows larger improvement when tested on high-resolution images. . . . .	26

3.1 A non-local block (left) <i>vs.</i> our position-sensitive axial-attention applied along the width-axis (right). “ $\otimes$ ” denotes matrix multiplication, and “ $\oplus$ ” denotes element-wise sum. The softmax is performed on the last axis. Blue boxes denote $1 \times 1$ convolutions, and red boxes denote relative positional encoding. The channels $d_{in} = 128$ , $d_q = 8$ , and $d_{out} = 16$ is what we use in the first stage of ResNet after ‘stem’ . . . . .	38
3.2 An axial-attention block, which consists of two axial-attention layers operating along height- and width-axis sequentially. The channels $d_{in} = 128$ , $d_{out} = 16$ is what we use in the first stage of ResNet after ‘stem’. We employ $N = 8$ attention heads . . . . .	39
3.3 Comparing parameters and M-Adds against accuracy on ImageNet classification. Our position-sensitive self-attention (Conv-Stem + PS-Attention) and axial-attention (Conv-Stem + Axial-Attention) consistently outperform ResNet-50 [10], [71] and attention models [71] (both Conv-Stem + Attention, and Full Attention), across a range of network widths ( <i>i.e.</i> , different channels). Our Full Axial-Attention works the best in terms of both parameters and M-Adds . . . . .	43
3.4 Scale stress test on COCO val set. Axial-DeepLab gains the most when tested on extreme resolutions. On the x-axis, ratio 4.0 means inference with resolution $4097 \times 4097$ . . . . .	45
4.1 Our method predicts panoptic segmentation masks <b>directly from images</b> , while previous methods (Panoptic-FPN as an example) rely on <b>a tree of surrogate sub-tasks</b> . Panoptic segmentation masks are obtained by merging semantic and instance segmentation results. Instance segmentation is further decomposed into box detection and box-based segmentation, while box detection is achieved by anchor regression and anchor classification. . . . .	52

4.2 A case study for our method and state-of-the-art <i>box-free</i> and <i>box-based</i> methods. (a) Our end-to-end MaX-DeepLab correctly segments a dog sitting on a chair. (b) Axial-DeepLab [2] relies on a surrogate sub-task of regressing object center offsets [78]. It fails because the centers of the dog and the chair are close to each other. (c) DetectoRS [138] classifies object bounding boxes, instead of masks, as a surrogate sub-task. It filters out the chair mask because the chair bounding box has a low confidence. . . . .	53
4.3 (a) An image and a global memory are fed into a dual-path transformer, which directly predicts a set of masks and classes (residual connections omitted). (b) A dual-path transformer block is equipped with all 4 types of attention between the two paths. . . . .	63
4.4 Training curves for (a) validation PQ, (b) average class confidence, $\hat{p}_{\hat{\sigma}(i)}(c_i)$ , of matched masks, (c) average mask dice, $\text{Dice}(m_i, \hat{m}_{\hat{\sigma}(i)})$ , of matched masks, (d) per-pixel instance discrimination accuracy, and (e) per-pixel mask-ID prediction accuray. . . . .	72
4.5 (b) Pixels of the same instance have similar colors (features), while pixels of different instances have distinct colors. (c) The transformer predicts mask colors (features) and classes. . . . .	74

5.1	Illustration of <b>(a)</b> instance discrimination and <b>(b)</b> our consistency regularization. $\mathbf{q}$ is a query and $\mathbf{p}$ is a positive key. Both are encoded from crops of the same image. $\{\mathbf{n}_k\}_{k=1}^K$ are negative keys, encoded from random crops. In <b>(a)</b> , the similarity is softmax cosine distances between $\mathbf{q}$ and all keys. This similarity is optimized towards an artificial one-hot label which identifies $\mathbf{p}$ among all keys. However, some negatives can be semantically similar but not reflected by the one-hot label (e.g., the one rounded by a red box). In <b>(b)</b> , our proposed consistency regularization encourages the agreement between $P$ , the positive-negative similarity, and $Q$ , the query-negative similarity, reflecting the heterogeneous similarity of the query/positive to the negatives.	81
5.2	Ablation on the effect of hyper-parameters. . . . .	89
5.3	Training curves of ResNet-18 on ImageNet-100. . . . .	90
6.1	<b>Linear probing accuracy on ImageNet.</b> We compare iBOT with other unsupervised baselines. . . . .	96
6.2	<b>Masked image modeling.</b> Given a masked image, the vision transformer learns to predict the output of a visual tokenizer. . . . .	97
6.3	<b>Overview of iBOT framework, performing masked image modeling with an <i>online tokenizer</i>.</b> Given two views $u$ and $v$ of an image $x$ , each view is passed through a teacher network $h_t \circ f_t$ and a student network $h_s \circ f_s$ . iBOT minimizes two losses. The first loss $\mathcal{L}_{[\text{CLS}]}$ is self-distillation between cross-view [CLS] tokens. The second loss $\mathcal{L}_{\text{MIM}}$ is self-distillation between in-view patch tokens, with some tokens masked and replaced by $e_{[\text{MASK}]}$ for the student network. The objective is to reconstruct the masked tokens with the teacher networks' outputs as supervision. . . . .	100
6.4	<b>Pattern layout of patch tokens.</b> Two left figures showcase patterns, <i>headlight of the vehicle</i> and <i>ear of the dog</i> , that share part semantics. Two right figures showcase patterns, <i>stripped</i> and <i>curly surface</i> , that share part textures. . . . .	107

6.5	<b>Part-wise linear probing accuracy.</b> Top- $k$ tokens with the highest attention scores are averaged for classification. . . . .	109
6.6	<b>Visualization for self-attention map.</b> Self-attention map from multiple heads are visualized with different color. . . . .	109
A.1	<b>Semantic segmentation results on PASCAL VOC.</b> Elastic improves most on scale-challenging images. . . . .	120
B.1	An axial-decoder block. We augment an axial-attention block with up-samplings, and encoder features . . . . .	123
B.2	Visualization on COCO val set. Axial-DeepLab shows robustness to occlusion. In row 1 and row 4, Axial-DeepLab captures the occluded left leg and the remote control cable respectively, which are not even present in ground truth labels. In the last row, Axial-DeepLab distinguishes one person occluding another correctly, whereas the ground truth treats them as one instance . . . . .	124
B.3	Attention maps in block 2 of stage 3. Blue pixels are queries that we take, and red pixels indicate the corresponding attention weights. We notice that column head 1 focuses on human heads, while column head 4 correlates with the field. Row head 6 focuses more on local regions whereas column head 5 pools all over the whole image . . . . .	126
B.4	Attention maps in block 3 of stage 4. They focus more on long range context than those in Figure B.3, although all of them have a global receptive field .	127
B.5	Training loss on COCO. Equipped with position-sensitive axial-attention, our Axial-DeepLab fits data distribution better than Panoptic-DeepLab [78], especially on the task of predicting the offset to the object center, which requires precise and long range positional information . . . . .	128
B.6	Scale stress test on COCO val set . . . . .	129

C.1 Comparing MaX-DeepLab with other representative methods on the COCO <i>val</i> set. (Colors modified for better visualization) . . . . .	131
C.2 Failure cases of MaX-DeepLab on the COCO <i>val</i> set. . . . .	135
C.3 The joint distribution for our $N = 128$ mask slots and 133 classes with 80 ‘thing’ classes on the left and 53 ‘stuff’ classes on the right. We observe that a few mask slots predict a lot of the masks. Some mask slots are used less frequently, probably only when there are a lot of objects in one image. Some other slots do not fire at all. In addition, we see automatic functional segregation between ‘thing’ mask slots and ‘stuff’ mask slots, with a few exceptions that can predict both thing and stuff masks. . . . .	136
C.4 The average masks that each mask slot predicts, normalized by image shape. Mask slots are categorized by their total number of firings and sorted from most firings to few firings. We observe spatial clustered patterns, meaning that the mask slots specialize on certain regions of an input image. For example, the most firing mask slot 71, focusing on the center of an image, predicts almost all 80 ‘thing’ classes but ignores ‘stuff’ classes (Figure C.3). The top three categories are tennis rackets, cats, and dogs. The second firing mask slot 106 segments 14 classes of masks on the bottom of an image, such as road, floor, or dining-tables. The third firing mask slot 125 concentrates 99.9% on walls or trees that are usually on the top of an image. The fourth firing mask slot 69 focuses entirely on the person class and predicts 2663 people in the 5000 validation images. . . . .	137

C.5 More visualizations of the decoder feature $g$ with $D = 3$ . Similar to Figure 4.5, we observe a clustering effect of instance colors, <i>i.e.</i> , pixels of the same instance have similar colors (features) while pixels of different instances have distinct colors. Note that in this extreme case of $D = 3$ (that achieves 37.8% PQ), there are not enough colors for all masks, which causes missing objects or artifacts at object boundaries, but these artifacts do not present in our normal setting of $D = 128$ (that achieves 45.7% PQ). . . . .	138
C.6 Visualizing the transformer $M2P$ attention maps for selected predicted masks. We observe that head 2, together with head 5, 7, and 8, mainly attends to the output mask regions. Head 1, 3, and 4 gather more context from broader regions, such as semantically-similar instances (scene 1 head 1) or mask boundaries (scene 2 head 4). In addition, we see that head 6 does not pay much attention to the pixel-path, except for some minor firings on the playing field and on the table. Instead, it focuses more on $M2M$ self-attention which shares the same softmax with $M2P$ attention (Equation (4.14)). . . . .	139
C.7 An example Axial-Block from Axial-DeepLab [2]. This axial-attention bottleneck block consists of two axial-attention layers operating along height- and width-axis sequentially. . . . .	140
C.8 Building blocks for our MaX-DeepLab architectures. . . . .	140

C.9 More detailed MaX-DeepLab architectures. <b>Pretrain</b> labels where we use a classification head to pretrain our models on ImageNet [8]. (a) A dual-path transformer block with C intermediate bottleneck channels. (b) The baseline architecture for our ablation studies in Section 4.4.2. (c) MaX-DeepLab-S that matches the number of parameters and M>Adds of DETR-R101-Panoptic [142]. <b>Axial-Block</b> (Figure C.7) is an axial-attention bottleneck block borrowed from Axial-DeepLab-L [2]. (d) MaX-DeepLab-L that achieves the state-of-the-art performance on COCO [15]. <b>Wide-Axial</b> is a wide version of Axial-Block with doubled intermediate bottleneck channels, similar to the one used in Axial-DeepLab-XL [2]. (The residual connections are dropped for neatness). . . . .	141
E.1 Computation pipelines for iBOT with or without multi-crop augmentation. (a) iBOT w/o multi-crop augmentation. (b), (c), and (d) are three pipelines w/ multi-crop augmentation. (b) does not perform MIM for local crops, whereas (c) performs MIM for all crops. (d) only performs MIM for one of the two global crops. iBOT uses (b) with random MIM. . . . .	147
E.2 Training curves of different multi-crop strategy. . . . .	147
E.3 Robustness against occlusion. Model’s robustness against occlusion with different information loss ratios is studied. 3 patch dropping settings: Random Patch Dropping (left), Salient Patch Dropping (middle), and Non-Salient Patch Dropping (right) are considered. . . . .	156
E.4 Robustness against shuffle. Model’s robustness against shuffle with different grid shuffle sizes is studied. . . . .	156
E.5 Impact of the prediction ratio. $\pm$ denotes to randomly sample from a region.	160
E.6 Impact of the training epochs. Models are ViT-S/16 with multi-crop augmentation. . . . .	160

<b>E.7 Visualization for pattern layout of patch tokens that share high-level semantics.</b> In the first row, we visualize different human-related semantic parts. We observe patterns for <i>human hair</i> , <i>human shoulder &amp; arm</i> , and <i>human elbow</i> respectively. In other rows of the figure, we show semantic parts related to animals ( <i>dog's ear</i> , <i>dog's nose</i> , <i>bird's wing</i> , <i>dragonfly's wing</i> ), outdoor scenes ( <i>front window of the vehicle</i> , <i>window of the architecture</i> ) and indoor objects ( <i>ceiling</i> , <i>glass bottle</i> ). . . . . .	166
<b>E.8 Visualization for pattern layout of patch tokens that share low-level patterns.</b> In the first two rows, we visualize patches that share similar textures. In the third row, we show some shape-related patterns, such as those of lines and similar curvatures. We also notice that some tokens capture color information, as shown in the last row. . . . .	167
<b>E.9 Top-4 representative patches with each of their pattern layout.</b> Order index 0, 1, 2, 3 are ranked according to its self-attention score. In the top-left corner for each pattern layout subfigure, its order index and cluster index are annotated. In the top panel, we can observe that pattern 0,2,3 show explicit semantic information of <i>nose</i> , <i>eyes</i> , <i>ears</i> respectively. Interestingly, patch 1 also locates around the eyes of the <i>Samoyed</i> but its corresponding pattern share visual similarity in shape instead of semantics. This illustrates the diverse behavior for each learned pattern. In the bottom panel, a <i>library</i> is represented by 0 <i>two- or multi-color joints</i> , 1,3 <i>knurling texture</i> , 2 <i>texts</i> . Similarly, we have patterns 0,1,3 focusing more on texture & color and pattern 2 focusing more on semantics. All of these visualized results illustrate versatile behavior for each index. . . . .	168

<b>E.10 Visualization for pattern layout of patch tokens using BEiT (top) and DINO (bottom).</b> In the layout extracted from the DALL-E encoder, we observe minimal semantic patterns. In most cases, patches with similar color (e.g., <i>black area</i> in left figure) or texture (e.g., <i>line</i> in right figure) are clustered. In the layout extracted from DINO, while more complex textures are visible, most patches share similar local details instead of high-level semantics. In the right figure, the semantic part <i>eyes</i> can be somehow observed, yet it is mixed with plenty of irrelevant semantic parts. . . . .	169
<b>E.11 Visualization for pattern layout of [CLS] token.</b> We here indicate the high quality of semantic layout brought by self-distillation of cross-view images on [CLS] token. This property is not brought by MIM and is also prominent in DINO. . . . .	170
<b>E.12 Visualization for self-attention map from Multiple Heads.</b> In the first 8 columns, we showcase iBOT’s attention map along with DINO’s. In the last 10 columns, we showcase more attention map from iBOT. We indicate that iBOT shows visually stronger ability to separate different objects or different parts of one object apart by giving more attentive visualized results for each part, compared with DINO. For example, in the fifth column, there is an attention head in iBOT accounting for the <i>ear of the fox</i> solely, while in DINO, it emerges with other parts; In the eighth column, iBOT separates the <i>mushroom</i> into more semantically meaningful parts. . . . .	171

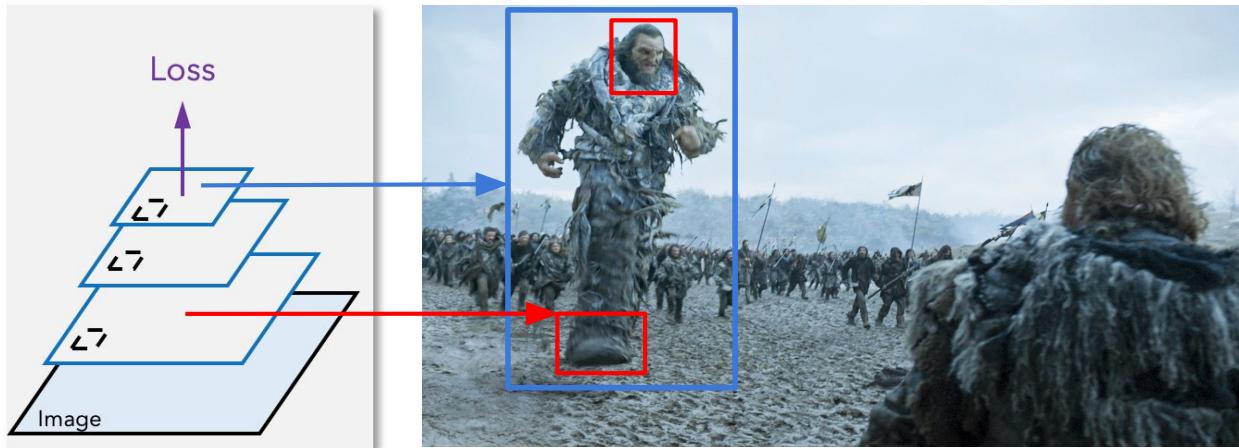


# Chapter 1

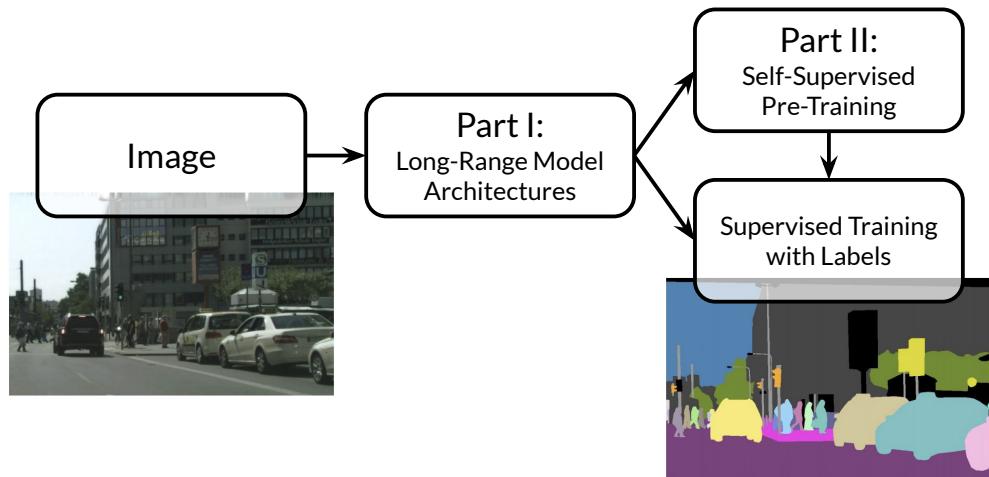
## Introduction

Computer vision seeks to extract useful information from visual input signal which is usually high dimensional and consists of hundreds of thousands of pixels in a 2D natural image. Extracting information from such input requires modeling relations of the input pixels rather than treating them independently as a bag of intensities. Convolution has been used to capture such relations since gradient operators were applied to images in the 1960s. In these early days of computer vision, convolution was usually adopted to extract local relations such as edges with image gradient kernels. In order to model more context in a larger window, one could simply increase the size of the convolution kernel or the template, but the number of templates needed grows dramatically with respect to the size of the window, due to the large variations in natural images. It is infeasible to enumerate all the patterns or even all the images with large templates. Another approach is to stack a hierarchy of local convolutions (Figure 1.1), as is done in almost all of recent convolutional neural networks. However, such mechanism usually ignores the long-range dependencies between two parts, such as the appearance consistencies between the parts, leading to unsatisfactory performance and robustness to out-of-distribution examples.

In this dissertation, I aim to enable neural networks to capture long-range dependencies in the context of 2D image perception. As illustrated in Figure 1.2, this dissertation consists of two parts. In Part I, I will present our efforts at designing long-range modeling



**Figure 1.1.** A typical CNN performs convolution in a hierarchical manner. The low-level filters detect part-like patterns (the red boxes) independently. The high-level convolution, operating on a small resolution, applies a high-level template matching and combines the part patterns into a possible giant (the blue box).



**Figure 1.2.** The focus of this dissertation in a typical visual learning pipeline. Part I discusses neural architectures proposed for long-range dependency modeling and Part II presents self-supervised learning methods that train the models to capture long-range dependencies without human annotation.

architectures with and without convolution. These architectures are more flexible than typical CNNs and thus may require more labeled data to train. In order to address this issue and prevent overfitting of such models, in Part II, I will discuss our self-supervised pre-training methods that learn long-range dependencies from image data without human annotation. The proposed model architectures and training methods in the two parts will be evaluated on supervised learning tasks such as image classification and image segmentation.

## 1.1 Neural Architectures of Long-Range Models

Part I of this dissertation is focused on relaxing the local constraint imposed on convolutional models.

In Chapter 2, we will try to address the long-range modeling issue within the common convolutional neural network framework. Most previous solutions to this issue have a similar theme: a set of intuitive and manually designed policies that are generic and fixed (e.g. SIFT or feature pyramid). We argue that the scaling policy should be learned from data. In this chapter, we introduce ELASTIC, a simple, efficient and yet very effective approach to learn a dynamic scale policy from data. We formulate the scaling policy as a non-linear function inside the network's structure that (a) is learned from data, (b) is instance specific, (c) does not add extra computation, and (d) can be applied on any network architecture. We applied ELASTIC to several state-of-the-art network architectures and showed consistent improvement without extra computation on classification and segmentation tasks. The results show major improvement for images with scale challenges.

In Chapter 3, we will go beyond fully convolutional framework and explore the possibility of adopting self-attention mechanisms in all the layers of a neural network. The vanilla self-attention operation is able to model non-local interactions, but is also extremely expensive to compute on large images, limiting its usage as a stand-alone operator for a

neural network. In this chapter, we attempt to remove the complexity constraint by factorizing 2D self-attention into two 1D self-attentions. This reduces computation complexity and allows performing attention within a larger or even global region. In companion, we also propose a position-sensitive self-attention design. Combining both yields our position-sensitive axial-attention layer, a novel building block that one could stack to form axial-attention models for image classification and dense prediction. We will demonstrate the effectiveness of the model on four large-scale datasets.

In Chapter 4, the attention mechanism is extended from the pixel space to the object mask space. The pixel-to-mask transformer allows direct communication and relation prediction between all pixels and all masks. When the mask transformer is trained with a panoptic segmentation loss function, it is able to predict class-labeled masks directly, enabling end-to-end panoptic segmentation for the first time. This approach simplifies the previous pipeline that depends heavily on surrogate sub-tasks and hand-designed components, such as box detection, non-maximum suppression, thing-stuff merging, *etc.* As a result, our mask transformer closes the gap between box-based and box-free methods for the first time on the challenging COCO dataset by improving a significant 7.1% PQ in the box-free regime.

## 1.2 Self-Supervised Pre-Training of Long-Range Models

Part II of this dissertation discusses training techniques that can be used to extract long-range relations from data without human annotations.

In Chapter 5, we study contrastive learning methods that perform an instance discrimination task: Given a query image crop, label crops from the same image as positives, and crops from other randomly sampled images as negatives. An important limitation of this label assignment is that it can not reflect the heterogeneous similarity of the query crop to crops from other images, but regarding them as equally negative. To address this

issue, inspired by consistency regularization in semi-supervised learning, we propose a consistency term in unsupervised contrastive learning framework. The consistency term takes the similarity of the query crop to crops from other images as “unlabeled”, and the corresponding similarity of a positive crop as a pseudo label. It then encourages consistency between these two similarities and learns better visual representations for downstream tasks as a result.

In Chapter 6, we study masked image modeling as a more promising pretext task for long-range models, or more specifically, for transformers. Taking inspiration from the successful masked language modeling [6], where texts are first tokenized into semantically meaningful pieces, we show the necessity and challenges of adopting a semantically meaningful visual tokenizer. We present a self-supervised framework that can perform masked prediction with an *online tokenizer*. Specifically, we perform self-distillation on masked patch tokens and take the teacher network as the online tokenizer, along with self-distillation on the class token to acquire visual semantics. The online tokenizer is jointly learnable with the MIM objective and dispenses with a multi-stage training pipeline where the tokenizer needs to be pre-trained beforehand. When transformers are trained in this way, they learn more generalizable representations measured by linear probing, fine-tuning, transfer learning, and robustness to out-of-distribution examples.

### 1.3 Relevant Publications

- [1] **Huiyu Wang**, Aniruddha Kembhavi, Ali Farhadi, Alan Yuille, and Mohammad Rastegari, “ELASTIC: Improving cnns with dynamic scaling policies,” in *CVPR*, 2019.
- [2] **Huiyu Wang**, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen, “Axial-DeepLab: Stand-alone axial-attention for panoptic segmentation,” in *ECCV*, 2020.
- [3] **Huiyu Wang**, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen, “MaX-DeepLab: End-to-end panoptic segmentation with mask transformers,” in *CVPR*, 2021.

- [4] Chen Wei, **Huiyu Wang**, Wei Shen, and Alan Yuille, “CO2: Consistent contrast for unsupervised visual representation learning,” in *ICLR*, 2021.
- [5] Jinghao Zhou, Chen Wei, **Huiyu Wang**, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong, “iBOT: Image BERT pre-training with online tokenizer,” *arXiv:2111.07832*, 2021.

## **Part I**

# **Neural Architectures of Long-Range Models**

# Chapter 2

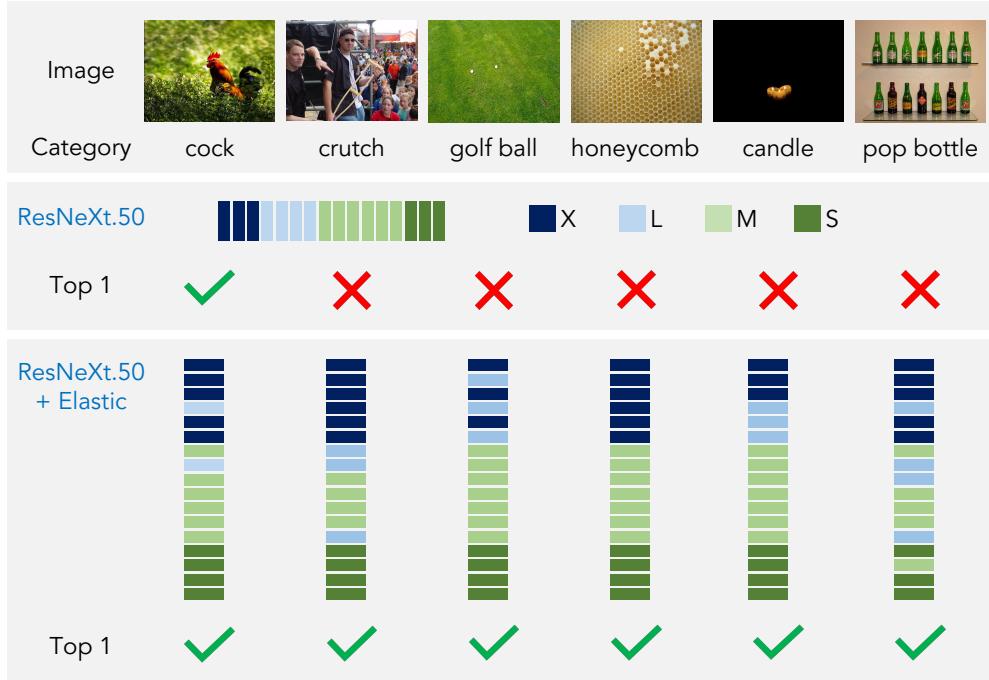
## ELASTIC: Improving CNNs with Dynamic Scaling Policies

This chapter aims to improve CNNs by introducing more context in each layer.

### 2.1 Introduction

Scale variation has been one of the main challenges in computer vision. There is a rich literature on different approaches to encoding scale variations in computer vision algorithms [7]. In feature engineering, there have been manually prescribed solutions that offer scale robustness. For example, the idea of searching for scale first and then extracting features based on a known scale used in SIFT or the idea of using feature pyramids are examples of these prescribed solutions. Some of these ideas have also been migrated to feature learning using deep learning in modern recognition solutions.

The majority of the solutions in old-school and even modern approaches to encode scale are manually designed and fixed solutions. For example, most state-of-the-art image classification networks [8]–[13] use the feature pyramid policy where a network looks at the larger resolution first and then goes to smaller ones as it proceeds through the layers. Despite the fact that this common practice seems to be a natural and intuitive choice, we argue that this scale policy is not necessarily the best one for all possible scale variations



**Figure 2.1. Dynamic scale policy.** Scaling policies in CNNs are typically integrated into the network architecture manually in a pyramidal fashion. The color bar in this figure (second row) shows the scales at different blocks of the ResNext50 architecture. The early layers receive **eXtra-large** resolutions and in the following layers resolutions decrease as **Large**, **Medium**, and **Small**. We argue that scaling policies in CNNs should be instance-specific. Our Elastic model (the third row) allows different scaling policies for different input images and it learns from the training data how to pick the best policy. For scale challenging images e.g. images with lots of small(or diverse scale) objects, it is crucial that network can adapt its scale policy based on the input. As it can be seen in this figure, Elastic gives a better prediction for these scale challenging images. (See section 2.4.2 for more details).

in images. We claim that an ideal scale policy should (1) be learned from the data; (2) be instance specific; (3) not add extra computational burden; and (4) be applicable to any network architecture.

For example, instead of looking at the scales according to the feature pyramid policy if we process the images in Figure 2.1 based on a learned and instance specific policy we see an improved performance. In images with scale challenges like the golf ball image in Figure 2.1 the learned scale policy might differ dramatically from a pyramid policy, resulting in correct classification of that instance. The learned policy for this instance starts from looking at the image from a large scale (dark blue color), and then goes immediately to a smaller scale, and then goes back to a large scale followed by a small scale and so on.

In this chapter, we introduce ELASTIC, an approach to learn instance-specific and not-necessarily-pyramidal scale policies with no extra(or lower) computational cost. Our solution is simple, efficient, and very effective on a wide range of network architectures for image classification and segmentation. Our Elastic model can be applied on any CNN architectures simply by adding downsamplings and upsamplings in parallel branches at each layer and let the network learn from data a scaling policy in which inputs being processed at different resolutions in each layer. We named our model ELASTIC because each layer in the network is flexible in terms of choosing the best scale by a soft policy.

Our experimental evaluations show improvements in image classification on ImageNet[14], multi-label classification on MSCOCO[15], and semantic segmentation on PASCAL VOC for ResNeXt[16], SE-ResNeXt[17], DenseNet[11], and Deep Layer Aggregation (DLA)[12] architectures. Furthermore, our results show major improvements (about 4%) on images with scale challenges (lots of small objects or large variation across scales within the same image) and lower improvements for images without scale challenges. Our qualitative analysis shows that images with similar scaling policies (over the layers of the network) are sharing similar complexity pattern in terms of scales of the objects appearing in the image.

## 2.2 Related Work

The idea behind Elastic is conceptually simple and there are several approaches in the literature using similar concepts. Therefore, we study all the categories of related CNN models and clarify the differences and similarities to our model. There are several approaches to fusing information at different visual resolutions. The majority of them are classified into four categories (depicted in Figure 2.2(b-e)).

**Image pyramid.** An input image is passed through a model multiple times at different resolutions and predictions are made independently at all levels. The final output is computed as an ensemble of outputs from all resolutions. This approach has been a common practice in [18]–[20].

**Loss pyramid.** This method enforces multiple loss functions at different resolutions. [21] uses this approach to improve the utilization of computing resources inside the network. SSD [22] and MS-CNN [23] also use losses at multiple layers of the feature hierarchy.

**Filter pyramid.** Each layer is divided into multiple branches with different filter sizes (typically referred to as the split-transform-merge architecture). The variation in filter sizes results in capturing different scales but with additional parameters and operations. The inception family of networks [21], [24], [25] use this approach. To further reduce the complexity of the filter pyramid [26]–[28] use dilated convolutions to cover a larger receptive field with the same number of FLOPs. In addition, [29] used 2 CNNs to deal with high and low frequencies, and [30] proposed to adaptively choose from 2 CNNs with different capacity.

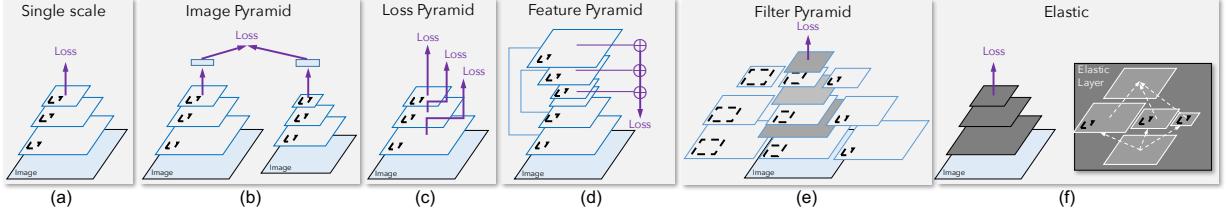
**Feature pyramid.** This is the most common approach to incorporate multiple scales in a CNN architecture. Features from different resolutions are fused in a network by either concatenation or summation. Fully convolutional networks [31] add up the scores from multiple scales to compute the final class score. Hypercolumns [32] use earlier layers in

the network to capture low-level information and describe a pixel in a vector. Several other approaches (HyperNet [33], ParseNet [34], and ION [35]) concatenate the outputs from multiple layers to compute the final output. Several recent methods including SharpMask [36] and U-Net [37] for segmentation, Stacked Hourglass networks [38] for keypoint estimation and Recombinator networks [39] for face detection, have used skip connections to incorporate low-level feature maps on multiple resolutions and semantic levels. [40] extends DenseNet[11] to fuse features across different resolution blocks. Feature pyramid networks (FPNs) [41] are designed to normalize resolution and equalize semantics across the levels of a pyramidal feature resolution hierarchy through top-down and lateral connections. Likewise, DLA [12] proposes an iterative and hierarchical deep aggregation that fuses features from different resolutions.

Elastic resembles models from the Filter pyramid family as well as the Feature pyramid family, in that it introduces parallel branches of computation (a la Filter pyramid) and also fuses information from different scales (a la Feature pyramid). The major difference to the feature pyramid models is that in Elastic every layer in the network considers information at multiple scales uniquely whereas in feature pyramid the information for higher or lower resolution is injected from the other layers. Elastic provides an exponential number of scaling paths across the layers and yet keeps the computational complexity the same (or even lower) as the base model. The major difference to the filter pyramid is that the number of FLOPs to cover a higher receptive field in Elastic is proportionally lower, due to the downsampling whereas in the filter pyramid the FLOPs is higher or the same as the original convolution.

## 2.3 Model

In this section, we elaborate the structure of our proposed Elastic and illustrate standard CNN architectures being augmented with our Elastic. We also contrast our model with



**Figure 2.2. Multi-scaling model structures.** This figure illustrates different approaches to multi-scaling in CNN models and our Elastic model. The solid-line rectangles show the input size and the dashed-line rectangles shows the filter size.

other multi-scale approaches.

### 2.3.1 Scale Policy in CNN Blocks

Formally, a layer in a CNN can be expressed as

$$\mathcal{F}(x) = \sigma \left( \sum_{i=1}^q \mathcal{T}_i(x) \right) \quad (2.1)$$

where  $q$  is the number of branches to be aggregated,  $\mathcal{T}_i(x)$  can be an arbitrary function (normally it is a combination of convolution, batch normalization and activation), and  $\sigma$  are nonlinearities. A few  $\mathcal{F}(x)$  are stacked into a stage to process information in one spatial resolution. Stages with decreasing spatial resolutions are stacked to integrate a pyramid scale policy in the network architecture. A network example of 3 stages with 2 layers in each stage is

$$\mathcal{N} = \mathcal{F}_{32} \circ \mathcal{F}_{31} \circ \mathcal{D}_{r_2} \circ \mathcal{F}_{22} \circ \mathcal{F}_{21} \circ \mathcal{D}_{r_1} \circ \mathcal{F}_{12} \circ \mathcal{F}_{11} \quad (2.2)$$

where  $\mathcal{D}_{r_i}$  indicates the resolution decrease by ratio  $r_i > 1$  after a few layers.  $\mathcal{D}_{r_i}$  can be simply implemented by increasing the stride in the convolution right after. For example, ResNeXt[16] stacks bottleneck layers in each resolution and use convolution with stride 2 to reduce spatial resolution. This leads to a fixed scaling policy that enforces a linear

relationship between number of layers and the effective receptive field of those layers. Parameters of  $\mathcal{T}_i(x)$  and the elements in input tenors  $x$  are all of the tangible ingredients in a CNN that define computational capacity of the model. Under a fixed computational capacity measured by FLOPs, to improve the accuracy of such a model, we can either increase number of parameters in  $\mathcal{T}_i(x)$  and decrease the resolution of  $x$  or increase the resolution of  $x$  and decrease number of parameters in  $\mathcal{T}_i(x)$ . By adjusting the input resolutions at each layer and number of parameters, we can define a scaling policy across the network. We argue that finding the optimal scaling policy (a trade-off between the resolution and number of parameters in each layer) is not trivial. There are several model designs toward increasing the accuracy and manually injecting variations of feature pyramid but most of them are at the cost of higher FLOPs and more parameters in the network. In the next section, we explain our solution that can learn an optimal scaling policy and maintain or reduce number of parameters and FLOPs while improving the accuracy.

### 2.3.2 The ELASTIC Structure

In order to learn image features at different scales, we propose to add down-samplings and up-samplings in parallel branches at each layer and let the network make decision on adjusting its process toward various resolutions at each layer. Networks can learn this policy from training data. We add down-samplings and up-samplings in parallel branches at each layer and divide all the parameters across these branches as follows:

$$\mathcal{F}(x) = \sigma \left( \sum_{i=1}^q \mathcal{U}_{r_i}(\mathcal{T}_i(\mathcal{D}_{r_i}(x))) \right) \quad (2.3)$$

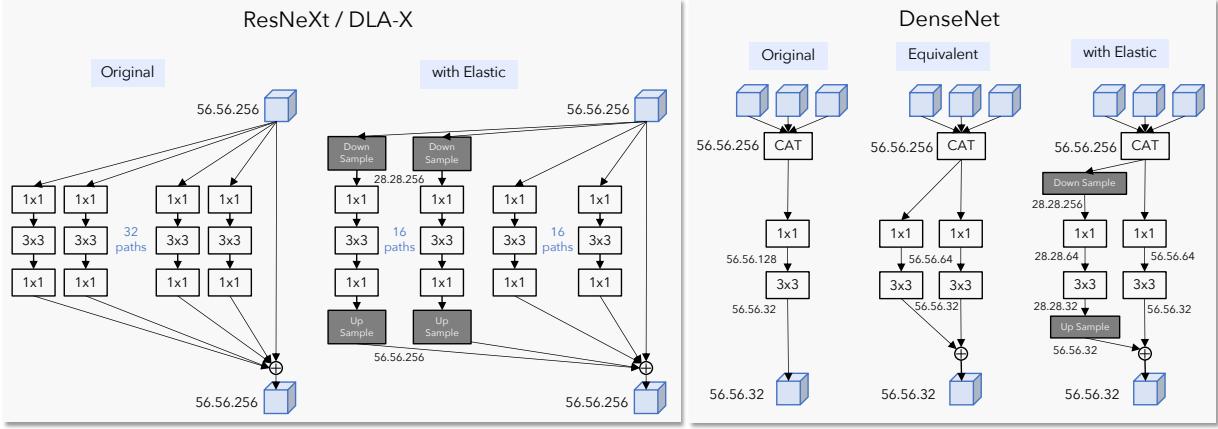
$$\mathcal{N} = \mathcal{F}_{32} \circ \mathcal{F}_{31} \circ \mathcal{F}_{22} \circ \mathcal{F}_{21} \circ \mathcal{F}_{12} \circ \mathcal{F}_{11} \quad (2.4)$$

where  $\mathcal{D}_{r_i}(x)$  and  $\mathcal{U}_{r_i}(x)$  are respectively downsampling and upsampling functions which change spatial resolutions of features in a layer. Unlike in equation 2.2, a few  $\mathcal{F}$  are applied sequentially without downsampling the main stream, and  $\mathcal{N}(x)$  has exactly the same resolution as original  $x$ .

Note that the learned scaling policy in this formulation will be instance-specific i.e. for different image instances, the network may activate branches in different resolutions at each layer. In section 2.4 we show that this instance-specific scaling policy improves prediction on images with scale challenges e.g. images consist of lots of small objects or highly diverse object sizes.

Conceptually, we propose a new structure where information is always kept at a high spatial resolution, and each layer or branch processes information at a lower or equal resolution. In this way we decouple feature processing resolution ( $\mathcal{T}_i$  processes information at different resolutions) from feature storage resolution (the main stream resolution of the network). This encourages the model to process different scales separately at different branches in a layer and thus capture cross-scale information. More interestingly, since we apply Elastic to almost all blocks, the dynamic combination of multiple scaling options at each layer leads to exponentially many different scaling paths. They interpolate between the largest and the smallest possible scale and collectively capture various scales. In fact, this intuition is aligned with our experiments, where we have observed different categories of images adopt different scaling paths (see section 2.4.2). For example, categories with clean and uniform background images mostly choose the low-resolution paths across the network and categories with complex and cluttered objects and background mostly choose the high-resolution paths across the network.

The computational cost of our Elastic model is equal to or lower than the base model, because at each layer the maximum resolution is the original resolution of the input tensor. Low resolution branches reduce the computation and give us extra room for adding more layers to match the computation of the original model.



**Figure 2.3. Left:** ResNeXt bottleneck vs. Elastic bottleneck. **Right:** DenseNet block vs. its equivalent form vs. Elastic block. Elastic blocks spend half of the paths processing downsampled inputs in a low resolution, then the processed features are upsampled and added back to features with the original resolution. Elastic blocks have the same number of parameters and less FLOPs than original blocks.

This simple add-on of downsamplings and upsamplings (Elastic) can be applied to any CNN layers  $T_i(x)$  in any architecture to improve accuracy of a model. Our applications are introduced in the next section.

### 2.3.3 Augmenting Models with Elastic

Now, we show how to apply Elastic on different network architecture. To showcase the power of Elastic, we apply Elastic on some state-of-the-art network architectures: ResNeXt[16], Deep Layer Aggregation (DLA)[12], and DenseNet[11]. A natural way of applying Elastic on current classification models is to augment bottleneck layers with multiple branches. This makes our modification on ResNeXt and DLA almost identical. At each layer we apply downsampling and bilinear upsampling to a portion of branches, as shown in Figure 2.3-left. In DenseNet we compile an equivalent version by parallelizing a single branch into two branches and then apply downsampling and upsampling on some of the branches, as shown in Figure 2.3-right. Note that applying Elastic reduces FLOPs

in each layer. To match the original FLOPs we increase number of layers in the network while dividing similar number of FLOPs across resolutions.

**Relation to other multi-scaling approaches.** As discussed in section 2.2, most of current multi-scaling approaches can be categorized into four different categories (1) *image pyramid*, (2) *loss pyramid* (3) *filter pyramid*, and (4) *feature pyramid*. Figure 2.2(b-e) demonstrates the structure of these categories. All of these models can improve the accuracy usually under a higher computational budget. Elastic (Figure 2.2) guarantees no extra computational cost while achieving better accuracy. Filter pyramid is the most similar model to Elastic. The major difference to the filter pyramid is that the number of FLOPs to cover a higher receptive field in Elastic is proportionally lower due to the downsampling whereas in the filter pyramid the FLOPs is higher or the same as the original convolution depending of filter size or dilation parameters. Table 2.1 compares the FLOPs and number of parameters between Elastic and feature/filter pyramid for a single convolutional operation. Note that the FLOPs and parameters in Elastic is always (under any branching  $q$  and scaling ratio  $r$ ) lower or equal to the original model whereas in filter/feature pyramid this is higher or equal. Feature pyramid methods are usually applied on top of an existing classification model, by concatenating features from different resolutions. It is capable of merging features from different scales in the backbone model and shows improvements on various tasks, but it does not intrinsically change the scaling policy. Our Elastic structure can be viewed as a feature pyramid inside a layer, which is able to model different scaling policies. Spatial pyramid pooling or Atrous(dilated) spatial pyramid shares the same limitation as feature pyramid methods.

## 2.4 Experiments

In this section, we present experiments on applying Elastic to current strong classification models. We evaluate their performances on ImageNet classification, and we show consis-

**Table 2.1. Computation in multi-scaling models.** This table compares the FLOPs and number of parameters between Elastic and feature/filter pyramid for a single convolutional operation, where the input tensor is  $n \times n \times c$  and the filter size is  $k \times k$ .  $q$  denotes the number of branches in the layer, where  $\sum_1^q \frac{1}{b_i} = 1$  and  $b_i > 1$  and  $r_i > 1$  denote the branching and scaling ratio respectively. Note that the FLOPs and parameters in Elastic is always (under any branching  $q$  and scaling ratio  $r$ ) lower than or equal to the original model whereas in feature/filter pyramid is higher or equal.

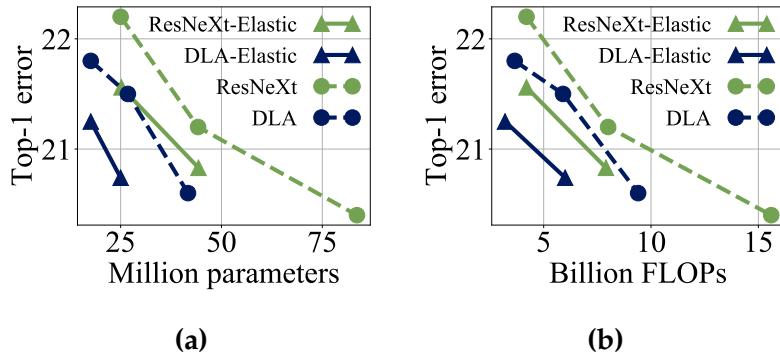
Multi-Scaling Method	FLOPs	Parameters
Single Scale	$n^2ck^2$	$ck^2$
Feature Pyramid (concat)	$n^2(qc)k^2$	$(qc)k^2$
Feature Pyramid (add)	$n^2ck^2$	$ck^2$
Filter Pyramid (standard)	$\sum_{i=1}^q \frac{n^2c(kr_i)^2}{b_i}$	$\sum_{i=1}^q \frac{c(kr_i)^2}{b_i}$
Filter Pyramid (dilated)	$n^2ck^2$	$ck^2$
Elastic	$\sum_{i=1}^q \frac{(\frac{n}{r_i})^2ck^2}{b_i}$	$ck^2$

tent improvements over current models. Furthermore, in order to show the generality of our approach, we transfer our pre-trained Elastic models to multi-label image classification and semantic segmentation. We use ResNeXt [16], DenseNet[11] and DLA [12] as our base models to be augmented with Elastic.

**Implementation details.** We use the official PyTorch ImageNet codebase with random crop augmentation but without color or lighting augmentation, and we report standard  $224 \times 224$  single crop error on the validation set. We train our model with 8 workers (GPUs) and 32 samples per worker. Following DLA [12], all models are trained for 120 epochs with learning rate 0.1 and divided by 10 at epoch 30, 60, 90. We initialize our models using normal He initialization [42]. Stride-2 average poolings are adopted as our downsamplings unless otherwise notified since most of our downsamplings are  $2 \times$  downsamplings, in which case bilinear downsampling is equivalent to average pooling. Also, Elastic add-on is applied to all blocks except stride-2 ones or high-level blocks operating at resolution 7.

### 2.4.1 ImageNet Classification

We evaluate Elastic on ImageNet[14] 1000 way classification task (ILSVRC2012). The ILSVRC 2012 dataset contains 1.2 million training images and 50 thousand validation images. In this experiment, we show that our Elastic add-on consistently improves the accuracy of the state-of-the-art models without introducing extra computation or parameters. Table 2.2 compares the top-1 and top-5 error rates of all of the base models with the Elastic augmentation (indicated by '+Elastic') and shows the number of parameters and FLOPs used for a single inference. Besides DenseNet, ResNeXt, DLA, SE-ResNeXt50+Elastic is also reported. In all the tables "\*" denotes our implementation of the model. It shows that our improvement is almost orthogonal to the channel calibration proposed in [17]. In addition, we include ResNeXt50x2+Elastic to show that our improvement does not come from more depth added to ResNeXt101. In Figure 4 we project the numbers in the Table 2.2 into two plots: accuracy vs. number of parameters (Figure 2.4a) and accuracy vs. FLOPs (Figure 2.4b). This plot shows that our Elastic model can reach to a higher accuracy without any extra (or with lower) computational cost.



**Figure 2.4. Imagenet Accuracy vs. FLOPS and Parameters.** This figure shows our Elastic model can achieve a lower error without any extra (or with lower) computational cost.

**Table 2.2. State-of-the-art model comparisons on ImageNet validation set.** Base models (DenseNet, ResNeXt, and DLA) are augmented by Elastic (indicated by '+Elastic'). \* indicates our implementation of these models. Note that augmenting with Elastic always improves accuracy across the board.

Model	# Params	FLOPs	Top-1	Top-5
DenseNet201*	20.0M	4.4B	22.25	6.26
DenseNet201+Elastic	19.5M	4.3B	<b>22.07</b>	<b>6.00</b>
ResNeXt50	25.0M	4.2B	22.2	-
ResNeXt50*	25.0M	4.2B	22.23	6.25
ResNeXt50+Elastic	25.2M	4.2B	<b>21.56</b>	<b>5.83</b>
SE-ResNeXt50*	27.6M	4.2B	21.87	5.93
SE-ResNeXt50+Elastic	27.8M	4.2B	<b>21.38</b>	<b>5.86</b>
ResNeXt101	44.2M	8.0B	21.2	5.6
ResNeXt101*	44.2M	8.0B	21.18	5.83
ResNeXt101+Elastic	44.3M	7.9B	<b>20.83</b>	<b>5.41</b>
ResNeXt50x2+Elastic	45.6M	7.9B	20.86	5.52
DLA-X60	17.6M	3.6B	21.8	-
DLA-X60*	17.6M	3.6B	21.92	6.03
DLA-X60+Elastic	17.6M	3.2B	<b>21.25</b>	<b>5.71</b>
DLA-X102	26.8M	6.0B	21.5	-
DLA-X102+Elastic	25.0M	6.0B	<b>20.71</b>	<b>5.38</b>

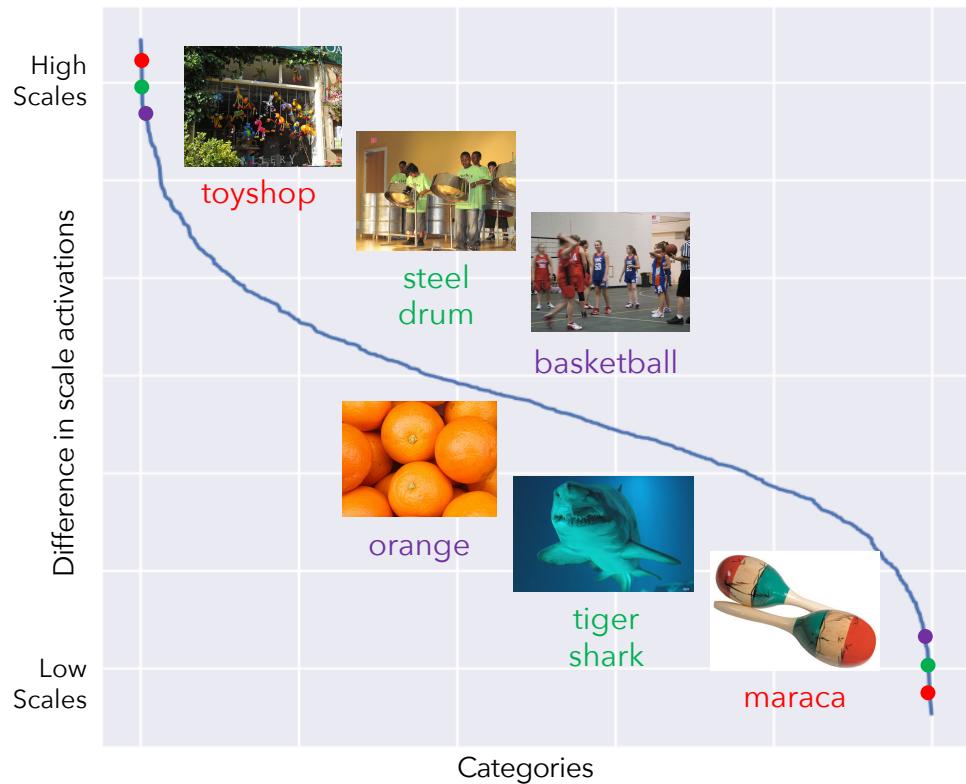
## 2.4.2 Scale Policy Analysis

To analyze the learned scale policy of our Elastic model, we define a simple score that shows at each block what was the resolution level (high or low) that the input tensor was processed. We formally define this scale policy score at each block by differences of mean activations in high-resolution and low-resolution branches.

$$S = \frac{1}{4HWC} \sum_{h=1}^{2H} \sum_{w=1}^{2W} \sum_{c=1}^C x_{hwc}^{high} - \frac{1}{HWC} \sum_{h=1}^H \sum_{w=1}^W \sum_{c=1}^C x_{hwc}^{low} \quad (2.5)$$

where  $H, W, C$  are the height, width and number of channels in low resolution branches.  $x^{high}$  and  $x^{low}$  are the activations after  $3 \times 3$  convolutions, fixed batch normalizations, and ReLU in high-resolution and low-resolution branches respectively. Figure 2.5 shows all of the categories in ImageNet validation sorted by the mean scale policy score  $S$  (average over all layers for all images within a category). As it can be seen, categories with more complex images appear to have a larger  $S$  i.e. they mostly go through high-resolution branches in each block and images with simpler patterns appear to have smaller  $S$  which means they mostly go through the low-resolution branches in each block.

To analyze the impact of the scale policy on the accuracy of the Elastic, we represent each image (in the ImageNet validation set) by a 17-dimensional vector such that the values of the 17 elements are the scale policy score  $S$  for the 17 Elastic blocks in a ResNeXt50+Elastic model. Then we apply tsne[43] on all these vectors to get a two-dimensional visualization. In figure 2.6-(left) we draw all the images in the tsne coordinates. It can be seen that images are clustered based on their complexity pattern. In figure 2.6-(middle) for all of the images we show the 17 scale policy scores  $S$  in 17 blocks. As it can be seen most of the images go through the high-resolution branches on the early layers and low-resolution branches at the later layers but some images break this pattern. For examples, images pointed by the green circle are activating high-resolution branches in the 13<sup>th</sup> block of the network. These images usually contain a complex pattern that the network needs to



**Figure 2.5. Scale policy for complex vs. simple image categories.** This figure shows the overall block scale policy score on the entire ImageNet categories. It shows that categories with complex image patterns mostly go through the high-resolution branches in the network and categories with simpler image pattern go through the low-resolution branches.

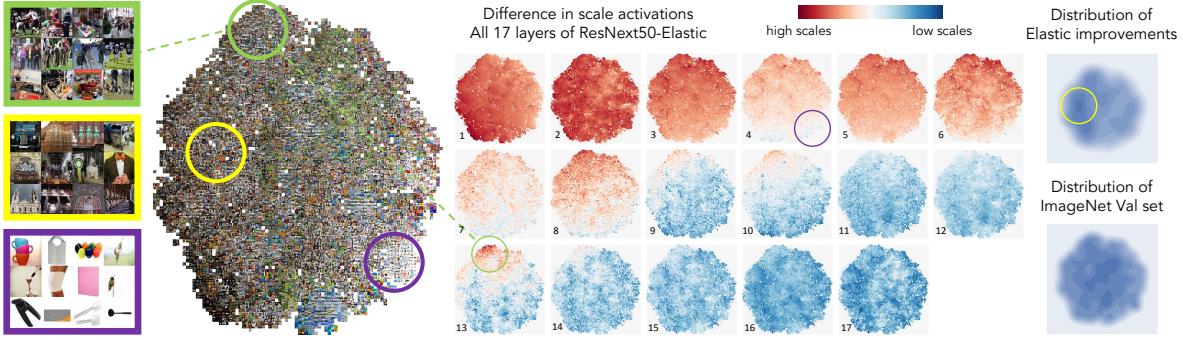
extract features in high-resolution to classify correctly. Images pointed by the purple circle are activating low-resolution branches at early layers, the 4<sup>th</sup> block of the network. These images usually contain a simple pattern that the network can classify at low-resolution early on. In Figure 2.6-(right) we show the density of all validation images in the tsne space in the bottom row, and in the top row, we show the density of images that are correctly classified by our Elastic model and miss-classified by the base ResNeXt model. This comparison shows that most of the images that Elastic can improve predictions on are the ones with more challenging scale properties. Some of them are pointed out by the yellow circle.

### 2.4.3 MS COCO Multi-Label Classification

To further investigate the generality of our model, we finetune our ImageNet pre-trained model and evaluate on MS COCO multi-label classification task. The MSCOCO images are far more complicated in that there exist multiple objects from different categories and scales in each image.

**Implementation details.** All models that we report are finetuned from ImageNet pre-trained model for 36 epochs with learning rate starting at 0.001 and being divided by 10 at epoch 24, 30. We train on 4 workers and 24 images per worker with SGD and weight decay of 0.0005. We train our models with binary cross entropy (BCE) loss, which is usually used as a baseline for domain-specific works that explicitly model spatial or semantic relations. We use the same data augmentations as our ImageNet training, and adopt standard multi-label testing on images resized to  $224 \times 224$ .

**Evaluation metrics.** Following the literature of multi-label classification[44]–[47], results are evaluated using macro/micro evaluations. After training the models with BCE loss, labels with greater than 0.5 probability are considered positive. Then, macro and micro F1-scores are calculated to measure overall performance and the average of per-class



**Figure 2.6. Scale policy analysis.** This figure shows the impact of the scale policy on the accuracy of our Elastic model. (left) shows all the ImageNet validation set clustered using tsne by their scale policy pattern in the ResNeXt50+Elastic as discussed in section 2.4.2. (middle) shows the the scale policy score of all the images at 17 blocks of the network. Most of the images use high-resolution features at early layers and low-resolution features at later layers but some images break this pattern. Images pointed in the **green circle** use high-resolution features in the 13<sup>th</sup> block. Images pointed in the **purple circle** use low-resolution features in the 4<sup>th</sup> block. These images usually contain a simpler pattern. (right)-bottom shows the density of images in the tsne space and (right)-top shows the density of the images that got correctly classified by Elastic model but miss-classified by the base ResNeXt model. This shows that Elastic can improve prediction when images are challenging in terms of their scale information. Some samples are pointed by the **yellow circle**. Best viewed in color.

performances respectively.

**Results.** Table 2.3 shows that elastic consistently improves per-class F1 and overall F1. In the case of DLA, Elastic augmentation even reduces the FLOPs and number of parameters by a large margin.

**Scale challenging images.** We claimed that Elastic is very effective on scale challenging images. Now, we empirically show that a large portion of the accuracy improvement of our Elastic model is rooted in a better scale policy learning. We follow MSCOCO official

**Table 2.3. MSCOCO multi-class classification.** This table shows the generality of our Elastic model by finetuning pre-trained ImageNet models on MSCOCO multi-class images with binary cross entropy loss. Elastic improves F1 scores all across the board.

Model	F1-PerClass	F1-Overall
ResNet101*	69.98	74.58
DenseNet201*	69.95	74.50
DenseNet201+Elastic	<b>70.40</b>	<b>74.99</b>
DLA-X60*	70.79	75.41
DLA-X60+Elastic	<b>71.35</b>	<b>75.77</b>
ResNeXt50*	70.12	74.52
ResNeXt50+Elastic	<b>71.08</b>	<b>75.37</b>
ResNeXt101*	70.95	75.21
ResNeXt101+Elastic	<b>71.83</b>	<b>75.93</b>

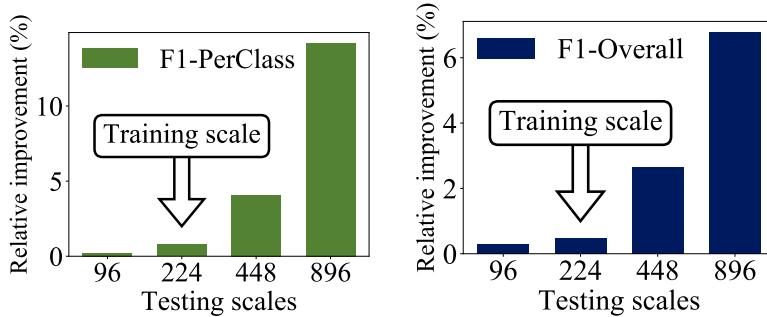
split of *small*, *medium*, and *large* objects. Per-class and overall F1, on small, medium and large objects, are computed. Since we don't have per-scale predictions, false positives are shared and re-defined as cases where none of small, medium, large object appears, but the model predicts positive. Results in Table 2.4 show that ResNeXt50 + Elastic provides the largest gains on small objects. Elastic allows large objects to be dynamically captured by low resolution paths, so filters in high resolution branches do not waste capacity dealing with parts of large objects. Elastic blocks also merge various scales and feed scale-invariant features into the next block, so it shares computation in all higher blocks, and thus allows more capacity for small objects, at high resolution. This proves our hypothesis that Elastic understands scale challenging images better through scale policy learning.

**Scale stress test.** Besides standard testing where images are resized to  $224 \times 224$ , we also perform a stress test on the validation set. MSCOCO images' resolutions are around  $640 \times 480$ . Given a DLA-X60 model trained with  $224 \times 224$  images, we also test it with images from different resolutions:  $96 \times 96$ ,  $448 \times 448$ ,  $896 \times 896$  and change the last average pooling layer accordingly. Figure 2.7 shows that Elastic does not only perform well on

**Table 2.4.** F1 scores on small, medium, and large objects respectively. C means per-class F1 and O means overall F1. ResNeXt50 + Elastic improves the most on small objects.

Model	Sm-C	Md-C	Lg-C	Sm-O	Md-O	Lg-O
ResNeXt50	45.57	61.99	65.88	58.51	68.51	77.53
+Elastic	46.67	63.05	66.46	59.47	69.47	78.03
Relative	2.43%	1.72%	0.88%	1.64%	1.40%	0.65%

trained scale, but also shows greater improvement on higher resolution images at test time. In addition, we do not observe an accuracy drop on  $96 \times 96$  test, though the total computation assigned to low level is reduced in DLA-X60+Elastic.



**Figure 2.7. Scale stress test** on MSCOCO multi-label classification. This bar chart shows the relative F1 improvement of DLA-x60 being augmented Elastic over different image resolutions. Although both models are trained on  $224 \times 224$  images, Elastic shows larger improvement when tested on high-resolution images.

#### 2.4.4 PASCAL VOC Semantic Segmentation

To show the strength of our Elastic model on a pixel level classification task, we report experiments on PASCAL VOC semantic segmentation. ResNeXt models use weight decay  $5e-4$  instead of  $1e-4$  in ResNet. All models are trained for 50 epochs and we report mean intersection-over-union (IOU) on the val set. Other implementation details follow [48], with MG(1, 2, 4), ASPP(6, 12, 18), image pooling, OS=16, batch size of 16, for both training

and validation, without bells and whistles. Our ResNet101 reproduces the mIOU of 77.21% reported in [48]. Our DLA models use the original iterative deep aggregation as a decoder and are trained with the same scheduling as [48]. In Table. 2.5, Elastic shows a large margin of improvement. This verifies that Elastic finds the scale policy that allows processing high-level semantic information and low-level boundary information together, which is critical in the task of semantic segmentation.

**Table 2.5. PASCAL VOC semantic segmentation.** This table compares the accuracy of semantic image segmentation (mIOU%) using Elastic models vs. the original model. Elastic models outperform original models by a large margin. This supports that Elastic learns a scale policy that allows processing high-level semantic information and low-level boundary information together.

Model	Original	Elastic
ResNeXt50*	75.29	<b>77.70</b>
ResNeXt101*	77.47	<b>78.51</b>
DLA-X60*	69.96	<b>73.59</b>

#### 2.4.5 Ablation Study

In this section, we study the effect of different elements in Elastic models. We chose DLA-X60 as our baseline and applied Elastic to perform the ablation experiments.

**Upsampling/Downsampling methods.** We carried our experiments with bilinear up-sampling and downsampling on DLA-X60+Elastic. In Table 2.6 we show the accuracy of ImageNet classification using Elastic by different choices of up(down)sampling methods: Bilinear, Nearest, Trained filters and Trained Dilated filters with and without average pooling (indicated by w/ AP). Our experiment shows Elastic with the bilinear up(down)sampling is the best choice.

Method	# FLOPs	Top-1 error
Original (no Elastic)	3.6B	21.92
Bilinear w/ AP	<b>3.2B</b>	21.25
Nearest w/ AP	3.2B	21.49
Trained Dilated Filter w/ AP	3.6B	<b>21.20</b>
Trained Dilated Filter	3.6B	21.60
Trained Filter	3.2B	21.52

**Table 2.6. Ablation study of up(down)sampling methods.** In this table, we show the accuracy of ImageNet classification using Elastic by different choices of up(down)sampling methods. **w/ AP** indicates average pooling. Our experiment shows Elastic with bilinear up(down)sampling is the best choice with reduced FLOPs.

**High/low-resolution branching rate.** We sweep over different choices of dividing parallel branches in the blocks into the high and low-resolutions. In table 2.7 we compare the variations of the percentage of branches allocated to high and low-resolutions at each block. This experiment shows that the best trade-off is when we equally divide the branches into high and low-resolutions. Interestingly, all of the branching options are outperforming the vanilla model (without Elastic). This shows that our Elastic model is quite robust to this parameter.

**Table 2.7. Ablation study of high(low) resolution branching rates.** In this table, we evaluate different branching rate across high and low-resolutions at each block. We observe that the best trade-off is when we equally divide the branches into high and low-resolutions. Independent of the ratio, all variations of branching are better than the base model.

High-Res	Low-Res	FLOPs	Top-1 error
100%	0%	3.6B	21.92
50%	50%	3.2B	21.25
75%	25%	3.4B	21.35
25%	75%	2.9B	21.44

## 2.5 Conclusion

We proposed Elastic, a model that captures scale variations in images by learning the scale policy from data. Our Elastic model is simple, efficient and very effective. Our model can easily be applied to any CNN architectures and improve accuracy while maintaining the same computation (or lower) as the original model. We applied Elastic to several state-of-the-art network architectures and showed consistent improvement on ImageNet classification, MSCOCO multi-class classification, and PASCAL VOC semantic segmentation. Our results show major improvement for images with scale challenges e.g. images consist of several small objects or objects with large scale variations.

# Chapter 3

## Axial-DeepLab: Stand-Alone Axial-Attention for Panoptic Segmentation

This chapter studies the possibility of completely replacing local convolution with self-attention layers that model global relations in all the layers.

### 3.1 Introduction

Convolution is a core building block in computer vision. Early algorithms employ convolutional filters to blur images, extract edges, or detect features. It has been heavily exploited in modern neural networks [8], [49] due to its efficiency and generalization ability, in comparison to fully connected models [50]. The success of convolution mainly comes from two properties: translation equivariance, and locality. Translation equivariance, although not exact [51], aligns well with the nature of imaging and thus generalizes the model to different positions or to images of different sizes. Locality, on the other hand, reduces parameter counts and M-Adds. However, it makes modeling long range relations challenging.

A rich set of literature has discussed approaches to modeling long range interactions

in convolutional neural networks (CNNs). Some employ atrous convolutions [52]–[55], larger kernel [56], or image pyramids [1], [57], either designed by hand or searched by algorithms [58]–[60]. Another line of works adopts attention mechanisms. Attention shows its ability of modeling long range interactions in language modeling [61], [62], speech recognition [63], [64], and neural captioning [65]. Attention has since been extended to vision, giving significant boosts to image classification [66], object detection [67], semantic segmentation [68], video classification [69], and adversarial defense [70]. These works enrich CNNs with non-local or long-range attention modules.

Recently, stacking attention layers as stand-alone models without any spatial convolution has been proposed [71], [72] and shown promising results. However, naive attention is computationally expensive, especially on large inputs. Applying local constraints to attention, proposed by [71], [72], reduces the cost and enables building fully attentional models. However, local constraints limit model receptive field, which is crucial to tasks such as segmentation, especially on high-resolution inputs. In this chapter, we propose to adopt axial-attention [68], [73], which not only allows efficient computation, but recovers the large receptive field in stand-alone attention models. The core idea is to factorize 2D attention into two 1D attentions along height- and width-axis sequentially. Its efficiency enables us to attend over large regions and build models to learn long range or even global interactions. Additionally, most previous attention modules do not utilize positional information, which degrades attention’s ability in modeling position-dependent interactions, like shapes or objects at multiple scales. Recent works [66], [71], [72] introduce positional terms to attention, but in a context-agnostic way. In this chapter, we augment the positional terms to be context-dependent, making our attention position-sensitive, with marginal costs.

We show the effectiveness of our axial-attention models on ImageNet [14] for classification, and on three datasets (COCO [15], Mapillary Vistas [74], and Cityscapes [75]) for panoptic segmentation [76], instance segmentation, and semantic segmentation. In

particular, on ImageNet, we build an Axial-ResNet by replacing the  $3 \times 3$  convolution in all residual blocks [10] with our position-sensitive axial-attention layer, and we further make it fully attentional [71] by adopting axial-attention layers in the ‘stem’. As a result, our Axial-ResNet attains state-of-the-art results among stand-alone attention models on ImageNet. For segmentation tasks, we convert Axial-ResNet to Axial-DeepLab by replacing the backbones in Panoptic-DeepLab [77]. On COCO [15], our Axial-DeepLab outperforms the current *bottom-up* state-of-the-art, Panoptic-DeepLab [78], by 2.8% PQ on test-dev set. We also show state-of-the-art segmentation results on Mapillary Vistas [74], and Cityscapes [75].

To summarize, our contributions are four-fold:

- The proposed method is the first attempt to build stand-alone attention models with large or global receptive field.
- We propose position-sensitive attention layer that makes better use of positional information without adding much computational cost.
- We show that axial attention works well, not only as a stand-alone model on image classification, but also as a backbone on panoptic segmentation, instance segmentation, and segmantic segmentation.
- Our Axial-DeepLab improves significantly over bottom-up state-of-the-art on COCO, achieving comparable performance of two-stage methods. We also surpass previous state-of-the-art methods on Mapillary Vistas and Cityscapes.

## 3.2 Related Work

**Top-down panoptic segmentation.** Most state-of-the-art panoptic segmentation models employ a two-stage approach where object proposals are firstly generated followed by sequential processing of each proposal. We refer to such approaches as top-down or

proposal-based methods. Mask R-CNN [79] is commonly deployed in the pipeline for instance segmentation, paired with a light-weight stuff segmentation branch. For example, Panoptic FPN [80] incorporates a semantic segmentation head to Mask R-CNN [79], while Porzi *et al.* [81] append a light-weight DeepLab-inspired module [82] to the multi-scale features from FPN [41]. Additionally, some extra modules are designed to resolve the overlapping instance predictions by Mask R-CNN. TASCNet [83] and AUNet [84] propose a module to guide the fusion between ‘thing’ and ‘stuff’ predictions, while Liu *et al.* [85] adopt a Spatial Ranking module. UPSNet [86] develops an efficient parameter-free panoptic head for fusing ‘thing’ and ‘stuff’, which is further explored by Li *et al.* [87] for end-to-end training of panoptic segmentation models. AdaptIS [88] uses point proposals to generate instance masks.

**Bottom-up panoptic segmentation.** In contrast to top-down approaches, bottom-up or proposal-free methods for panoptic segmentation typically start with the semantic segmentation prediction followed by grouping ‘thing’ pixels into clusters to obtain instance segmentation. DeeperLab [89] predicts bounding box four corners and object centers for class-agnostic instance segmentation. SSAP [90] exploits the pixel-pair affinity pyramid [91] enabled by an efficient graph partition method [92]. BBFNet [93] obtains instance segmentation results by Watershed transform [94], [95] and Hough-voting [96], [97]. Recently, Panoptic-DeepLab [78], a simple, fast, and strong approach for bottom-up panoptic segmentation, employs a class-agnostic instance segmentation branch involving a simple instance center regression [98]–[100], coupled with DeepLab semantic segmentation outputs [48], [55], [101]. Panoptic-DeepLab has achieved state-of-the-art results on several benchmarks, and our method builds on top of it.

**Self-attention.** Attention, introduced by [102] for the encoder-decoder in a neural sequence-to-sequence model, is developed to capture correspondence of tokens between two sequences. In contrast, self-attention is defined as applying attention to a single context

instead of across multiple modalities. Its ability to directly encode long-range interactions and its parallelizability, has led to state-of-the-art performance for various tasks [6], [61], [103]–[107]. Recently, self-attention has been applied to computer vision, by augmenting CNNs with non-local or long-range modules. Non-local neural networks [69] show that self-attention is an instantiation of non-local means [108] and achieve gains on many vision tasks such as video classification and object detection. Additionally, [66], [109] show improvements on image classification by combining features from self-attention and convolution. State-of-the-art results on video action recognition tasks [109] are also achieved in this way. On semantic segmentation, self-attention is developed as a context aggregation module that captures multi-scale context [68], [110]–[112]. Efficient attention methods are proposed to reduce its complexity [68], [107], [113]. Additionally, CNNs augmented with non-local means [108] are shown to be more robust to adversarial attacks [70]. Besides discriminative tasks, self-attention is also applied to generative modeling of images [73], [114], [115]. Recently, [71], [72] show that self-attention layers alone could be stacked to form a fully attentional model by restricting the receptive field of self-attention to a *local* square region. Encouraging results are shown on both image classification and object detection. In this chapter, we follow this direction of research and propose a stand-alone self-attention model with large or global receptive field, making self-attention models *non-local* again. Our models are evaluated on bottom-up panoptic segmentation and show significant improvements.

### 3.3 Method

We begin by formally introducing our position-sensitive self-attention mechanism. Then, we discuss how it is applied to axial-attention and how we build stand-alone Axial-ResNet and Axial-DeepLab with axial-attention layers.

### 3.3.1 Position-Sensitive Self-Attention

**Self-Attention.** Self-attention mechanism is usually applied to vision models as an add-on to augment CNNs outputs [68], [69], [114]. Given an input feature map  $x \in \mathbb{R}^{h \times w \times d_{in}}$  with height  $h$ , width  $w$ , and channels  $d_{in}$ , the output at position  $o = (i, j)$ ,  $y_o \in \mathbb{R}^{d_{out}}$ , is computed by pooling over the projected input as:

$$y_o = \sum_{p \in \mathcal{N}} \text{softmax}_p(q_o^T k_p) v_p \quad (3.1)$$

where  $\mathcal{N}$  is the whole location lattice, and queries  $q_o = W_Q x_o$ , keys  $k_o = W_K x_o$ , values  $v_o = W_V x_o$  are all linear projections of the input  $x_o \forall o \in \mathcal{N}$ .  $W_Q, W_K \in \mathbb{R}^{d_q \times d_{in}}$  and  $W_V \in \mathbb{R}^{d_{out} \times d_{in}}$  are all learnable matrices. The  $\text{softmax}_p$  denotes a softmax function applied to all possible  $p = (a, b)$  positions, which in this case is also the whole 2D lattice.

This mechanism pools values  $v_p$  globally based on affinities  $x_o^T W_Q^T W_K x_p$ , allowing us to capture related but non-local context in the whole feature map, as opposed to convolution which only captures local relations.

However, self-attention is extremely expensive to compute ( $\mathcal{O}(h^2w^2)$ ) when the spatial dimension of the input is large, restricting its use to only high levels of a CNN (*i.e.*, downsampled feature maps) or small images. Another drawback is that the global pooling does not exploit positional information, which is critical to capture spatial structures or shapes in vision tasks.

These two issues are mitigated in [71] by adding local constraints and positional encodings to self-attention. For each location  $o$ , a local  $m \times m$  square region is extracted to serve as a memory bank for computing the output  $y_o$ . This significantly reduces its computation to  $\mathcal{O}(hwm^2)$ , allowing self-attention modules to be deployed as stand-alone layers to form a fully self-attentional neural network. Additionally, a learned relative positional encoding term is incorporated into the affinities, yielding a dynamic prior of

where to look at in the receptive field (*i.e.*, the local  $m \times m$  square region). Formally, [71] proposes

$$y_o = \sum_{p \in \mathcal{N}_{m \times m}(o)} \text{softmax}_p(q_o^T k_p + q_o^T r_{p-o}) v_p \quad (3.2)$$

where  $\mathcal{N}_{m \times m}(o)$  is the local  $m \times m$  square region centered around location  $o = (i, j)$ , and the learnable vector  $r_{p-o} \in \mathbb{R}^{d_q}$  is the added relative positional encoding. The inner product  $q_o^T r_{p-o}$  measures the compatibility from location  $p = (a, b)$  to location  $o = (i, j)$ . We do not consider absolute positional encoding  $q_o^T r_p$ , because they do not generalize well compared to the relative counterpart [71]. In the following paragraphs, we drop the term relative for conciseness.

In practice,  $d_q$  and  $d_{out}$  are much smaller than  $d_{in}$ , and one could extend single-head attention in Equation (3.2) to multi-head attention to capture a mixture of affinities. In particular, multi-head attention is computed by applying  $N$  single-head attentions in parallel on  $x_o$  (with different  $W_Q^n, W_K^n, W_V^n, \forall n \in \{1, 2, \dots, N\}$  for the  $n$ -th head), and then obtaining the final output  $z_o$  by concatenating the results from each head, *i.e.*,  $z_o = \text{concat}_n(y_o^n)$ . Note that positional encodings are often shared across heads, so that they introduce marginal extra parameters.

**Position-Sensitivity.** We notice that previous positional bias only depends on the query pixel  $x_o$ , not the key pixel  $x_p$ . However, the keys  $x_p$  could also have information about which location to attend to. We therefore add a key-dependent positional bias term  $k_p^T r_{p-o}^k$ , besides the query-dependent bias  $q_o^T r_{p-o}^q$ .

Similarly, the values  $v_p$  do not contain any positional information in Equation (3.2). In the case of large receptive fields or memory banks, it is unlikely that  $y_o$  contains the precise location from which  $v_p$  comes. Thus, previous models have to trade-off between using smaller receptive fields (*i.e.*, small  $m \times m$  regions) and throwing away precise spatial structures. In this chapter, we enable the output  $y_o$  to retrieve relative positions  $r_{p-o}^v$ ,

besides the content  $v_p$ , based on query-key affinities  $q_o^T k_p$ . Formally,

$$y_o = \sum_{p \in \mathcal{N}_{m \times m}(o)} \text{softmax}_p(q_o^T k_p + q_o^T r_{p-o}^q + k_p^T r_{p-o}^k)(v_p + r_{p-o}^v) \quad (3.3)$$

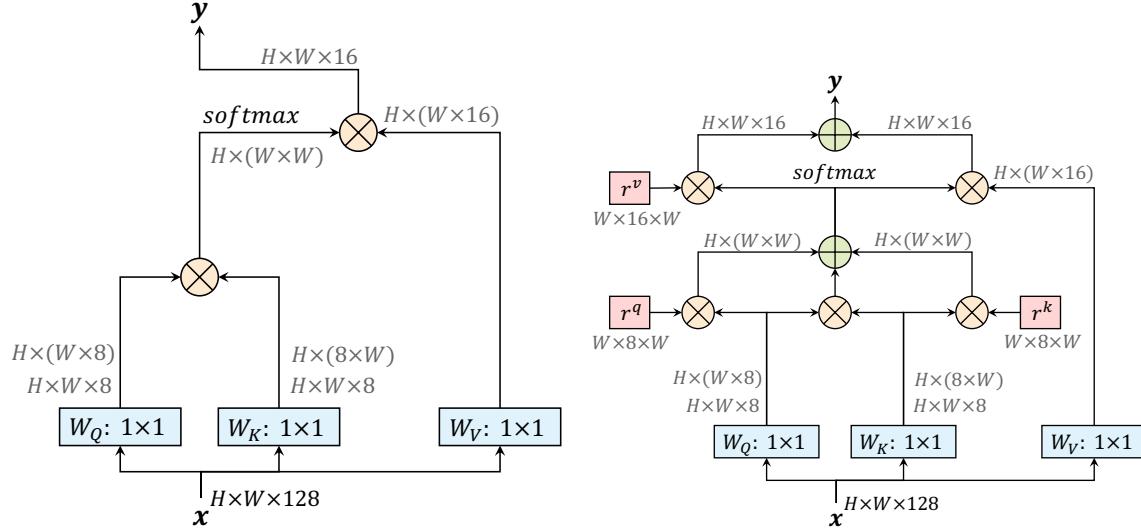
where the learnable  $r_{p-o}^k \in \mathbb{R}^{d_q}$  is the positional encoding for keys, and  $r_{p-o}^v \in \mathbb{R}^{d_{out}}$  is for values. Both vectors do not introduce many parameters, since they are shared across attention heads in a layer, and the number of local pixels  $|\mathcal{N}_{m \times m}(o)|$  is usually small.

We call this design *position-sensitive* self-attention, which captures long range interactions with precise positional information at a reasonable computation overhead, as verified in our experiments.

### 3.3.2 Axial-Attention

The local constraint, proposed by the stand-alone self-attention models [71], significantly reduces the computational costs in vision tasks and enables building fully self-attentional model. However, such constraint sacrifices the global connection, making attention's receptive field no larger than a depthwise convolution with the same kernel size. Additionally, the local self-attention, performed in local square regions, still has complexity quadratic to the region length, introducing another hyper-parameter to trade-off between performance and computation complexity. In this chapter, we propose to adopt axial-attention [68], [73] in stand-alone self-attention, ensuring both global connection and efficient computation. Specifically, we first define an axial-attention layer on the *width-axis* of an image as simply a one dimensional *position-sensitive* self-attention, and use the similar definition for the *height-axis*. To be concrete, the axial-attention layer along the width-axis is defined as follows.

$$y_o = \sum_{p \in \mathcal{N}_{1 \times m}(o)} \text{softmax}_p(q_o^T k_p + q_o^T r_{p-o}^q + k_p^T r_{p-o}^k)(v_p + r_{p-o}^v) \quad (3.4)$$

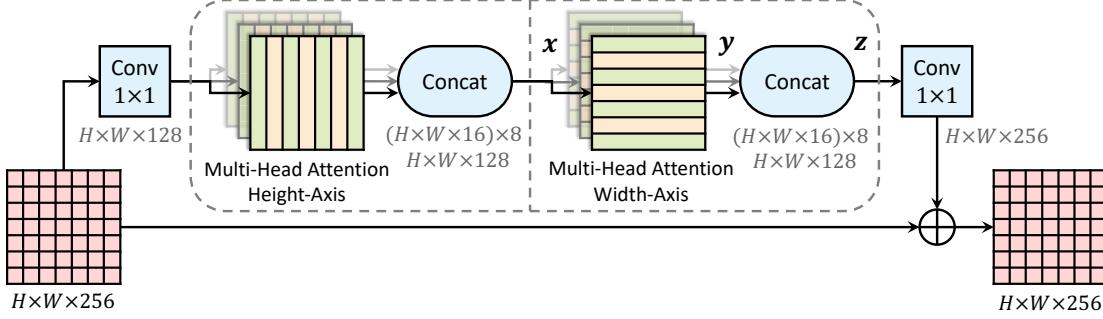


**Figure 3.1.** A non-local block (left) *vs.* our position-sensitive axial-attention applied along the width-axis (right). “ $\otimes$ ” denotes matrix multiplication, and “ $\oplus$ ” denotes element-wise sum. The softmax is performed on the last axis. Blue boxes denote  $1 \times 1$  convolutions, and red boxes denote relative positional encoding. The channels  $d_{in} = 128$ ,  $d_q = 8$ , and  $d_{out} = 16$  is what we use in the first stage of ResNet after ‘stem’

One axial-attention layer propagates information along one particular axis. To capture global information, we employ two axial-attention layers consecutively for the height-axis and width-axis, respectively. Both of the axial-attention layers adopt the multi-head attention mechanism, as described above.

Axial-attention reduces the complexity to  $\mathcal{O}(hwm)$ . This enables global receptive field, which is achieved by setting the span  $m$  directly to the whole input features. Optionally, one could also use a fixed  $m$  value, in order to reduce memory footprint on huge feature maps.

**Axial-ResNet.** To transform a ResNet [10] to an *Axial*-ResNet, we replace the  $3 \times 3$  convolution in the residual bottleneck block by two multi-head axial-attention layers (one for height-axis and the other for width-axis). Optional striding is performed on each axis after the corresponding axial-attention layer. The two  $1 \times 1$  convolutions are kept to shuffle the



**Figure 3.2.** An axial-attention block, which consists of two axial-attention layers operating along height- and width-axis sequentially. The channels  $d_{in} = 128$ ,  $d_{out} = 16$  is what we use in the first stage of ResNet after ‘stem’. We employ  $N = 8$  attention heads

features. This forms our (residual) axial-attention block, as illustrated in Figure 3.2, which is stacked multiple times to obtain Axial-ResNets. Note that we do not use a  $1 \times 1$  convolution in-between the two axial-attention layers, since matrix multiplications ( $W_Q, W_K, W_V$ ) follow immediately. Additionally, the stem (*i.e.*, the first strided  $7 \times 7$  convolution and  $3 \times 3$  max-pooling) in the original ResNet is kept, resulting in a *conv-stem* model where convolution is used in the first layer and attention layers are used everywhere else. In *conv-stem* models, we set the span  $m$  to the whole input from the first block, where the feature map is  $56 \times 56$ .

In our experiments, we also build a full axial-attention model, called Full Axial-ResNet, which further applies axial-attention to the stem. Instead of designing a special spatially-varying attention stem [71], we simply stack three axial-attention bottleneck blocks. In addition, we adopt local constraints (*i.e.*, a local  $m \times m$  square region as in [71]) in the first few blocks of Full Axial-ResNets, in order to reduce computational cost.

**Axial-DeepLab.** To further convert Axial-ResNet to Axial-DeepLab for segmentation tasks, we make several changes as discussed below.

Firstly, to extract dense feature maps, DeepLab [55] changes the stride and atrous rates of the last one or two stages in ResNet [10]. Similarly, we remove the stride of the last stage but we do not implement the ‘atrous’ attention module, since our axial-attention

already captures global information for the whole input. In this chapter, we extract feature maps with output stride (*i.e.*, the ratio of input resolution to the final backbone feature resolution) 16. We do not pursue output stride 8, since it is computationally expensive.

Secondly, we do not adopt the atrous spatial pyramid pooling module (ASPP) [48], [82], since our axial-attention block could also efficiently encode the multi-scale or global information. We show in the experiments that our Axial-DeepLab without ASPP outperforms Panoptic-DeepLab [78] with and without ASPP.

Lastly, following Panoptic-DeepLab [78], we adopt exactly the same stem [24] of three convolutions, dual decoders, and prediction heads. The heads produce semantic segmentation and class-agnostic instance segmentation, and they are merged by majority voting [89] to form the final panoptic segmentation.

In cases where the inputs are extremely large (*e.g.*,  $2177 \times 2177$ ) and memory is constrained, we resort to a large span  $m = 65$  in all our axial-attention blocks. Note that we do not consider the axial span as a hyper-parameter because it is already sufficient to cover long range or even global context on several datasets, and setting a smaller span does not significantly reduce M-Adds.

### 3.4 Experimental Results

We conduct experiments on four large-scale datasets. We first report results with our Axial-ResNet on ImageNet [14]. We then convert the ImageNet pretrained Axial-ResNet to Axial-DeepLab, and report results on COCO [15], Mapillary Vistas [74], and Cityscapes [75] for panoptic segmentation, evaluated by panoptic quality (PQ) [76]. We also report average precision (AP) for instance segmentation, and mean IoU for semantic segmentation on Mapillary Vistas and Cityscapes. Our models are trained using TensorFlow [116] on 128 TPU cores for ImageNet and 32 cores for panoptic segmentation.

**Training protocol.** On ImageNet, we adopt the same training protocol as [71] for a fair comparison, except that we use batch size 512 for Full Axial-ResNets and 1024 for all other models, with learning rates scaled accordingly [117].

For panoptic segmentation, we strictly follow Panoptic-DeepLab [78], except using a linear warm up Radam [118] Lookahead [119] optimizer (with the same learning rate 0.001). All our results on panoptic segmentation use this setting. We note this change does not improve the results, but smooths our training curves. Panoptic-DeepLab yields similar result in this setting.

### 3.4.1 ImageNet

For ImageNet, we build Axial-ResNet-L from ResNet-50 [10]. In detail, we set  $d_{in} = 128$ ,  $d_{out} = 2d_q = 16$  for the first stage after the ‘stem’. We double them when spatial resolution is reduced by a factor of 2 [9]. Additionally, we multiply all the channels [120]–[122] by 0.5, 0.75, and 2, resulting in Axial-ResNet-{S, M, XL}, respectively. Finally, *Stand-Alone* Axial-ResNets are further generated by replacing the ‘stem’ with three axial-attention blocks where the first block has stride 2. Due to the computational cost introduced by the early layers, we set the axial span  $m = 15$  in all blocks of Stand-Alone Axial-ResNets. We always use  $N = 8$  heads [71]. In order to avoid careful initialization of  $W_Q, W_K, W_V, r^q, r^k, r^v$ , we use batch normalizations [123] in all attention layers.

Table 3.1 summarizes our ImageNet results. The baselines ResNet-50 [10] (done by [71]) and Conv-Stem + Attention [71] are also listed. In the conv-stem setting, adding BN to attention layers of [71] slightly improves the performance by 0.3%. Our proposed position-sensitive self-attention (Conv-Stem + PS-Attention) further improves the performance by 0.4% at the cost of extra marginal computation. Our Conv-Stem + Axial-Attention performs on par with Conv-Stem + Attention [71] while being more parameter- and computation-efficient. When comparing with other full self-attention models, our Full Axial-Attention outperforms Full Attention [71] by 0.5%, while being  $1.44 \times$  more parameter-efficient and

$1.09 \times$  more computation-efficient.

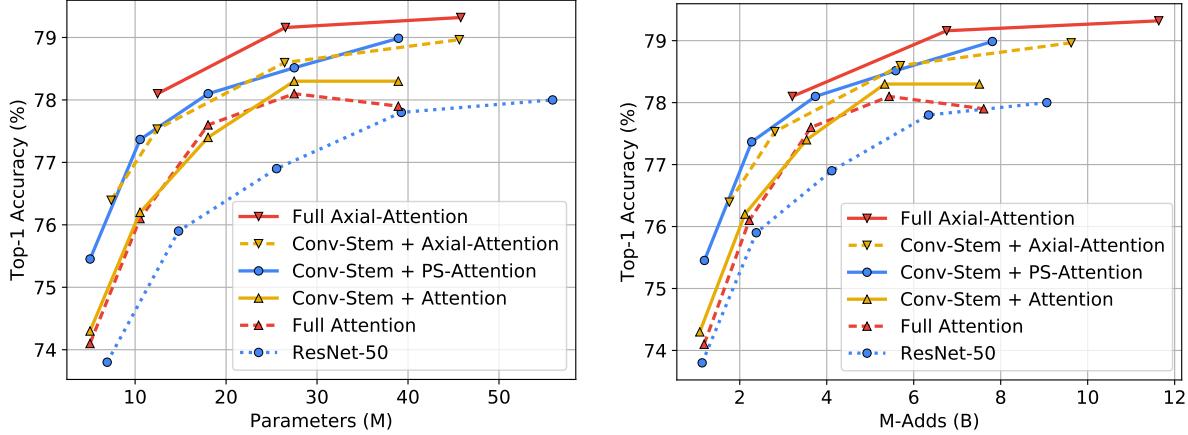
Following [71], we experiment with different network widths (*i.e.*, Axial-ResNets-{S,M,L,XL}), exploring the trade-off between accuracy, model parameters, and computational cost (in terms of M>Adds). As shown in Figure 3.3, our proposed Conv-Stem + PS-Attention and Conv-Stem + Axial-Attention already outperforms ResNet-50 [10], [71] and attention models [71] (both Conv-Stem + Attention, and Full Attention) at all settings. Our Full Axial-Attention further attains the best accuracy-parameter and accuracy-complexity trade-offs.

**Table 3.1.** ImageNet validation set results. **BN:** Use batch normalizations in attention layers. **PS:** Our position-sensitive self-attention. **Full:** Stand-alone self-attention models without spatial convolutions

Method	BN	PS	Full	Params	M-Adds	Top-1
Conv-Stem methods						
ResNet-50 [10], [71]				25.6M	4.1B	76.9
Conv-Stem + Attention [71]				18.0M	3.5B	77.4
Conv-Stem + Attention	✓			18.0M	3.5B	77.7
Conv-Stem + PS-Attention	✓	✓		18.0M	3.7B	78.1
Conv-Stem + Axial-Attention	✓	✓		12.4M	2.8B	77.5
Fully self-attentional methods						
LR-Net-50 [72]			✓	23.3M	4.3B	77.3
Full Attention [71]			✓	18.0M	3.6B	77.6
Full Axial-Attention	✓	✓	✓	<b>12.5M</b>	<b>3.3B</b>	<b>78.1</b>

### 3.4.2 COCO

The ImageNet pretrained Axial-ResNet model variants (with different channels) are then converted to Axial-DeepLab model variant for panoptic segmentation tasks. We first demonstrate the effectiveness of our Axial-DeepLab on the challenging COCO dataset [15], which contains objects with various scales (from less than  $32 \times 32$  to larger than  $96 \times 96$ ).



**Figure 3.3.** Comparing parameters and M-Adds against accuracy on ImageNet classification. Our position-sensitive self-attention (Conv-Stem + PS-Attention) and axial-attention (Conv-Stem + Axial-Attention) consistently outperform ResNet-50 [10], [71] and attention models [71] (both Conv-Stem + Attention, and Full Attention), across a range of network widths (*i.e.*, different channels). Our Full Axial-Attention works the best in terms of both parameters and M-Adds

**Val set:** In Table 3.2, we report our validation set results and compare with other bottom-up panoptic segmentation methods, since our method also belongs to the bottom-up family. As shown in the table, our *single-scale* Axial-DeepLab-S outperforms DeeperLab [89] by 8% PQ, *multi-scale* SSAP [90] by 5.3% PQ, and *single-scale* Panoptic-DeepLab by 2.1% PQ. Interestingly, our *single-scale* Axial-DeepLab-S also outperforms *multi-scale* Panoptic-DeepLab by 0.6% PQ while being  $3.8\times$  parameter-efficient and  $27\times$  computation-efficient (in M-Adds). Increasing the backbone capacity (via large channels) continuously improves the performance. Specifically, our *multi-scale* Axial-DeepLab-L attains 43.9% PQ, outperforming Panoptic-DeepLab [78] by 2.7% PQ.

**Test-dev set:** As shown in Table 3.3, our Axial-DeepLab variants show consistent improvements with larger backbones. Our *multi-scale* Axial-DeepLab-L attains the performance of 44.2% PQ, outperforming DeeperLab [89] by 9.9% PQ, SSAP [90] by 7.3% PQ, and Panoptic-DeepLab [78] by 2.8% PQ, setting a new state-of-the-art among bottom-up ap-

**Table 3.2.** COCO val set. **MS:** Multi-scale inputs

Method	Backbone	MS	Params	M-Adds	PQ	PQ <sup>Th</sup>	PQ <sup>St</sup>
DeeperLab [89]	Xception-71				33.8	-	-
SSAP [90]	ResNet-101	✓			36.5	-	-
Panoptic-DeepLab [78]	Xception-71		46.7M	274.0B	39.7	43.9	33.2
Panoptic-DeepLab [78]	Xception-71	✓	46.7M	3081.4B	41.2	44.9	35.7
Axial-DeepLab-S	Axial-ResNet-S		12.1M	110.4B	41.8	46.1	35.2
Axial-DeepLab-M	Axial-ResNet-M		25.9M	209.9B	42.9	47.6	35.8
Axial-DeepLab-L	Axial-ResNet-L		44.9M	343.9B	43.4	48.5	35.6
Axial-DeepLab-L	Axial-ResNet-L	✓	44.9M	3867.7B	43.9	48.6	36.8

proaches. We also list several top-performing methods adopting the top-down approaches in the table for reference.

**Scale Stress Test:** In order to verify that our model learns long range interactions, we perform a scale stress test besides standard testing. In the stress test, we train Panoptic-DeepLab (X-71) and our Axial-DeepLab-L with the standard setting, but test them on out-of-distribution resolutions (*i.e.*, resize the input to different resolutions). Figure 3.4 summarizes our relative improvements over Panoptic-DeepLab on PQ, PQ (thing) and PQ (stuff). When tested on huge images, Axial-DeepLab shows large gain (30%), demonstrating that it encodes long range relations better than convolutions. Besides, Axial-DeepLab improves 40% on small images, showing that axial-attention is more robust to scale variations.

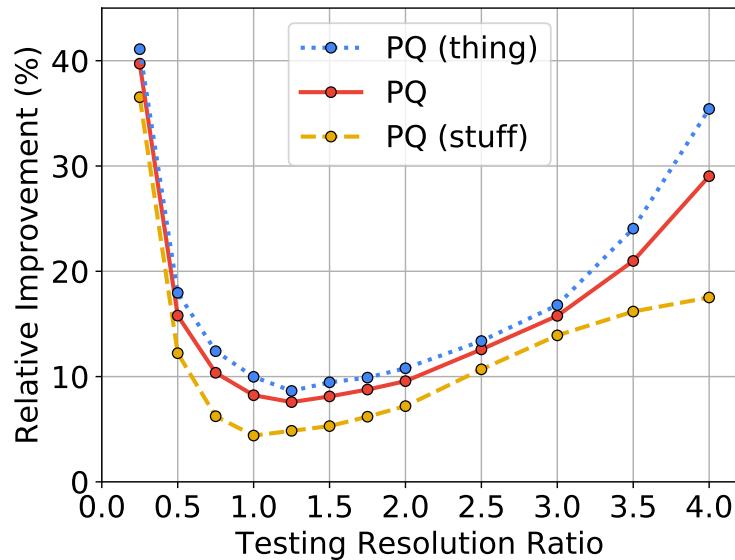
### 3.4.3 Mapillary Vistas

We evaluate our Axial-DeepLab on the large-scale Mapillary Vistas dataset [74]. We only report validation set results, since the test server is not available.

**Val set.** As shown in Table 3.4, our Axial-DeepLab-L outperforms all the state-of-the-art methods in both single-scale and multi-scale cases. Our *single-scale* Axial-DeepLab-L performs 2.4% PQ better than the previous best *single-scale* Panoptic-DeepLab (X-71) [78].

**Table 3.3.** COCO test-dev set. **MS:** Multi-scale inputs

Method	Backbone	MS	PQ	PQ <sup>Th</sup>	PQ <sup>St</sup>
Top-down panoptic segmentation methods					
TASCNet [83]	ResNet-50		40.7	47.0	31.0
Panoptic-FPN [80]	ResNet-101		40.9	48.3	29.7
AdaptIS [88]	ResNeXt-101	✓	42.8	53.2	36.7
AUNet [84]	ResNeXt-152		46.5	55.8	32.5
UPSNet [86]	DCN-101 [124]	✓	46.6	53.2	36.7
Li <i>et al.</i> [87]	DCN-101 [124]		47.2	53.5	37.7
SpatialFlow [125]	DCN-101 [124]	✓	47.3	53.5	37.9
SOGNet [126]	DCN-101 [124]	✓	47.8	-	-
Bottom-up panoptic segmentation methods					
DeeperLab [89]	Xception-71		34.3	37.5	29.6
SSAP [90]	ResNet-101	✓	36.9	40.1	32.0
Panoptic-DeepLab [78]	Xception-71	✓	41.4	45.1	35.9
Axial-DeepLab-S	Axial-ResNet-S		42.2	46.5	35.7
Axial-DeepLab-M	Axial-ResNet-M		43.2	48.1	35.9
Axial-DeepLab-L	Axial-ResNet-L		43.6	48.9	35.6
Axial-DeepLab-L	Axial-ResNet-L	✓	44.2	49.2	36.8



**Figure 3.4.** Scale stress test on COCO val set. Axial-DeepLab gains the most when tested on extreme resolutions. On the x-axis, ratio 4.0 means inference with resolution  $4097 \times 4097$

In multi-scale setting, our lightweight Axial-DeepLab-L performs better than Panoptic-DeepLab (Auto-DeepLab-XL++), not only on panoptic segmentation (0.8% PQ) and instance segmentation (0.3% AP), but also on semantic segmentation (0.8% mIoU), the task that Auto-DeepLab [60] was searched for. Additionally, to the best of our knowledge, our Axial-DeepLab-L attains the best *single-model* semantic segmentation result.

**Table 3.4.** Mapillary Vistas validation set. **MS:** Multi-scale inputs

Method	MS	Params	M-Adds	PQ	PQ <sup>Th</sup>	PQ <sup>St</sup>	AP	mIoU
Top-down panoptic segmentation methods								
TASCNet [83]				32.6	31.1	34.4	18.5	-
TASCNet [83]	✓			34.3	34.8	33.6	20.4	-
AdaptIS [88]				35.9	31.5	41.9	-	-
Seamless [81]				37.7	33.8	42.9	16.4	50.4
Bottom-up panoptic segmentation methods								
DeeperLab [89]				32.0	-	-	-	55.3
Panoptic-DeepLab (Xception-71 [127], [128]) [78]		46.7M	1.24T	37.7	30.4	47.4	14.9	55.4
Panoptic-DeepLab (Xception-71 [127], [128]) [78]	✓	46.7M	31.35T	40.3	33.5	49.3	17.2	56.8
Panoptic-DeepLab (HRNet-W48 [129]) [78]	✓	71.7M	58.47T	39.3	-	-	17.2	55.4
Panoptic-DeepLab (Auto-XL++ [60]) [78]	✓	72.2M	60.55T	40.3	-	-	16.9	57.6
Axial-DeepLab-L		44.9M	1.55T	40.1	32.7	49.8	16.7	57.6
Axial-DeepLab-L	✓	44.9M	39.35T	41.1	33.4	51.3	17.2	58.4

### 3.4.4 Cityscapes

**Val set.** In Table 3.5, we report our Cityscapes validation set results. Without using extra data (*i.e.*, only Cityscapes fine annotation), our Axial-DeepLab achieves 65.1% PQ, which is 1% better than the current best bottom-up Panoptic-DeepLab [78] and 3.1% better than proposal-based AdaptIS [88]. When using extra data (*e.g.*, Mapillary Vistas [74]), our *multi-scale* Axial-DeepLab-XL attains 68.5% PQ, 1.5% better than Panoptic-DeepLab [78] and 3.5% better than Seamless [81]. Our instance segmentation and semantic segmentation results are respectively 1.7% and 1.5% better than Panoptic-DeepLab [78].

**Test set.** Table 3.6 shows our test set results. Without extra data, Axial-DeepLab-XL attains 62.8% PQ, setting a new state-of-the-art result. Our model further achieves 66.6% PQ, 39.6%

**Table 3.5.** Cityscapes val set. **MS:** Multi-scale inputs. **MV:** Mapillary Vistas

Method	Extra Data	MS	PQ	AP	mIoU
AdaptIS [88]		✓	62.0	36.3	79.2
SSAP [90]		✓	61.1	37.3	-
Panoptic-DeepLab [78]			63.0	35.3	80.5
Panoptic-DeepLab [78]		✓	64.1	38.5	<b>81.5</b>
Axial-DeepLab-L			63.9	35.8	81.0
Axial-DeepLab-L		✓	64.7	37.9	<b>81.5</b>
Axial-DeepLab-XL			64.4	36.7	80.6
Axial-DeepLab-XL		✓	<b>65.1</b>	<b>39.0</b>	81.1
SpatialFlow [125]	COCO	✓	62.5	-	-
Seamless [81]	MV		65.0	-	80.7
Panoptic-DeepLab [78]	MV		65.3	38.8	82.5
Panoptic-DeepLab [78]	MV	✓	67.0	42.5	83.1
Axial-DeepLab-L	MV		66.5	40.2	83.2
Axial-DeepLab-L	MV	✓	67.7	42.9	83.8
Axial-DeepLab-XL	MV		67.8	41.9	84.2
Axial-DeepLab-XL	MV	✓	<b>68.5</b>	<b>44.2</b>	<b>84.6</b>

**Table 3.6.** Cityscapes test set. **C:** Cityscapes coarse annotation. **V:** Cityscapes video. **MV:** Mapillary Vistas

Method	Extra Data	PQ	AP	mIoU
GFF-Net [130]		-	-	82.3
Zhu <i>et al.</i> [131]	C, V, MV	-	-	83.5
AdaptIS [88]		-	32.5	-
UPSNet [86]	COCO	-	33.0	-
PANet [132]	COCO	-	36.4	-
PolyTransform [133]	COCO	-	40.1	
SSAP [90]		58.9	32.7	-
Li <i>et al.</i> [87]		61.0	-	-
Panoptic-DeepLab [78]		62.3	34.6	79.4
TASCNet [83]	COCO	60.7	-	-
Seamless [81]	MV	62.6	-	-
Li <i>et al.</i> [87]	COCO	63.3	-	-
Panoptic-DeepLab [78]	MV	65.5	39.0	84.2
Axial-DeepLab-L		62.7	33.3	79.5
Axial-DeepLab-XL		62.8	34.0	79.9
Axial-DeepLab-L	MV	65.6	38.1	83.1
Axial-DeepLab-XL	MV	<b>66.6</b>	39.6	84.1

AP, and 84.1% mIoU with Mapillary Vistas pretraining. Note that Panoptic-DeepLab [78] adopts the trick of output stride 8 during inference on test set, making their M-Adds comparable to our XL models.

### 3.4.5 Ablation Studies

We perform ablation studies on Cityscapes validation set.

**Importance of Position-Sensitivity and Axial-Attention.** In Table 3.1, we experiment with attention models on ImageNet. In this ablation study, we transfer them to Cityscapes segmentation tasks. As shown in Table 3.7, all variants outperform ResNet-50 [10]. Position-sensitive attention performs better than previous self-attention [71], which aligns with ImageNet results in Table 3.1. However, employing axial-attention, which is on-par with position-sensitive attention on ImageNet, gives more than 1% boosts on all three segmentation tasks (in PQ, AP, and mIoU), without ASPP, and with fewer parameters and M-Adds, suggesting that the ability to encode long range context of axial-attention significantly improves the performance on segmentation tasks with large input images.

**Table 3.7.** Ablating self-attention variants on Cityscapes val set. **ASPP:** Atrous spatial pyramid pooling. **PS:** Our position-sensitive self-attention

Backbone	ASPP	PS	Params	M-Adds	PQ	AP	mIoU
ResNet-50 [10] (our impl.)			24.8M	374.8B	58.1	30.0	73.3
ResNet-50 [10] (our impl.)	✓		30.0M	390.0B	59.8	32.6	77.8
Attention [71] (our impl.)			17.3M	317.7B	58.7	31.9	75.8
Attention [71] (our impl.)	✓		22.5M	332.9B	60.9	30.0	78.2
PS-Attention		✓	17.3M	326.7B	59.9	32.2	76.3
PS-Attention	✓	✓	22.5M	341.9B	<b>61.5</b>	<b>33.1</b>	<b>79.1</b>
Axial-DeepLab-S		✓	<b>12.1M</b>	<b>220.8B</b>	<b>62.6</b>	<b>34.9</b>	<b>80.5</b>
Axial-DeepLab-M		✓	25.9M	419.6B	63.1	35.6	80.3
Axial-DeepLab-L		✓	44.9M	687.4B	63.9	35.8	81.0
Axial-DeepLab-XL		✓	173.0M	2446.8B	64.4	36.7	80.6

**Importance of Axial-Attention Span:** In Table 3.8, we vary the span  $m$  (*i.e.*, spatial extent of local regions in an axial block), without ASPP. We observe that a larger span consistently improves the performance at marginal costs.

**Table 3.8.** Varying axial-attention span on Cityscapes val set

Backbone	Span	Params	M-Adds	PQ	AP	mIoU
ResNet-101	-	43.8M	530.0B	59.9	31.9	74.6
Axial-ResNet-L	$5 \times 5$	44.9M	617.4B	59.1	31.3	74.5
Axial-ResNet-L	$9 \times 9$	44.9M	622.1B	61.2	31.1	77.6
Axial-ResNet-L	$17 \times 17$	44.9M	631.5B	62.8	34.0	79.5
Axial-ResNet-L	$33 \times 33$	44.9M	650.2B	63.8	35.9	80.2
Axial-ResNet-L	$65 \times 65$	44.9M	687.4B	<b>64.2</b>	<b>36.3</b>	<b>80.6</b>

## 3.5 Conclusion and Discussion

In this chapter, we have shown the effectiveness of proposed position-sensitive axial-attention on image classification and segmentation tasks. On ImageNet, our Axial-ResNet, formed by stacking axial-attention blocks, achieves state-of-the-art results among stand-alone self-attention models. We further convert Axial-ResNet to Axial-DeepLab for bottom-up segmentation tasks, and also show state-of-the-art performance on several benchmarks, including COCO, Mapillary Vistas, and Cityscapes. We hope our promising results could establish that axial-attention is an effective building block for modern computer vision models.

Our method bears a similarity to decoupled convolution [134], which factorizes a depthwise convolution [120], [127], [135] to a column convolution and a row convolution. This operation could also theoretically achieve a large receptive field, but its convolutional template matching nature limits the capacity of modeling multi-scale interactions. Another related method is deformable convolution [124], [136], [137], where each point attends to a few points dynamically on an image. However, deformable convolution does not make

use of key-dependent positional bias or content-based relation. In addition, axial-attention propagates information densely, and more efficiently along the height- and width-axis sequentially.

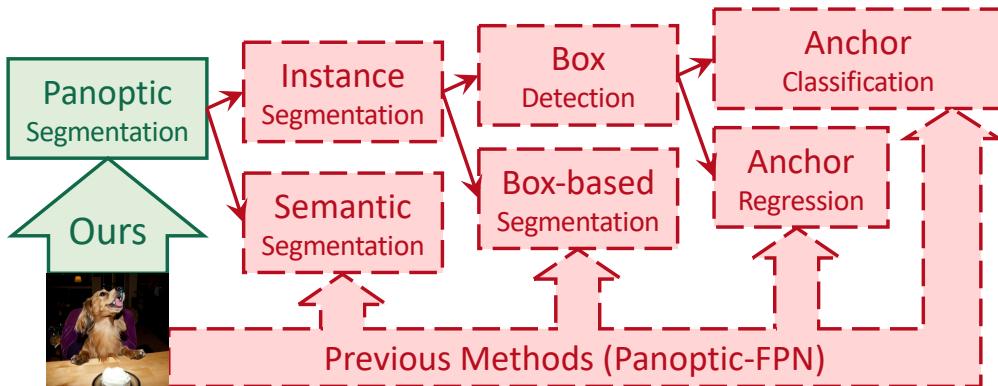
Although our axial-attention model saves M-Adds, it runs slower than convolutional counterparts, as also observed by [71]. This is due to the lack of specialized kernels on various accelerators for the time being. This might well be improved if the community considers axial-attention as a plausible direction.

# Chapter 4

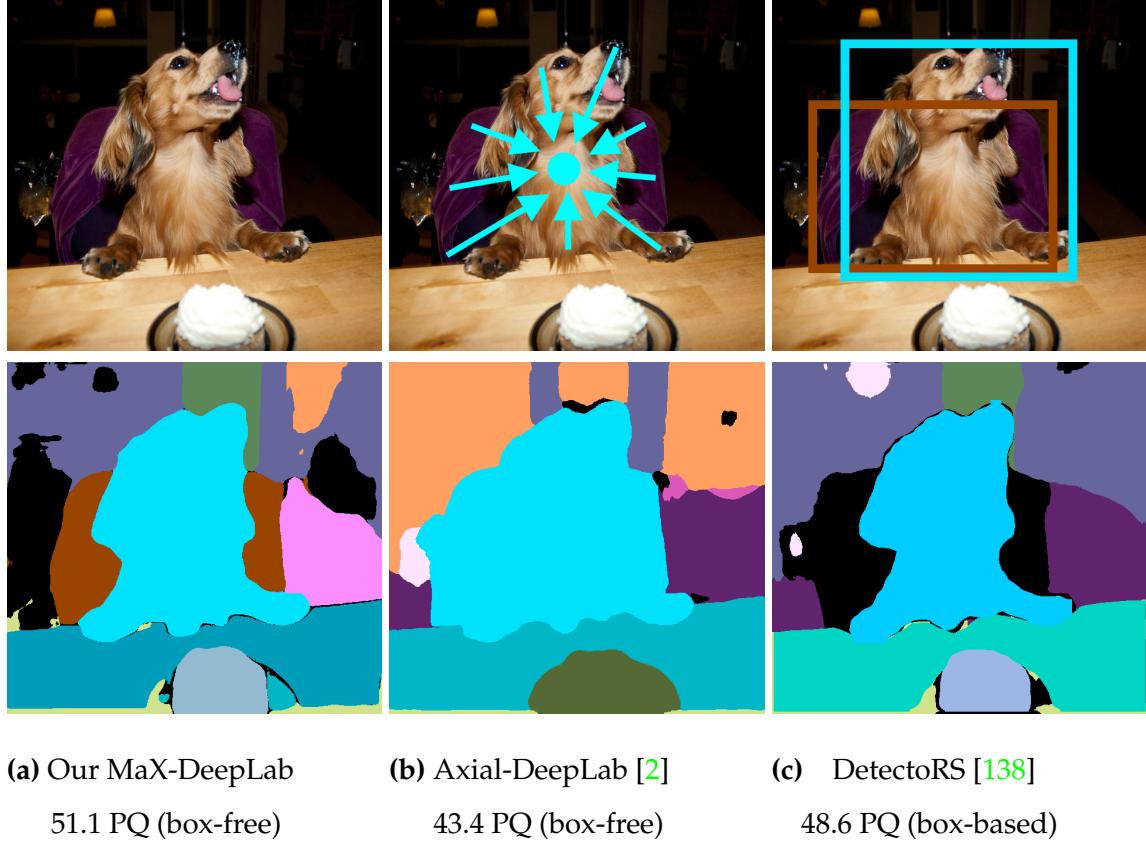
## MaX-DeepLab: End-to-End Panoptic Segmentation with Mask Transformers

This chapter extends the attention mechanism from pixel space to object mask space.

### 4.1 Introduction



**Figure 4.1.** Our method predicts panoptic segmentation masks directly from images, while previous methods (Panoptic-FPN as an example) rely on a tree of surrogate sub-tasks. Panoptic segmentation masks are obtained by merging semantic and instance segmentation results. Instance segmentation is further decomposed into box detection and box-based segmentation, while box detection is achieved by anchor regression and anchor classification.



**Figure 4.2.** A case study for our method and state-of-the-art *box-free* and *box-based* methods.

(a) Our end-to-end MaX-DeepLab correctly segments a dog sitting on a chair. (b) Axial-DeepLab [2] relies on a surrogate sub-task of regressing object center offsets [78]. It fails because the centers of the dog and the chair are close to each other. (c) DetectoRS [138] classifies object bounding boxes, instead of masks, as a surrogate sub-task. It filters out the chair mask because the chair bounding box has a low confidence.

The goal of panoptic segmentation [76] is to predict a set of non-overlapping masks along with their corresponding class labels. Modern panoptic segmentation methods address this mask prediction problem by approximating the target task with multiple surrogate sub-tasks. For example, Panoptic-FPN [80] adopts a ‘box-based pipeline’ with three levels of surrogate sub-tasks, as demonstrated in a tree structure in Figure 4.1. Each level of this proxy tree involves manually-designed modules, such as anchors [139], box assignment rules [140], non-maximum suppression (NMS) [141], thing-stuff merging [86], *etc.* Although there are good solutions [55], [79], [139] to individual surrogate sub-tasks and modules, undesired artifacts are introduced when these sub-tasks fit into a pipeline for panoptic segmentation, especially in the challenging conditions (Figure 4.2).

Method	Anchor -Free	Center -Free	NMS -Free	Merge -Free	Box -Free
Panoptic-FPN [80]	✗	✓	✗	✗	✗
UPSNet [86]	✗	✓	✗	✓	✗
DETR [142]	✓	✓	✓	✓	✗
Axial-DeepLab [2]	✓	✗	✗	✗	✓
MaX-DeepLab	✓	✓	✓	✓	✓

**Table 4.1.** Our end-to-end MaX-DeepLab dispenses with these common hand-designed components necessary for existing methods.

Recent work on panoptic segmentation attempted to simplify this box-based pipeline. For example, UPSNet [86] proposes a parameter-free panoptic head, permitting back-propagation to both semantic and instance segmentation modules. Recently, DETR [142] presents an end-to-end approach for box detection, which is used to replace detectors in panoptic segmentation, but the whole training process of DETR still relies heavily on the box detection task.

Another line of work made efforts to completely remove boxes from the pipeline, which aligns better with the mask-based definition of panoptic segmentation. The state-of-the-art method in this regime, Axial-DeepLab [2], along with other box-free methods [78], [89],

[143], predicts pixel-wise offsets to pre-defined instance centers. But this center-based surrogate sub-task makes it challenging to deal with highly deformable objects, or nearby objects with close centers. As a result, box-free methods do not perform as well as box-based methods on the challenging COCO dataset [15].

In this chapter, we streamline the panoptic segmentation pipeline with an end-to-end approach. Inspired by DETR [142], our model *directly* predicts a set of non-overlapping masks and their corresponding semantic labels with a mask transformer. The output masks and classes are optimized with a panoptic quality (PQ) style objective. Specifically, inspired by the definition of PQ [76], we define a similarity metric between two class-labeled masks as the multiplication of their mask similarity and their class similarity. Our model is trained by maximizing this similarity between ground truth masks and predicted masks via one-to-one bipartite matching [142], [144], [145]. This direct modeling of panoptic segmentation enables end-to-end training and inference, removing those hand-coded priors that are necessary in existing box-based and box-free methods (Table 4.1). Our method is dubbed MaX-DeepLab for extending Axial-DeepLab with a **Mask Xformer**.

In companion with direct training and inference, we equip our mask transformer with a novel architecture. Instead of stacking a traditional transformer [61], [142] on top of a Convolutional Neural Network (CNN) [49], we propose a dual-path framework for combining CNNs with transformers. Specifically, we enable any CNN layer to read and write a global memory, using our dual-path transformer block. This block supports all types of attention between the CNN-path and the memory-path, including memory-path self-attention ( $M2M$ ), pixel-path axial self-attention ( $P2P$ ), memory-to-pixel attention ( $M2P$ ), and finally pixel-to-memory attention ( $P2M$ ). The transformer block can be inserted anywhere in a CNN, enabling communication with the global memory at any layer. Besides this communication module, our MaX-DeepLab employs a stacked-hourglass-style decoder [37], [38], [146]. The decoder aggregates multi-scale features into a high resolution output, which is then multiplied with the global memory feature, to form our

mask set prediction. The classes for the masks are predicted with another branch of the mask transformer.

We evaluate MaX-DeepLab on one of the most challenging panoptic segmentation datasets, COCO [15], against the state-of-the-art box-free method, Axial-DeepLab [2], and state-of-the-art box-based method, DetectoRS [147] (Figure 4.2). Our MaX-DeepLab, *without* test time augmentation (TTA), achieves the state-of-the-art result of 51.3% PQ on the test-dev set. This result surpasses Axial-DeepLab (with TTA) by 7.1% PQ in the box-free regime, and outperforms DetectoRS (with TTA) by 1.7% PQ, bridging the gap between box-based and box-free methods for the first time. For a fair comparison with DETR [142], we also evaluate a lightweight model, MaX-DeepLab-S, that matches the number of parameters and M>Adds of DETR. We observe that MaX-DeepLab-S outperforms DETR by 3.3% PQ on the val set and 3.0% PQ on the test-dev set. In addition, we perform extensive ablation studies and analyses on our end-to-end formulation, model scaling, dual-path architectures, and our loss functions. We also notice that the extra-long training schedule of DETR [142] is not necessary for MaX-DeepLab.

To summarize, our contributions are four-fold:

- MaX-DeepLab is the first end-to-end model for panoptic segmentation, inferring masks and classes directly without hand-coded priors like object centers or boxes.
- We propose a training objective that optimizes a PQ-style loss function via a PQ-style bipartite matching between predicted masks and ground truth masks.
- Our dual-path transformer enables CNNs to read and write a global memory at any layer, providing a new way of combining transformers with CNNs.
- MaX-DeepLab closes the gap between box-based and box-free methods and sets a new state-of-the-art on COCO, even without using test time augmentation.

## 4.2 Related Work

**Transformers.** Transformers [61], first introduced for neural machine translation, have advanced the state-of-the-art in many natural language processing tasks [6], [105], [106]. Attention [102], as the core component of Transformers, was developed to capture both correspondence of tokens across modalities [102] and long-range interactions in a single context (self-attention) [61], [148]. Later, the complexity of transformer attention has been reduced [149], [150], by introducing local [151] or sparse attention [152], together with a global memory [153]–[156]. The global memory, which inspires our dual-path transformer, recovers long-range context by propagating information globally.

Transformer and attention have been applied to computer vision as well, by combining non-local modules [69], [108] with CNNs or by applying self-attention only [2], [71], [72]. Both classes of methods have boosted various vision tasks such as image classification [2], [66], [71], [72], [107], [109], [157], object detection [67], [69], [71], [142], [158], [159], semantic segmentation [68], [110]–[112], [160], [161], video recognition [69], [109], image generation [73], [104], and panoptic segmentation [2]. It is worth mentioning that DETR [142] stacked a transformer on top of a CNN for end-to-end object detection.

**Box-based panoptic segmentation.** Most panoptic segmentation models, such as Panoptic FPN [80], follow a box-based approach that detects object bounding boxes and predicts a mask for each box, usually with a Mask R-CNN [79] and FPN [41]. Then, the instance segments (‘thing’) and semantic segments (‘stuff’) [82] are fused by merging modules [81], [83]–[85], [162] to generate panoptic segmentation. For example, UPSNet [86] developed a parameter-free panoptic head, which facilitates unified training and inference [87]. Recently, DETR [142] extended box-based methods with its transformer-based end-to-end detector. And DetectoRS [138] advanced the state-of-the-art with recursive feature pyramid and switchable atrous convolution.

**Box-free panoptic segmentation.** Contrary to box-based approaches, box-free methods typically start with semantic segments [48], [55], [101]. Then, instance segments are obtained by grouping ‘thing’ pixels with various methods, such as instance center regression [77], [89], [98]–[100], Watershed transform [93]–[95], Hough-voting [93], [96], [97], or pixel affinity [88], [90]–[93]. Recently, Axial-DeepLab [2] advanced the state-of-the-art by equipping Panoptic-DeepLab [78] with a fully axial-attention [73] backbone. In this chapter, we extend Axial-DeepLab with a mask transformer for end-to-end panoptic segmentation.

## 4.3 Method

In this section, we describe how MaX-DeepLab directly predicts class-labeled masks for panoptic segmentation, followed by the PQ-style loss used to train the model. Then, we introduce our dual-path transformer architecture as well as the auxiliary losses that are helpful in training.

### 4.3.1 MaX-DeepLab Formulation

The goal of panoptic segmentation is to segment the image  $I \in \mathbb{R}^{H \times W \times 3}$  into a set of class-labeled masks:

$$\{y_i\}_{i=1}^K = \{(m_i, c_i)\}_{i=1}^K. \quad (4.1)$$

The  $K$  ground truth masks  $m_i \in \{0, 1\}^{H \times W}$  do not overlap with each other, *i.e.*,  $\sum_{i=1}^K m_i \leq 1^{H \times W}$ , and  $c_i$  denotes the ground truth class label of mask  $m_i$ .

Our MaX-DeepLab directly predicts outputs in the exact same form as the ground truth. MaX-DeepLab segments the image  $I$  into a fixed-size set of class-labeled masks:

$$\{\hat{y}_i\}_{i=1}^N = \{(\hat{m}_i, \hat{p}_i(c))\}_{i=1}^N. \quad (4.2)$$

The constant size  $N$  of the set is much larger than the typical number of masks in an

image [142]. The predicted masks  $\hat{m}_i \in [0, 1]^{H \times W}$  are softly exclusive to each other, *i.e.*,  $\sum_{i=1}^N \hat{m}_i = 1^{H \times W}$ , and  $\hat{p}_i(c)$  denotes the probability of assigning class  $c$  to mask  $\hat{m}_i$ . Possible classes  $\mathbb{C} \ni c$  include thing classes, stuff classes, and a  $\emptyset$  class (no object). In this way, MaX-DeepLab deals with thing and stuff classes in a unified manner, removing the need for merging operators.

**Simple inference.** End-to-end inference of MaX-DeepLab is enabled by adopting the same formulation for both ground truth definition and model prediction. As a result, the final panoptic segmentation prediction is obtained by simply performing argmax twice. Specifically, the first argmax predicts a class label for each mask:

$$\hat{c}_i = \arg \max_c \hat{p}_i(c) . \quad (4.3)$$

And the other argmax assigns a mask-ID  $\hat{z}_{h,w}$  to each pixel:

$$\begin{aligned} \hat{z}_{h,w} &= \arg \max_i \hat{m}_{i,h,w} , \\ \forall h \in \{1, 2, \dots, H\}, \quad \forall w \in \{1, 2, \dots, W\} . \end{aligned} \quad (4.4)$$

In practice, we filter each argmax with a confidence threshold – Masks or pixels with a low confidence are removed as described in Section 4.4. In this way, MaX-DeepLab infers panoptic segmentation directly, dispensing with common manually-designed post-processing, *e.g.*, NMS and thing-stuff merging in almost all previous methods [80], [86]. Besides, MaX-DeepLab does not rely on hand-crafted priors such as anchors, object boxes, or instance mass centers, *etc.*

### 4.3.2 PQ-Style Loss

In addition to simple inference, MaX-DeepLab enables end-to-end training as well. In this section, we introduce how we train MaX-DeepLab with our PQ-style loss, which draws inspiration from the definition of *panoptic quality* (PQ) [76]. This evaluation metric of panoptic segmentation, PQ, is defined as the multiplication of a *recognition quality* (RQ)

term and a *segmentation quality* (SQ) term:

$$PQ = RQ \times SQ. \quad (4.5)$$

Based on this decomposition of  $PQ$ , we design our objective in the same manner: First, we define a  $PQ$ -style similarity metric between a class-labeled ground truth mask and a predicted mask. Next, we show how we match a predicted mask to each ground truth mask with this metric, and finally how to optimize our model with the same metric.

**Mask similarity metric.** Our mask similarity metric  $\text{sim}(\cdot, \cdot)$  between a class-labeled ground truth mask  $y_i = (m_i, c_i)$  and a prediction  $\hat{y}_j = (\hat{m}_j, \hat{p}_j(c))$  is defined as

$$\text{sim}(y_i, \hat{y}_j) = \underbrace{\hat{p}_j(c_i)}_{\approx RQ} \times \underbrace{\text{Dice}(m_i, \hat{m}_j)}_{\approx SQ}, \quad (4.6)$$

where  $\hat{p}_j(c_i) \in [0, 1]$  is the probability of predicting the correct class (recognition quality) and  $\text{Dice}(m_i, \hat{m}_j) \in [0, 1]$  is the Dice coefficient between a predicted mask  $\hat{m}_j$  and a ground truth  $m_i$  (segmentation quality). The two terms are multiplied together, analogous to the decomposition of  $PQ$ .

This mask similarity metric has a lower bound of 0, which means either the class prediction is incorrect, OR the two masks do not overlap with each other. The upper bound, 1, however, is only achieved when the class prediction is correct AND the mask is perfect. The AND gating enables this metric to serve as a good optimization objective for both model training and mask matching.

**Mask matching.** In order to assign a predicted mask to each ground truth, we solve a one-to-one bipartite matching problem between the prediction set  $\{\hat{y}_i\}_{i=1}^N$  and the ground truth set  $\{y_i\}_{i=1}^K$ . Formally, we search for a permutation of  $N$  elements  $\sigma \in \mathfrak{S}_N$  that best assigns the predictions to achieve the maximum total similarity to the ground truth:

$$\hat{\sigma} = \arg \max_{\sigma \in \mathfrak{S}_N} \sum_{i=1}^K \text{sim}(y_i, \hat{y}_{\sigma(i)}). \quad (4.7)$$

The optimal assignment is computed efficiently with the Hungarian algorithm [144], following prior work [142], [145]. We refer to the  $K$  matched predictions as positive masks

which will be optimized to predict the corresponding ground truth masks and classes. The  $(N - K)$  masks left are negatives, which should predict the  $\emptyset$  class (no object).

Our one-to-one matching is similar to DETR [142], but with a different purpose: DETR allows only one positive match in order to remove duplicated boxes in the absence of NMS, while in our case, duplicated or overlapping masks are precluded by design. But in our case, assigning multiple predicted masks to one ground truth mask is problematic too, because multiple masks cannot possibly be optimized to fit a single ground truth mask at the same time. In addition, our one-to-one matching is consistent with the PQ metric, where only one predicted mask can theoretically match (*i.e.*, have an IoU over 0.5) with each ground truth mask.

**PQ-style loss.** Given our mask similarity metric and the mask matching process based on this metric, it is straight forward to optimize model parameters  $\theta$  by maximizing this same similarity metric over matched (*i.e.*, positive) masks:

$$\max_{\theta} \sum_{i=1}^K \text{sim}(y_i, \hat{y}_{\hat{\sigma}(i)}) \Leftrightarrow \max_{\theta, \sigma \in \mathfrak{S}_N} \sum_{i=1}^K \text{sim}(y_i, \hat{y}_{\sigma(i)}). \quad (4.8)$$

Substituting the similarity metric (Equation (4.6)) gives our PQ-style objective  $\mathcal{O}_{\text{PQ}}^{\text{pos}}$  to be maximized for positive masks:

$$\max_{\theta} \mathcal{O}_{\text{PQ}}^{\text{pos}} = \sum_{i=1}^K \underbrace{\hat{p}_{\hat{\sigma}(i)}(c_i)}_{\approx \text{RQ}} \times \underbrace{\text{Dice}(m_i, \hat{m}_{\hat{\sigma}(i)})}_{\approx \text{SQ}}. \quad (4.9)$$

In practice, we rewrite  $\mathcal{O}_{\text{PQ}}^{\text{pos}}$  into two common loss terms by applying the product rule of gradient and then changing a probability  $\hat{p}$  to a log probability  $\log \hat{p}$ . The change from  $\hat{p}$  to  $\log \hat{p}$  aligns with the common cross-entropy loss and scales gradients better in practice for optimization:

$$\begin{aligned} \mathcal{L}_{\text{PQ}}^{\text{pos}} &= \sum_{i=1}^K \underbrace{\hat{p}_{\hat{\sigma}(i)}(c_i)}_{\text{weight}} \cdot \underbrace{[-\text{Dice}(m_i, \hat{m}_{\hat{\sigma}(i)})]}_{\text{Dice loss}} \\ &\quad + \sum_{i=1}^K \underbrace{\text{Dice}(m_i, \hat{m}_{\hat{\sigma}(i)})}_{\text{weight}} \cdot \underbrace{[-\log \hat{p}_{\hat{\sigma}(i)}(c_i)]}_{\text{Cross-entropy loss}}, \end{aligned} \quad (4.10)$$

where the loss weights are constants (*i.e.*, no gradient is passed to them). This reformulation provides insights by bridging our objective with common loss functions: Our PQ-style loss is equivalent to optimizing a dice loss weighted by the class correctness and optimizing a cross-entropy loss weighted by the mask correctness. The logic behind this loss is intuitive: we want *both* of the mask and class to be correct at the same time. For example, if a mask is far off the target, it is a false negative anyway, so we disregard its class. This intuition aligns with the down-weighting of class losses for wrong masks, and vice versa.

Apart from the  $\mathcal{L}_{\text{PQ}}^{\text{pos}}$  for positive masks, we define a cross-entropy term  $\mathcal{L}_{\text{PQ}}^{\text{neg}}$  for negative (unmatched) masks:

$$\mathcal{L}_{\text{PQ}}^{\text{neg}} = \sum_{i=K+1}^N \left[ -\log \hat{p}_{\hat{\sigma}(i)}(\emptyset) \right]. \quad (4.11)$$

This term trains the model to predict  $\emptyset$  for negative masks. We balance the two terms by  $\alpha$ , as a common practice to weight positive and negative samples [163]:

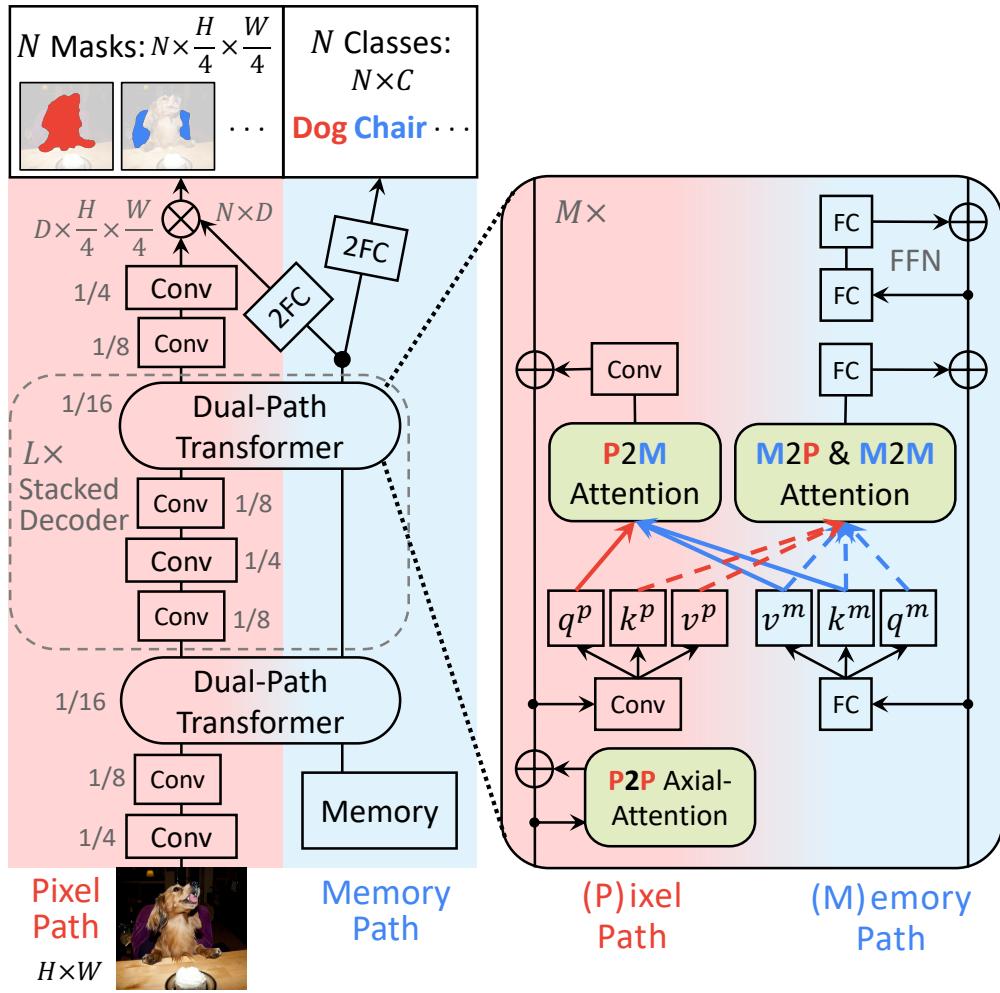
$$\mathcal{L}_{\text{PQ}} = \alpha \mathcal{L}_{\text{PQ}}^{\text{pos}} + (1 - \alpha) \mathcal{L}_{\text{PQ}}^{\text{neg}}, \quad (4.12)$$

where  $\mathcal{L}_{\text{PQ}}$  denotes our final PQ-style loss.

### 4.3.3 MaX-DeepLab Architecture

As shown in Figure 4.3, MaX-DeepLab architecture includes a dual-path transformer, a stacked decoder, and output heads that predict the masks and classes.

**Dual-path transformer.** Instead of stacking a transformer on top of a CNN [142], we integrate the transformer and the CNN in a dual-path fashion, with bidirectional communication between the two paths. Specifically, we augment a 2D *pixel*-based CNN with a 1D global *memory* of size  $N$  (*i.e.*, the total number of predictions) and propose a transformer block as a drop-in replacement for any CNN block or an add-on for a pretrained CNN block. Our transformer block enables all four possible types of communication between the 2D pixel-path CNN and the 1D memory-path: (1) the traditional memory-to-pixel (M2P) attention, (2) memory-to-memory (M2M) self-attention, (3) pixel-to-memory (P2M)



**Figure 4.3.** (a) An image and a global memory are fed into a dual-path transformer, which directly predicts a set of masks and classes (residual connections omitted). (b) A dual-path transformer block is equipped with all 4 types of attention between the two paths.

feedback attention that allows pixels to read from the memory, and (4) pixel-to-pixel (*P2P*) self-attention, implemented as axial-attention blocks [2], [68], [73]. We select axial-attention [2] rather than global 2D attention [66], [69], [142] for efficiency on high resolution feature maps. One could optionally approximate the pixel-to-pixel self-attention with a convolutional block that only allows local communication. This transformer design with a memory path besides the main CNN path is termed dual-path transformer. Unlike previous work [142], it allows transformer blocks to be inserted anywhere in the backbone at any resolution. In addition, the *P2M* feedback attention enables the pixel-path CNN to refine its feature given the memory-path features that encode mask information.

Formally, given a 2D input feature  $x^p \in \mathbb{R}^{\hat{H} \times \hat{W} \times d_{in}}$  with height  $\hat{H}$ , width  $\hat{W}$ , channels  $d_{in}$ , and a 1D global memory feature  $x^m \in \mathbb{R}^{N \times d_{in}}$  with length  $N$  (*i.e.*, the size of the prediction set). We compute pixel-path queries  $q^p$ , keys  $k^p$ , and values  $v^p$ , by learnable linear projections of the pixel-path feature map  $x^p$  at each pixel. Similarly,  $q^m, k^m, v^m$  are computed from  $x^m$  with another set of projection matrices. The query (key) and value channels are  $d_q$  and  $d_v$ , for both paths. Then, the output of feedback attention (*P2M*),  $y_a^p \in \mathbb{R}^{d_{out}}$ , at pixel position  $a$ , is computed as

$$y_a^p = \sum_{n=1}^N \text{softmax}_n(q_a^p \cdot k_n^m) v_n^m, \quad (4.13)$$

where the  $\text{softmax}_n$  denotes a softmax function applied to the whole memory of length  $N$ . Similarly, the output of memory-to-pixel (*M2P*) and memory-to-memory (*M2M*) attention  $y_b^m \in \mathbb{R}^{d_{out}}$ , at memory position  $b$ , is

$$\begin{aligned} y_b^m &= \sum_{n=1}^{\hat{H}\hat{W}+N} \text{softmax}_n(q_b^m \cdot k_n^{pm}) v_n^{pm}, \\ k^{pm} &= \begin{bmatrix} k^p \\ k^m \end{bmatrix}, \quad v^{pm} = \begin{bmatrix} v^p \\ v^m \end{bmatrix}, \end{aligned} \quad (4.14)$$

where a single softmax is performed over the concatenated dimension of size  $(\hat{H}\hat{W}, +N)$ , inspired by ETC [156].

**Stacked decoder.** Unlike previous work [2], [78] that uses a light-weight decoder, we explore stronger hourglass-style stacked decoders [37], [38], [146]. As shown in Figure 4.3, our decoder is stacked  $L$  times, traversing output strides (4, 8, and 16 [60], [101]) multiple times. At each decoding resolution, features are fused by simple summation after bilinear resizing. Then, convolutional blocks or transformer blocks are applied, before the decoder feature is sent to the next resolution. This stacked decoder is similar to feature pyramid networks [41], [132], [138], [164] designed for pyramidal anchor predictions [22], but our purpose here is only to aggregate multi-scale features, *i.e.*, intermediate pyramidal features are not directly used for prediction.

**Output heads.** From the memory feature of length  $N$ , we predict mask classes  $\hat{p}(c) \in \mathbb{R}^{N \times |\mathcal{C}|}$  with two fully-connected layers (2FC) and a softmax. Another 2FC head predicts mask feature  $f \in \mathbb{R}^{N \times D}$ . Similarly, we employ two convolutions (2Conv) to produce a normalized feature  $g \in \mathbb{R}^{D \times \frac{H}{4} \times \frac{W}{4}}$  from the decoder output at stride 4. Then, our mask prediction  $\hat{m}$  is simply the multiplication of transformer feature  $f$  and decoder feature  $g$ :

$$\hat{m} = \text{softmax}_N(f \cdot g) \in \mathbb{R}^{N \times \frac{H}{4} \times \frac{W}{4}}. \quad (4.15)$$

In practice, we use batch norm [123] on  $f$  and  $(f \cdot g)$  to avoid deliberate initialization, and we bilinear upsample the mask prediction  $\hat{m}$  to the original image resolution. Finally, the combination  $\{(\hat{m}_i, \hat{p}_i(c))\}_{i=1}^N$  is our mask transformer output to generate panoptic results as introduced in Section 4.3.1.

Our mask prediction head is inspired by CondInst [165] and SOLOv2 [166], which extend dynamic convolution [167], [168] to instance segmentation. However, unlike our end-to-end method, these methods require hand-designed object centers and assignment rules for instance segmentation, and a thing-stuff merging module for panoptic segmentation.

### 4.3.4 Auxiliary Losses

In addition to the PQ-style loss (Section 4.3.2), we find it beneficial to incorporate auxiliary losses in training. Specifically, we propose a pixel-wise instance discrimination loss that helps cluster decoder features into instances. We also use a per-pixel mask-ID cross-entropy loss that classifies each pixel into  $N$  masks, and a semantic segmentation loss. Our total loss function thus consists of the PQ-style loss  $\mathcal{L}_{\text{PQ}}$  and these three auxiliary losses.

**Instance discrimination.** We use a per-pixel instance discrimination loss to help the learning of the feature map  $g \in \mathbb{R}^{D \times \frac{H}{4} \times \frac{W}{4}}$ . Given a downsampled ground truth mask  $m_i \in \{0, 1\}^{\frac{H}{4} \times \frac{W}{4}}$ , we first compute a normalized feature embedding  $t_{i,:} \in \mathbb{R}^D$  for each annotated mask by averaging the feature vectors  $g_{:,h,w}$  inside the mask  $m_i$ :

$$t_{i,:} = \frac{\sum_{h,w} m_{i,h,w} \cdot g_{:,h,w}}{\|\sum_{h,w} m_{i,h,w} \cdot g_{:,h,w}\|}, \quad i = 1, 2, \dots, K. \quad (4.16)$$

This gives us  $K$  instance embeddings  $\{t_{i,:}\}_{i=1}^K$  representing  $K$  ground truth masks. Then, we let each pixel feature  $g_{:,h,w}$  perform an instance discrimination task, *i.e.*, each pixel should correctly identify which mask embedding (out of  $K$ ) it belongs to, as annotated by the ground truth masks. The contrastive loss at a pixel  $(h, w)$  is written as:

$$\mathcal{L}_{h,w}^{\text{InstDis}} = -\log \frac{\sum_{i=1}^K m_{i,h,w} \exp(t_{i,:} \cdot g_{:,h,w} / \tau)}{\sum_{i=1}^K \exp(t_{i,:} \cdot g_{:,h,w} / \tau)}, \quad (4.17)$$

where  $\tau$  denotes the temperature, and note that  $m_{i,h,w}$  is non-zero only when pixel  $(h, w)$  belongs to the ground truth mask  $m_i$ . In practice, this per-pixel loss is applied to all instance pixels in an image, encouraging features from the same instance to be similar and features from different instances to be distinct, in a contrastive fashion, which is exactly the property required for instance segmentation.

Our instance discrimination loss is inspired by previous works [169]–[174]. However, they discriminate instances either unsupervisedly or with image classes [174], whereas we perform a pixel-wise instance discrimination task, as annotated by panoptic segmentation ground truth.

**Mask-ID cross-entropy.** In Equation (4.4), we describe how we infer the mask-ID map given our mask prediction. In fact, we can train this per-pixel classification task by applying a cross-entropy loss on it. This is consistent with the literature [142], [175], [176] that uses a cross-entropy loss together with a dice loss [177] to learn better segmentation masks.

**Semantic segmentation.** We also use an auxiliary semantic segmentation loss to help capture per pixel semantic feature. Specifically, we apply a semantic head [78] on top of the backbone if no stacked decoder is used (*i.e.*,  $L = 0$ ). Otherwise, we connect the semantic head to the first decoder output at stride 4, because we find it helpful to separate the final mask feature  $g$  with semantic segmentation.

## 4.4 Experiments

We report our main results on COCO, comparing with state-of-the-art methods. Then, we provide a detailed ablation study on the architecture variants and losses. Finally, we analyze how MaX-DeepLab works with visualizations.

**Technical details.** Most of our default settings follow Axial-DeepLab [2]. Specifically, we train our models with 32 TPU cores for 100k (400k for main results) iterations (54 epochs), a batch size of 64, Radam [118] Lookahead [119], a ‘poly’ schedule learning rate of  $10^{-3}$  ( $3 \times 10^{-4}$  for MaX-DeepLab-L), a backbone learning rate multiplier of 0.1, a weight decay of  $10^{-4}$ , and a drop path rate [178] of 0.2. We resize and pad images to  $641 \times 641$  [2], [78] ( $1025 \times 1025$  for main results) for inference and M-Adds calculation. During inference, we set masks with class confidence below 0.7 to void and filter pixels with mask-ID confidence below 0.4. Finally, following previous work [2], [78], [86], we filter stuff masks with an area limit of 4096 pixels, and instance masks with a limit of 256 pixels. In training, we set our PQ-style loss weight (Equation (4.12), normalized by  $N$ ) to 3.0, with  $\alpha = 0.75$ . Our instance discrimination uses  $\tau = 0.3$ , and a weight of 1.0. We set the mask-ID cross-entropy weight to 0.3, and semantic segmentation weight to 1.0. We use an output size  $N = 128$  and

$D = 128$  channels. We fill the initial memory with learnable weights [142] (more details and architectures in Section C.6).

#### 4.4.1 Main Results

We present our main results on COCO *val* set and *test-dev* set [15], with a small model, MaX-DeepLab-S, and a large model, MaX-DeepLab-L.

MaX-DeepLab-S augments ResNet-50 [10] with axial-attention blocks [2] in the last two stages. After pretaining, we replace the last stage with dual-path transformer blocks and use an  $L = 0$  (not stacked) decoder. We match parameters and M-Adds to DETR-R101 [142], for fair comparison.

MaX-DeepLab-L stacks an  $L = 2$  decoder on top of Wide-ResNet-41 [143], [179], [180]. And we replace all stride 16 residual blocks by our dual-path transformer blocks with wide axial-attention blocks [2]. This large variant is meant to be compared with state-of-the-art results.

**Val set.** In Table 4.2, we report our validation set results and compare with both box-based and box-free panoptic segmentation methods. As shown in the table, our *single-scale* MaX-DeepLab-S already outperforms all other *box-free* methods by a large margin of more than 4.5 % PQ, no matter whether other methods use test time augmentation (TTA, usually flipping and multi-scale) or not. Specifically, it surpasses *single-scale* Panoptic-DeepLab by 8.7% PQ, and *single-scale* Axial-DeepLab by 5.0% PQ with similar M-Adds. We also compare MaX-DeepLab-S with DETR [142], which is based on an end-to-end detector, in a controlled environment of similar number of parameters and M-Adds. Our MaX-DeepLab-S outperforms DETR [142] by 3.3% PQ in this fair comparison. Next, we scale up MaX-DeepLab to a wider variant with stacked decoder, MaX-DeepLab-L. This scaling further improves the *single-scale* performance to 51.1% PQ, outperforming *multi-scale* Axial-DeepLab [2] by 7.2% PQ with similar inference M-Adds.

**Test-dev set.** Our improvements on the *val* set transfers well to the test-dev set, as shown in Table 4.3. On the test-dev set, we are able to compare with more competitive methods and stronger backbones equipped with group convolution [8], [16], deformable convolution [124], or recursive backbone [138], [181], while we do not use these improvements in our model. In the regime of no TTA, our MaX-DeepLab-S outperforms Axial-DeepLab [2] by 5.4% PQ, and DETR [142] by 3.0% PQ. Our MaX-DeepLab-L without TTA further attains 51.3% PQ, surpassing Axial-DeepLab with TTA by 7.1% PQ. This result also outperforms the best box-based method DetectoRS [138] with TTA by 1.7% PQ, closing the large gap between box-based and box-free methods on COCO for the first time. Our MaX-DeepLab sets a new state-of-the-art on COCO, even without using TTA.

Method	Backbone	TTA	Params	M-Addrs	PQ	PQ <sup>Th</sup>	PQ <sup>St</sup>
Box-based panoptic segmentation methods							
Panoptic-FPN [80]	RN-101				40.3	47.5	29.5
UPSNet [86]	RN-50				42.5	48.5	33.4
Detectron2 [147]	RN-101				43.0	-	-
UPSNet [86]	RN-50	✓			43.2	49.1	34.1
DETR [142]	RN-101		61.8M	314B <sup>1</sup>	45.1	50.5	37.0
Box-free panoptic segmentation methods							
Panoptic-DeepLab [78]	X-71 [127]		46.7M	274B	39.7	43.9	33.2
Panoptic-DeepLab [78]	X-71 [127]	✓	46.7M	3081B	41.2	44.9	35.7
Axial-DeepLab-L [2]	AX-L [2]		44.9M	344B	43.4	48.5	35.6
Axial-DeepLab-L [2]	AX-L [2]	✓	44.9M	3868B	43.9	48.6	36.8
MaX-DeepLab-S	MaX-S		61.9M	324B	48.4	53.0	41.5
MaX-DeepLab-L	MaX-L		451M	3692B	51.1	57.0	42.2

**Table 4.2.** COCO val set. **TTA:** Test-time augmentation

<sup>1</sup><https://github.com/facebookresearch/detr>

Method	Backbone	TTA	PQ	PQ <sup>Th</sup>	PQ <sup>St</sup>
Box-based panoptic segmentation methods					
Panoptic-FPN [80]	RN-101		40.9	48.3	29.7
DETR [142]	RN-101		46.0	-	-
UPSNet [86]	DCN-101 [124]	✓	46.6	53.2	36.7
DetectoRS [138]	RX-101 [16]	✓	49.6	57.8	37.1
Box-free panoptic segmentation methods					
Panoptic-DeepLab [78]	X-71 [127], [128]	✓	41.4	45.1	35.9
Axial-DeepLab-L [2]	AX-L [2]		43.6	48.9	35.6
Axial-DeepLab-L [2]	AX-L [2]	✓	44.2	49.2	36.8
MaX-DeepLab-S	MaX-S		49.0	54.0	41.6
MaX-DeepLab-L	MaX-L		51.3	57.2	42.4

**Table 4.3.** COCO test-dev set. **TTA:** Test-time augmentation

#### 4.4.2 Ablation Study

In this subsection, we provide more insights by teasing apart the effects of MaX-DeepLab components on the *val* set. We first define a default baseline setting and then vary each component of it: We augment Wide-ResNet-41 [143], [179], [180] by applying dual-path transformer to all blocks at stride 16, enabling all four types of attention. For faster wall-clock training, we use an  $L = 0$  (not stacked) decoder and approximate *P2P* attention with convolutional blocks.

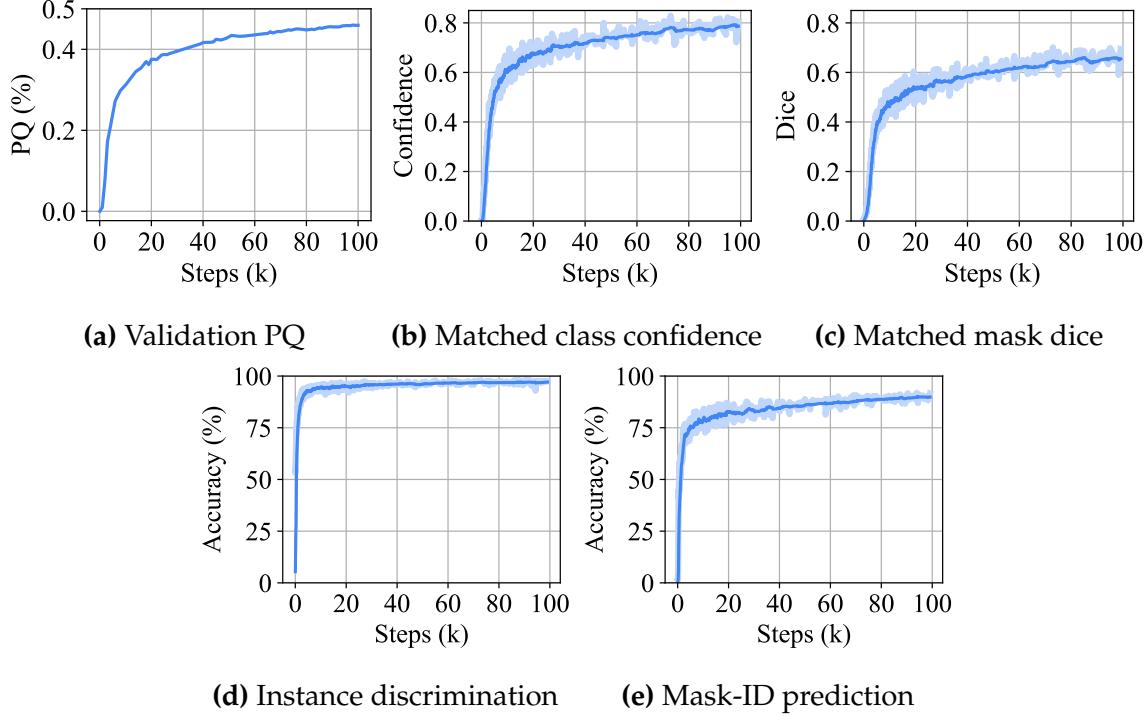
**Scaling.** We first study the scaling of MaX-DeepLab in Table 4.4. We notice that replacing convolutional blocks with axial-attention blocks gives the most improvement. Further changing the input resolution to  $1025 \times 1025$  improves the performance to 49.4% PQ, with a short 100k schedule (54 epochs). Stacking the decoder  $L = 1$  time improves 1.4% PQ, but further scaling to  $L = 2$  starts to saturate. Training with more iterations helps convergence, but we find it not as necessary as DETR which is trained for 500 epochs.

Res	Axial	$L$	Iter	Params	M-Adds	PQ	PQ <sup>Th</sup>	PQ <sup>St</sup>
641	$\times$	0	100k	196M	746B	45.7	49.8	39.4
641	✓	0	100k	277M	881B	47.8	51.9	41.5
1025	$\times$	0	100k	196M	1885B	46.1	50.7	39.1
1025	✓	0	100k	277M	2235B	<b>49.4</b>	<b>54.5</b>	<b>41.8</b>
641	$\times$	1	100k	271M	1085B	47.1	51.6	40.3
641	$\times$	2	100k	347M	1425B	47.5	52.3	40.2
641	$\times$	0	200k	196M	746B	46.9	51.5	40.0
641	$\times$	0	400k	196M	746B	47.7	52.5	40.4

**Table 4.4.** Scaling MaX-DeepLab by using a larger input **Resolution**, replacing convolutional blocks with **Axial**-attention blocks, stacking decoder  $L$  times, and training with more **Iterations**.

P2M	M2M	Stride	Params	M-Adds	PQ	PQ <sup>Th</sup>	PQ <sup>St</sup>
✓	✓	16	196M	746B	45.7	49.8	39.4
	✓	16	188M	732B	45.0	48.9	39.2
✓		16	196M	746B	45.1	49.3	38.9
		16	186M	731B	44.7	48.5	39.0
✓	✓	16 & 8	220M	768B	<b>46.7</b>	<b>51.3</b>	<b>39.7</b>
✓	✓	16 & 8 & 4	234M	787B	46.3	51.1	39.0

**Table 4.5.** Varying transformer **P2M** feedback attention, **M2M** self-attention, and the **Stride** where we apply the transformer.



**Figure 4.4.** Training curves for (a) validation PQ, (b) average class confidence,  $\hat{p}_{\hat{\sigma}(i)}(c_i)$ , of matched masks, (c) average mask dice,  $\text{Dice}(m_i, \hat{m}_{\hat{\sigma}(i)})$ , of matched masks, (d) per-pixel instance discrimination accuracy, and (e) per-pixel mask-ID prediction accuracy.

**Dual-path transformer.** Next, we vary attention types of our dual-path transformer and the stages (strides) where we apply transformer blocks. Note that we always apply *M2P* attention that attaches the transformer to the CNN. And *P2P* attention is already ablated above. As shown in Table 4.5, removing our *P2M* feedback attention causes a drop of 0.7% PQ. On the other hand, we find MaX-DeepLab robust (-0.6% PQ) to the removal of *M2M* self-attention. We attribute this robustness to our non-overlapping mask formulation. Note that DETR [142] relies on *M2M* self-attention to remove duplicated boxes. In addition, it is helpful (+1.0% PQ) to apply transformer blocks to stride 8 also, which is impossible for DETR without our dual-path design. Pushing it further to stride 4 does not show more improvements.

**Loss ablation.** Finally, we ablate our PQ-style loss and auxiliary losses in Table 4.6. We first switch our PQ-style similarity in Equation (4.6) from  $RQ \times SQ$  to  $RQ + SQ$ , which differs in the hungarian matching (Equation (4.7)) and removes dynamic loss weights in Equation (4.10). We observe that  $RQ + SQ$  works reasonably well, but  $RQ \times SQ$  improves 0.8% PQ on top of it, confirming the effect of our PQ-style loss in practice, besides its conceptual soundness. Next, we vary auxiliary losses applied to MaX-DeepLab, without tuning loss weights for remaining losses. Our PQ-style loss alone achieves a reasonable performance of 39.5% PQ. Adding instance discrimination significantly improves  $PQ^{\text{Th}}$ , showing the importance of a clustered feature embedding. Mask-ID prediction shares the same target with the Dice term in Equation (4.10), but helps focus on large masks when the Dice term is overwhelmed by small objects. Combining both of the auxiliary losses leads to a large 5.6% PQ gain. Further multi-tasking with semantic segmentation improves 0.6% PQ, because its class-level supervision helps stuff classes but not instance-level discrimination for thing classes.

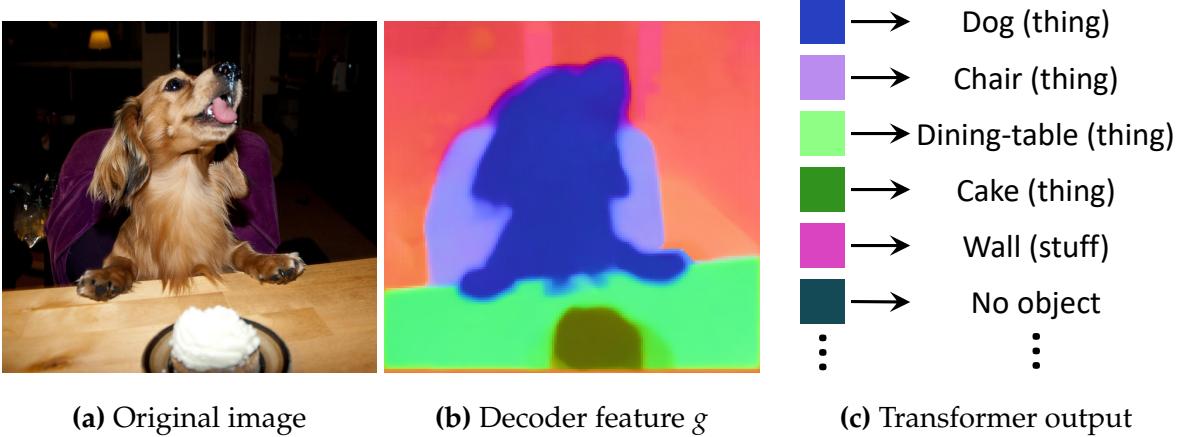
sim	InstDis	Mask	Sem	PQ	$PQ^{\text{Th}}$	$PQ^{\text{St}}$	SQ	RQ
$RQ \times SQ$	✓	✓	✓	45.7	49.8	39.4	80.9	55.3
$RQ + SQ$	✓	✓	✓	44.9	48.6	39.3	80.2	54.5
$RQ \times SQ$	✓	✓		45.1	50.1	37.6	80.6	54.5
$RQ \times SQ$		✓		43.3	46.4	38.6	80.1	52.6
$RQ \times SQ$	✓			42.6	48.1	34.1	80.0	52.0
$RQ \times SQ$				39.5	41.8	36.1	78.9	49.0

**Table 4.6.** Varying the similarity metric **sim** and whether to apply the auxiliary **Instance Discrimination** loss, **Mask-ID** cross-entropy loss or the **Semantic** segmentation loss.

#### 4.4.3 Analysis

We provide more insights of MaX-DeepLab by plotting our training curves and visualizing the mask output head.

**Training curves.** We first report the validation PQ curve in Figure 4.4a, with our default ablation model. MaX-DeepLab converges quickly to around 46% PQ within 100k iterations (54 epochs), 1/10 of DETR [142]. In Figure 4.4b and Figure 4.4c, we plot the characteristics of all matched masks in an image. The matched masks tend to have a better class correctness than mask correctness. Besides, we report per-pixel accuracies for instance discrimination (Figure 4.4d) and mask-ID prediction (Figure 4.4e). We see that most pixels learn quickly to find their own instances (out of  $K$ ) and predict their own mask-IDs (out of  $N$ ). Only 10% of all pixels predict wrong mask-IDs, but they contribute to most of the PQ error.



**Figure 4.5.** (b) Pixels of the same instance have similar colors (features), while pixels of different instances have distinct colors. (c) The transformer predicts mask colors (features) and classes.

**Visualization.** In order to intuitively understand the normalized decoder output  $g$ , the transformer mask feature  $f$ , and how they are multiplied to generate our mask output  $\hat{m}$ , we train a MaX-DeepLab with  $D = 3$  and directly visualize the normalized features as RGB colors. As shown in Figure 4.5, the decoder feature  $g$  assigns similar colors (or feature vectors) to pixels of the same mask, no matter the mask is a thing or stuff, while different masks are colored differently. Such effective instance discrimination (as colorization) facilitates our simple mask extraction with an inner product.

## 4.5 Conclusion

In this chapter, we have shown for the first time that panoptic segmentation can be trained end-to-end. Our MaX-DeepLab directly predicts masks and classes with a mask transformer, removing the needs for many hand-designed priors such as object bounding boxes, thing-stuff merging, *etc.* Equipped with a PQ-style loss and a dual-path transformer, MaX-DeepLab achieves the state-of-the-art result on the challenging COCO dataset, closing the gap between box-based and box-free methods for the first time.

## **Part II**

# **Self-Supervised Pre-Training of Long-Range Models**

# Chapter 5

## CO2: Consistent Contrast for Unsupervised Visual Representation Learning

This chapter studies the label consistency issue in self-supervised contrastive learning framework.

### 5.1 Introduction

Unsupervised visual representation learning has attracted increasing research interests for it unlocks the potential of large-scale pre-training for vision models without human annotation. Most of recent works learn representations through one or more pretext tasks, in which labels are automatically generated from image data itself. Several early methods propose pretext tasks that explore the inherent structures within a single image. For example, by identifying spatial arrangement [182], orientation [183], or chromatic channels [184], models learn useful representations for downstream tasks. Recently, another line of works [169], [172], [173], [185]–[188], e.g. Momentum Contrast (MoCo), falls within the framework of contrastive learning [189], which directly learns relations of images as the pretext task. In practice, contrastive learning methods show better

generalization in downstream tasks.

Although designed differently, most contrastive learning methods perform an instance discrimination task, *i.e.*, contrasting between image instances. Specifically, given a query crop from one image, a positive sample is an image crop from the same image; negative samples are crops randomly sampled from other images in the training set. Thus, the label for instance discrimination is a one-hot encoding over the positive and negative samples. This objective is to bring together crops from the same image and keep away crops from different images in the feature space, forming an instance discrimination task.

However, the one-hot label used by instance discrimination might be problematic, since it takes all the crops from other images as equally negative, which cannot reflect the heterogeneous similarities between the query crop and each of them. For example, some “negative” samples are semantically similar to the query, or even belong to the same semantic class as the query. This is referred to as “class collision” in [190] and “sampling bias” in [191]. The ignorance of the heterogeneous similarities between the query crop and the crops from other images can thus raise an obstacle for contrastive methods to learn a good representation. A recent work, supervised contrastive learning [174], fixes this problem by using human annotated class labels and achieves strong classification performance. However, in unsupervised representation learning, the human annotated class labels are unavailable, and thus it is more challenging to capture the similarities between crops.

In this chapter, we propose to view this instance discrimination task from the perspective of semi-supervised learning. The positive crop should be similar to the query for sure since they are from the same image, and thus can be viewed as labeled. On the contrary, the similarity between the query and each crop from other images is unknown, or unlabeled. With the viewpoint of semi-supervised learning, we introduce Consistent Contrast (CO2), a consistency regularization method which fits into current contrastive learning framework. Consistency regularization [192] is at the core of many state-of-the-art

semi-supervised learning algorithms [193]–[195]. It generates pseudo labels for unlabeled data by relying on the assumption that a good model should output similar predictions on perturbed versions of the same image. Similarly, in unsupervised contrastive learning, since the query crop and the positive crop naturally form two perturbed versions of the same image, we encourage them to have consistent similarities to each crop from other images. Specifically, the similarity of the positive sample predicted by the model is taken as a pseudo label for that of the query crop.

Our model is trained with both the original instance discrimination loss term and the introduced consistency regularization term. The instance discrimination label and the pseudo similarity label jointly construct a virtual soft label on-the-fly, and the soft label further guides the model itself in a bootstrap manner. In this way, CO2 exploits the consistency assumption on unlabeled data, mitigates the “class collision” effect introduced by the one-hot labels, and results in a better visual representation. More importantly, CO2 brings a new perspective of unsupervised visual representation learning. It relaxes the stereotype that the pretext task can only be *self-supervised* which aims to construct artificial labels for every sample, *e.g.*, a specific degree of rotation [183], a configuration of jigsaw puzzle [196], and a one-hot label that indicates whether a crop comes from the same instance or not [169]. In contrast, the pretext task can also be *self-semi-supervised*, allowing the task itself to be partially labeled. This relaxation is especially helpful when information for artificial label construction is not enough and imposing a label is harmful, such as the case of imposing the one-hot labels in instance discrimination.

This simple modification brings consistent gains on various evaluation protocols. We first benchmark CO2 on ImageNet [197] linear classification protocol. CO2 improves MoCo by 2.9% on top-1 accuracy. It also provides 3.8% and 1.1% top-5 accuracy gains under the semi-supervised setting on ImageNet with 1% and 10% labels respectively, showing the effectiveness of the introduced consistency regularization. We also evaluate the transfer ability of the learned representations on three different downstream tasks:

image classification, object detection and semantic segmentation. CO2 models consistently surpass their MoCo counterparts, showing that CO2 can improve the generalization ability of learned representation. Besides, our experiments on ImageNet-100 [187] demonstrate the efficacy of CO2 on SimCLR [172], showing the generality of our method on different contrastive learning frameworks.

## 5.2 Method

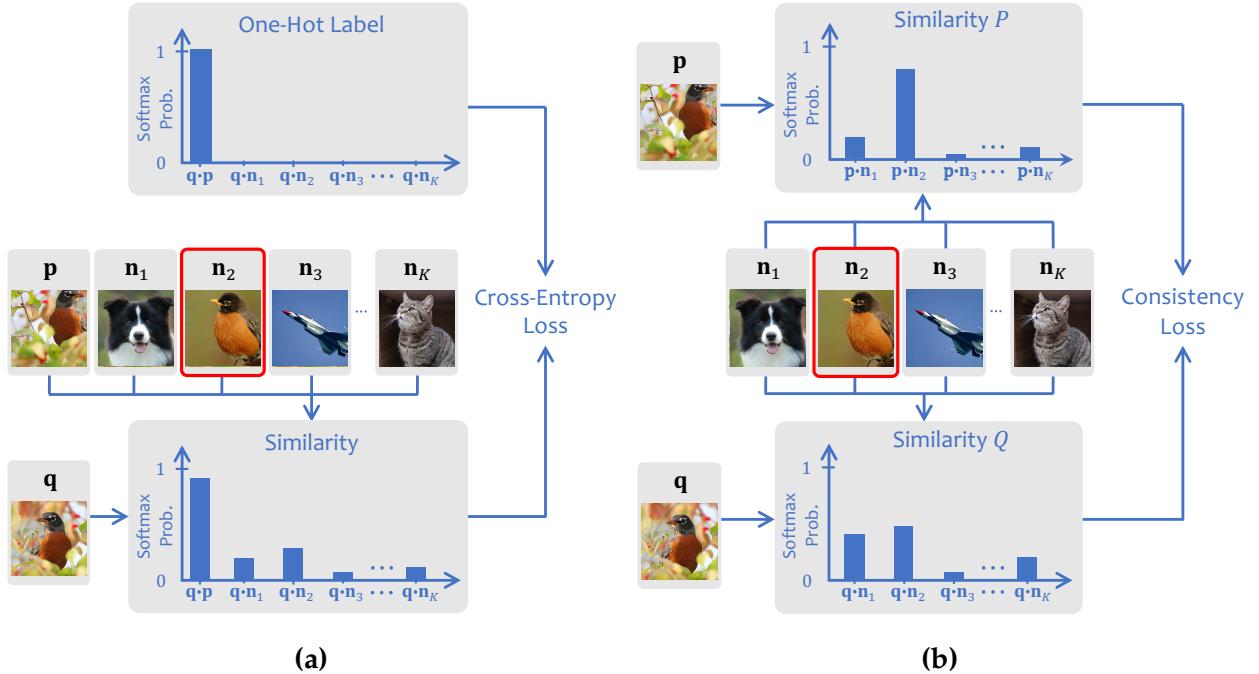
In this section, we begin by formulating current unsupervised contrastive learning as an instance discrimination task. Then, we propose our consistency regularization term which addresses the ignorance of the heterogeneous similarity between the query crop and each crop of other images in the instance discrimination task.

### 5.2.1 Contrastive Learning

Contrastive learning [189] is recently adopted as an objective for unsupervised learning of visual representations. Its goal is to find a parametric function  $f_\theta : \mathbb{R}^D \rightarrow \mathbb{R}^d$  that maps an input vector  $\mathbf{x}$  to a feature vector  $f_\theta(\mathbf{x}) \in \mathbb{R}^d$  with  $D \gg d$ , such that a simple distance measure (e.g., cosine distance) in the low-dimensional feature space can reflect complex similarities in the high-dimensional input space.

For each input vector  $\mathbf{x}_i$  in the training set  $\mathbb{S}$ , the similarity measure in the input space is defined by a subset of training vectors  $\mathbb{S}_i \subset \mathbb{S}$ , called similarity set. The sample  $\mathbf{x}_i$  is deemed similar to samples in the similarity set  $\mathbb{S}_i$ , but dissimilar to samples in  $\mathbb{S} \setminus \mathbb{S}_i$ . Then, the contrastive objective encourages  $f_\theta(\mathbf{x}_j)$  to be close to  $f_\theta(\mathbf{x}_i)$  in the feature space if  $\mathbf{x}_j \in \mathbb{S}_i$ , and otherwise to be distant.

By training with contrastive loss, the similarities defined by the similarity set determine characteristics of the learned representation and the mapping function  $f_\theta$ . For example, if the similarity is defined as samples from the same semantic class, then  $f_\theta$  will probably



**Figure 5.1.** Illustration of (a) instance discrimination and (b) our consistency regularization.  $q$  is a query and  $p$  is a positive key. Both are encoded from crops of the same image.  $\{n_k\}_{k=1}^K$  are negative keys, encoded from random crops. In (a), the similarity is softmax cosine distances between  $q$  and all keys. This similarity is optimized towards an artificial one-hot label which identifies  $p$  among all keys. However, some negatives can be semantically similar but not reflected by the one-hot label (e.g., the one rounded by a red box). In (b), our proposed consistency regularization encourages the agreement between  $P$ , the positive-negative similarity, and  $Q$ , the query-negative similarity, reflecting the heterogeneous similarity of the query/positive to the negatives.

learn invariances to other factors, e.g., object deformation. In the supervised setting, this definition of similarity requires a large amount of human labeling. On the contrary, unsupervised contrastive learning exploits similarities with no need of human labels. One natural definition of unsupervised similarity is multiple views of an image, as explored by many recent methods. For example, random augmented crops [169], [172], [173], [198], [199] of an image could be defined as a similarity set. In this case, the contrastive objective is effectively solving an instance discrimination task [169] as illustrated in Figure 5.1a.

The training of this instance discriminator involves randomly sampling a query crop  $\mathbf{x}^q \in \mathcal{S}_i$ , a positive crop  $\mathbf{x}^p \in \mathcal{S}_i$  from the same image, and  $K$  negative crops  $\{\mathbf{x}^k \in \mathcal{S} \setminus \mathcal{S}_i\}_{k=1}^K$  from other images. These  $K + 2$  crops (the query, the positive, and  $K$  negatives) are encoded with  $f_\theta$  respectively,  $\mathbf{q} = f_\theta(\mathbf{x}^q)$ ,  $\mathbf{p} = f_\theta(\mathbf{x}^p)$ ,  $\mathbf{n}_k = f_\theta(\mathbf{x}^k)$ . Then, an effective contrastive loss function, InfoNCE [186], is written as:

$$\mathcal{L}_{ins} = -\log \frac{\exp(\mathbf{q} \cdot \mathbf{p} / \tau_{ins})}{\exp(\mathbf{q} \cdot \mathbf{p} / \tau_{ins}) + \sum_{k=1}^K \exp(\mathbf{q} \cdot \mathbf{n}_k / \tau_{ins})}, \quad (5.1)$$

where  $\tau_{ins}$  is a temperature hyper-parameter [200]. This loss can be interpreted as a cross entropy loss that trains the model to discriminate the positive crop (labeled as 1) from negative crops (labeled as 0) given the query crop. We denote this loss as  $\mathcal{L}_{ins}$  as it performs an instance discrimination task. One direct instantiation of InfoNCE loss, represented by SimCLR [172], formulates  $f_\theta$  as an end-to-end encoder. In this case, two crops of the same image are exchangeable or symmetric to each other as both are encoded by  $f_\theta$ . The final loss is also symmetric with either one of the two crops as the query and the other crop as the positive. Another popular instantiation, represented by MoCo [173], encodes the query with  $f_\theta$  and encodes the positive and the negatives with  $f_{\theta'}$  which is the moving average of  $f_\theta$ . In this case, only  $\mathbf{q}$  can propagate gradients, which causes  $\mathcal{L}_{ins}$  to be asymmetric.

## 5.2.2 Consistent Contrast

The one-hot labels used by InfoNCE loss is effective, showing good generalization ability across tasks and datasets [172], [199]. Nevertheless, we argue that the hard, zero-one labels is uninformative. Specifically, crops from other images are taken as equally negative as they are all labeled as 0. This is contradictory to the fact that some so-called “negative” crops can be similar or even in the same semantic class, especially when  $K$  is large. For example, SimCLR [172] uses 16,382 negative samples in a batch, and MoCo [173], [199] uses a memory bank of 65,536 features as negative samples. Even worse, the current objective forces negatives to be as far from the query as possible, with larger weights for closer negatives since they are “hard negatives”. However, these “hard negative” crops in fact tend to be semantically close. These issues impair good representation learning because the one-hot labels can not faithfully reflect the heterogeneous similarities between the query crop and the crops from other images.

Although generating labels based on instance discrimination is trivial, revealing the similarity between two arbitrary crops is exactly what we want to learn from unsupervised pre-training. Therefore, the label of the similarity between the query crop and each crop from other images is of little hope to get. This situation is similar to the usage of unlabeled data in semi-supervised learning setting, in which consistency regularization is widely used to propagate knowledge from labeled data to discover the structures in unlabeled data. Inspired by this, we propose to encourage the consistency between the similarities of crops from the same image, *i.e.*, the query crop and the positive crop. We illustrate the consistency regularization in Figure 5.1b.

First, we denote the similarity between the query  $\mathbf{q}$  and the negatives  $\mathbf{n}_i (i \in \{1, \dots, K\})$  as:

$$Q(i) = \frac{\exp(\mathbf{q} \cdot \mathbf{n}_i / \tau_{con})}{\sum_{k=1}^K \exp(\mathbf{q} \cdot \mathbf{n}_k / \tau_{con})}, \quad (5.2)$$

where  $\tau_{con}$  is also a temperature hyper-parameter.  $Q(i)$  is the probability that the query  $\mathbf{q}$

selects  $\mathbf{n}_i$  as its match from  $\{\mathbf{n}_k\}_{k=1}^K$ . Similarly, the similarity between the positive  $\mathbf{p}$  and the negatives is written as:

$$P(i) = \frac{\exp(\mathbf{p} \cdot \mathbf{n}_i / \tau_{con})}{\sum_{k=1}^K \exp(\mathbf{p} \cdot \mathbf{n}_k / \tau_{con})}. \quad (5.3)$$

We impose the consistency between the probability distributions  $P$  and  $Q$  by using symmetric Kullback-Leibler (KL) Divergence as the measure of disagreement:

$$\mathcal{L}_{con} = \frac{1}{2} D_{KL}(P \| Q) + \frac{1}{2} D_{KL}(Q \| P). \quad (5.4)$$

When  $\mathbf{p}$  and  $\mathbf{q}$  are encoded by the same end-to-end encoder  $f_\theta$ , it is natural to use symmetric KL as their disagreement measure, since  $\mathbf{p}$  and  $\mathbf{q}$  are exchangeable. Even when  $\mathbf{p}$  and  $\mathbf{n}_i$  are encoded by the momentum encoder  $f'_\theta$ , symmetric KL empirically works as well as forward KL, *i.e.*,  $D_{KL}(P \| Q)$ , as shown in Section 5.3.5. Thus, we use symmetric KL as a unified objective for both cases.

The total loss is a weighted average of the original instance discrimination loss term and the consistency regularization term:

$$\mathcal{L} = \mathcal{L}_{ins} + \alpha \mathcal{L}_{con}, \quad (5.5)$$

where  $\alpha$  denotes the coefficient to balance the two terms. It is possible to merge the two terms by creating a unique label containing information of both the one-hot label and the pseudo similarity label, but we find the weighted average can already get good performance and is easy to control.

The pseudo label is informative to reveal the similarity between the query  $\mathbf{q}$  and each  $\mathbf{n}_i$ , while the one-hot label is unable to provide such information, since it only describe co-occurrence within one image. Note that, the pseudo label is also dynamic since the embedding function  $f_\theta$  is updated in every training step, and thus generating better pseudo labels during training. It indicates that the unsupervised embedding function and the soft similarity labels give positive feedback to each other.

Our method is simple and low-cost. It captures the similarity to each  $\mathbf{n}_i$  while introduc-

ing unnoticeable computational overhead with only one extra loss term computed. This is unlike clustering based unsupervised learning methods, which are costly, since they explicitly compute the similarity sets in the training set after every training epoch [201]–[204].

## 5.3 Experiments

Herein, we first report our implementation details and benchmark the learned representations on ImageNet. Next, we examine how the unsupervised pre-trained models transfer to other datasets and tasks. We then analyze the characteristics of our proposed method.

### 5.3.1 Linear Classification

**Table 5.1.** Linear classification protocol on ImageNet-1K

Pretext Task	Arch.	Head	#epochs	Top-1 Acc. (%)
ImageNet Classification	R50	-	90	76.5
Exemplar [205]	R50w3×	-	35	46.0
Relative Position [182]	R50w2×	-	35	51.4
Rotation [183]	Rv50w4×	-	35	55.4
Jigsaw [196]	R50	-	90	45.7
<i>Methods based on contrastive learning:</i>				
InsDisc [169]	R50	Linear	200	54.0
Local Agg. [202]	R50	Linear	200	58.2
CPC v2 [186]	R170 <sub>w</sub>	-	~200	65.9
CMC [187]	R50	Liner	240	60.0
AMDIM [185]	AMDIM <sub>large</sub>	-	150	68.1
PIRL [188]	R50	Linear	800	63.6
SimCLR [172]	R50	MLP	1000	69.3
MoCo [173]	R50	Linear	200	60.6
MoCo [173] + CO2	R50	Linear	200	63.5
MoCo v2 [199]	R50	MLP	200	67.5
MoCo v2 [199] + CO2	R50	MLP	200	68.0

**Table 5.2.** Top-5 accuracy for semi-supervised learning on ImageNet

Pretext Task	1% labels	10% labels
Supervised Baseline	48.4	80.4
InsDisc [169]	39.2	77.4
PIRL [188]	57.2	83.8
MoCo [173]	62.4	84.1
MoCo [173] + CO2	66.2	85.2
MoCo v2 [199]	69.5	85.1
MoCo v2 [199] + CO2	70.6	85.4

**Setup.** We mainly evaluate CO2 based on MoCo [173] and MoCo v2 [199]. Both of them use instance discrimination as pretext task, while MoCo v2 adopts more sophisticated design choices on projection head architecture, learning rate schedule and data augmentation strategy. We test CO2 on MoCo for its representativeness and simplicity. On MoCo v2, we evaluate how CO2 is compatible with advanced design choices. We also demonstrate the impact of CO2 on the end-to-end contrastive framework in Section 5.3.5.

The unsupervised training is performed on the train split of ImageNet-1K [197] without using label information. We keep aligned every detail with our baseline MoCo to effectively pin-point the contribution of our approach, except the number of GPUs (MoCo uses 8 GPUs while we use 4). A further search on MoCo-related hyper-parameters might lead to better results of our method. For the hyper-parameters of CO2, we set  $\tau_{con}$  as 0.04,  $\alpha$  as 10 for MoCo-based CO2, and  $\tau_{con}$  as 0.05,  $\alpha$  as 0.3 for MoCo v2-based CO2. Please refer to the appendix for more detailed implementation description.

### 5.3.2 Linear Classification

We first benchmark the learned representations on the common linear classification protocol. After the unsupervised pre-training stage, we freeze the backbone network including the batch normalization parameters, and train a linear classifier consisting of a fully-connected layer and a softmax layer on the 2048-D features following the global average

pooling layer. Table 5.1 summarizes the single-crop top-1 classification accuracy on the validation set of ImageNet-1K. Our method consistently improves by 2.9% on MoCo and by 0.5% on MoCo v2. We also list several top-performing methods in the table for reference. These results indicate that the representation is more linearly separable on ImageNet with consistency regularization, since the consistency regularization mitigates the “class collision” effect caused by semantically similar negative samples.

### 5.3.3 Semi-Supervised Learning

We next perform semi-supervised learning on ImageNet to evaluate the effectiveness of the pre-trained network in data-efficient settings. Following [169], [172], [188], we finetune the whole pre-trained networks with only 1% and 10% labels which are sampled in a class-balanced way. Table 5.2 summarizes the mean of the top-5 accuracy on the validation set of ImageNet-1K over three runs. The results for MoCo and MoCo v2 are produced by us using their officially released models. The proposed consistency regularization term can provide 3.8% and 1.1% top-5 accuracy gains for MoCo with 1% and 10% labels respectively. CO2 also improves from MoCo v2 by 1.1% top-5 accuracy with 1% labels, and by 0.3% with 10% labels.

### 5.3.4 Transfer Learning

To further investigate the generalization ability of our models across different datasets and tasks, we evaluate the transfer learning performance on PASCAL VOC [208] with three typical visual recognition tasks, *i.e.*, image classification, object detection and semantic segmentation. Table 5.3 reports the transfer learning performance comparing with other methods using ResNet-50. CO2 shows competitive or better performance comparing with the corresponding baselines, In addition, it achieves better performance than state-of-the-art unsupervised representation learning methods.

**Table 5.3.** Transfer learning performance on PASCAL VOC datasets

Pretext Task	mAP	Image Classification		Object Detection		Semantic Segmentation
		AP <sub>50</sub>	AP <sub>all</sub>	AP <sub>75</sub>	mIoU	
ImageNet Classification	88.0	81.3	53.5	58.8	74.4	
Rotation [183]	63.9	72.5	46.3	49.3	-	
Jigsaw [196]	64.5	75.1	48.9	52.9	-	
InsDisc [169]	76.6	79.1	52.3	56.9	-	
PIRL [188]	81.1	80.7	54.0	59.7	-	
SimCLR [172]*	-	81.8	55.5	61.4	-	
BYOL [206]*	-	81.4	55.3	61.1	-	
SwAV [204]*	-	81.5	55.4	61.4	-	
SimSiam [207]*	-	82.4	57.0	63.7	-	
MoCo [173]	-	81.5	55.9	62.6	72.5	
MoCo [173] ( <i>our impl.</i> )	79.7	81.6	56.2	62.4	72.6	
MoCo [173] + CO2	82.6	81.9	56.0	62.6	73.3	
MoCo v2 [199]	85.0	82.4	57.0	63.6	74.2	
MoCo v2 [199] + CO2	85.2	82.7	57.2	64.1	74.7	

\* Results reported in SimSiam [207]

**Image Classification.** Following the evaluation setup in [209], we train a linear SVM [210] on the frozen 2048-D features extracted after the global average pooling layer. The results of MoCo are produced by us with their official models. In this case, CO2 is 2.9% better than MoCo, and 0.2% than MoCo v2.

**Object Detection.** Following the detection benchmark set up in [173], we use Faster R-CNN [139] object detector and ResNet-50 C4 [211] backbone, and all the layers are finetuned including the batch normalization parameters. The numbers of our method are averaged over three runs. Our reproduced results for MoCo are also listed in the table for reference. CO2 provides 0.3% AP<sub>50</sub> gains on both MoCo and MoCo v2.

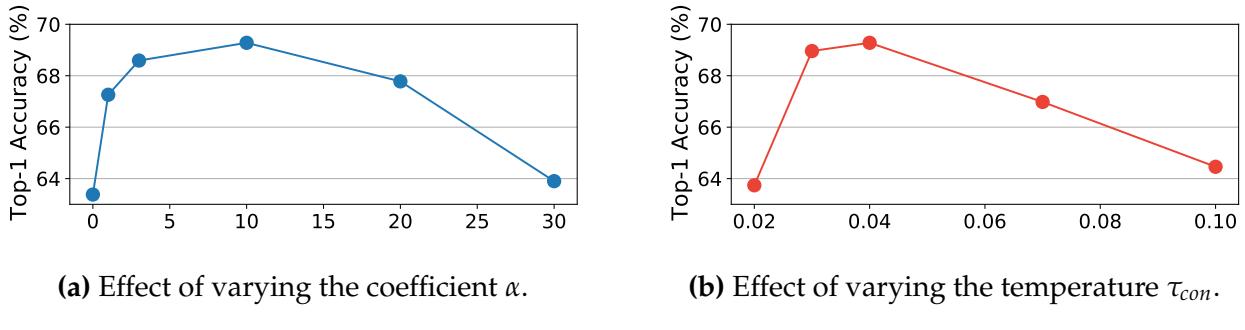
**Semantic Segmentation.** We follow the settings in [173] for semantic segmentation. Results are average over three runs. Similarly, we include our reproduced results of MoCo as a reference. The result of MoCo v2 is produced by us using its officially released model.

CO2 gives 0.9% mIoU improvement upon MoCo, and 0.5% upon MoCo v2, which finally surpasses its supervised counterpart.

The overall transfer learning improvements, though consistent, are smaller than linear classification and semi-supervised learning. Similar observations have also been made in [199]. We hypothesize that the current unsupervised contrastive methods, which bring close different crops from the same image, reduce the representation’s sensitivity to location which is useful for tasks like detection. It is still an open question which properties of an unsupervised representation benefit the transfer ability to various downstream tasks.

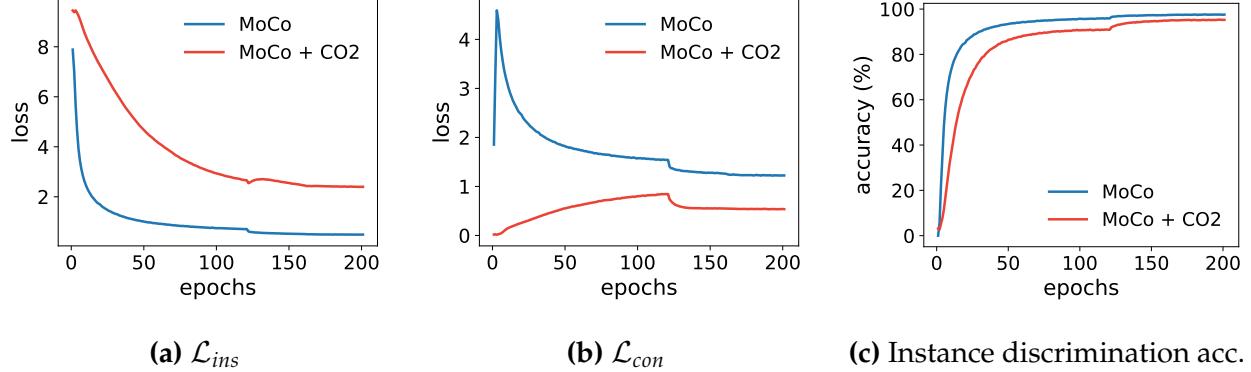
### 5.3.5 Analysis

In this section, we study the characteristics of the proposed method on a smaller backbone ResNet-18 and a smaller dataset ImageNet-100 due to the consideration of the computational resource. ImageNet-100 is firstly used in [187] and consists of 100 randomly selected classes from all 1,000 classes of ImageNet.



**Figure 5.2.** Ablation on the effect of hyper-parameters.

**Hyper-parameter.** Our method introduces two new hyper-parameters, the coefficient of consistency regularization term  $\alpha$ , and its temperature  $\tau_{con}$ . In Figure 5.2, we show the top-1 accuracy of a linear classifier on models pre-trained by CO2 with different hyper-parameters. In Figure 5.2a, we fix the temperature  $\tau_{con}$  as 0.04 and vary the coefficient  $\alpha$ . The best coefficient is 10. We see that by using the consistency regularization term,



**Figure 5.3.** Training curves of ResNet-18 on ImageNet-100.

the linear classification accuracy can be boosted from 63.6% to 69.2%. Increasing  $\alpha$  to 20 and beyond causes performance degeneration. We hypothesize that the model is over-regularized by the consistency loss, and thus it loses some discrimination among different instances. In Figure 5.2b, we fix the coefficient to be 10 and varying the temperature. As other consistency regularization methods (e.g., [194]), temperature  $\tau_{con}$  effectively influences the quality of the learned representation, and the best to use is 0.04.

**Training Curves.** In Figure 5.3 we show the training curves of the instance discrimination loss  $\mathcal{L}_{ins}$ , the consistency loss  $\mathcal{L}_{con}$  and the instance discrimination accuracy. Instance discrimination accuracy represents the percent of query crops which successfully select their corresponding positive crops, *i.e.*, successfully identify their instances. MoCo is trained with  $\mathcal{L}_{ins}$  only and its  $\mathcal{L}_{con}$  is just calculated out for comparison. We see that  $\mathcal{L}_{ins}$  of MoCo drops quickly from the beginning at the cost of a jump of  $\mathcal{L}_{con}$ . As the training proceeds,  $\mathcal{L}_{con}$  of MoCo decreases spontaneously, possibly because more semantic knowledge has been learned, but it is still relatively high. Training with  $\mathcal{L}_{con}$  and  $\mathcal{L}_{ins}$  together, *i.e.*, MoCo + CO2,  $\mathcal{L}_{con}$  is kept very low from beginning, and  $\mathcal{L}_{con}$  increases gradually since the model is trained to discriminate between images at the same time. At the end of the training,  $\mathcal{L}_{con}$  stays much lower than  $\mathcal{L}_{con}$  of MoCo.

We also notice that with CO2, the instance discrimination accuracy drops from 97.57%

to 95.26%. Although CO2 results in lower instance discrimination accuracy, it still does better in the downstream classification task. The linear classification accuracy improves from 63.6% to 69.2%, as shown in Figure 5.2a. It suggests again that there is a gap between instance discrimination and the downstream tasks.

**Comparison with Label Smoothing.** With the consistency regularization term, our approach assigns soft pseudo labels to crops from other images. This looks similar to label smoothing regularization on supervised classification [212], a useful trick which assigns a small constant value to the labels of all the negative classes to avoid overconfidence. We equip MoCo with label smoothing, *i.e.*, assigning a small constant value to crops from other images (the “negatives”). Surprisingly, it reports 61.2% linear classification accuracy, 2.4% lower than MoCo alone. This suggests that assigning a constant value as label smoothing can be harmful for unsupervised contrastive learning, since it ignores the heterogeneous similarity relationship. And it is better to assign labels according to the similarities as our consistency regularization.

**Table 5.4.** Linear classification accuracy (%) using an end-to-end encoder and with different choices of  $\mathcal{L}_{con}$ . The results are summarized as mean and standard deviation over three different runs.

Method	No $\mathcal{L}_{con}$	Forward KL	Reverse KL	Symmetric KL (CO2)
SimCLR	$68.9 \pm 0.06$	-	-	$72.3 \pm 0.14$
MoCo	$63.1 \pm 0.29$	$69.6 \pm 0.27$	$65.1 \pm 0.52$	$69.7 \pm 0.41$

**End-to-End Encoder.** To further verify the effectiveness of the proposed consistency regularization term on different contrastive learning frameworks, we apply CO2 to SimCLR [172], a representative method with an end-to-end encoder (without a momentum encoder). The results are presented in Table 5.4. On ImageNet-100 [187] with a ResNet-18, SimCLR obtains 68.9% top-1 linear classification accuracy with batch size 128 and temperature  $\tau_{ins}$  0.1. Equipped with CO2 whose coefficient  $\alpha$  is 0.07 and temperature  $\tau_{con}$  is 1.0,

the linear classification accuracy is boosted to 72.3%. The improvement demonstrates that CO2 can be applied to different unsupervised contrastive frameworks and improve the quality of the learned representation regardless of whether using a momentum encoder or not.

**Varying the choices of  $\mathcal{L}_{con}$ .** We ablate on different variants of  $\mathcal{L}_{con}$  (Eq. 5.4) on MoCo including forward KL ( $D_{KL}(P\|Q)$ ), reverse KL ( $D_{KL}(Q\|P)$ ), and the objective of CO2, *i.e.*, symmetric KL. Each of models uses a coefficient  $\alpha$  of 10 and a temperature  $\tau_{con}$  of 0.04. We present the linear classification accuracy in Table 5.4. Our CO2 (symmetric KL) improves over the baseline MoCo by a large margin, from 63.1% to 69.7%. Forward KL alone improves similarly to 69.6%. And reserve KL alone can also provide a nontrivial 2.0% gain in accuracy.

## 5.4 Related Work

Our method falls in the area of unsupervised visual representation learning, especially for image data. In this section, we first revisit various design strategies of pretext tasks for unsupervised learning. Then we elaborate on the pretext tasks based on contrastive learning, which is the focus of this chapter. Next, we review the methods using consistency regularization in semi-supervised learning, which inspire this chapter.

**Unsupervised Learning and Pretext Tasks** To learn from unlabeled image data, a wide range of pretext tasks have been established. The models can be taught to specify the relative position of a patch [182], solve spatial jigsaw puzzles [196], [213], colorize gray scale images [184], [214], inpaint images [215], count objects [216], discriminate orientation [183], iteratively cluster [201], [202], [217], generate images [218], [219], *etc.* [220] evaluates the combination of different pretext tasks. [221] and [209] revisit and benchmark different pretext tasks.

**Contrastive Learning** Contrastive learning [189] recently puts a new perspective on the design of pretext task and holds the key to most state-of-the-art methods. Most of them perform an instance discrimination task while differ in i) the strategies to synthesize positives and negatives, and ii) the mechanisms to manage a large amount of negatives. The synthesizing can base on context with patches [186], [222], random resized crops with data augmentation [169], [172], [173], [185], [198], jigsaw puzzle transformation [188] or luminance-chrominance decomposition [187]. Regarding the mechanisms to maintain negative features, some methods [169], [188] keep tracking the features of all images, some directly utilize the samples within the minibatch [172], [187], [198], and [173] proposes to use a momentum encoder. [206] recently proposes to only use positive examples without negatives. Recently, [203] argues that the lack of semantic structure is one fundamental weakness of instance discrimination, and proposes to generate prototypes by k-means clustering. However, the computational overhead and the degeneration introduced by clustering are difficult to address. [191] also points out the possible sampling bias of instance discrimination, and proposes a debiased objective.

**Consistency Regularization** Consistency regularization is an important component of many successful semi-supervised learning methods. It is firstly proposed in [192], encouraging similar predictions on perturbed versions of the same image. Besides the consistency regularization on unlabeled data, the model is simultaneously trained with a supervised loss on a small set of labeled data. Several works made improvements on the way of perturbation, including using an adversarial transformation [223], using the prediction of a moving average or previous model [224], [225], and using strong data augmentation [193]. Recently, several larger pipelines are proposed [194], [195], [226], in which consistency regularization still serves as a core component.

The instance discrimination loss in unsupervised contrastive learning is analogous to the supervised loss in semi-supervised learning, as both of them rely on some concrete information, *i.e.*, co-occurrence in one image and human annotation, respectively. Mean-

while, CO2 on the similarities between crops is analogous to consistency regularization on unlabeled samples of semi-supervised methods as their labels are both unknown. The main difference, however, is that semi-supervised methods crucially rely on the supervised loss to warm up the model, while there is no human annotation at all in unsupervised contrastive learning. This chapter presents an example that a model learned completely without human annotations can also generate surprisingly effective pseudo labels for similarities between different crops and benefit from consistency regularization.

## 5.5 Discussion

Unsupervised visual representation learning has shown encouraging progress recently, thanks to the introduction of instance discrimination and the contrastive learning framework. However, in this chapter, we point out that instance discrimination is ignorant of the heterogeneous similarities between image crops. We address this issue with a consistency regularization term on the similarities between crops, inspired by semi-supervised learning methods which impose consistency regularization on unlabeled data. In such a simple way, the proposed CO2 consistently improves on supervised and semi-supervised image classification. It also transfers to other datasets and downstream tasks.

More broadly, we encourage researchers to rethink label correctness in existing pretext tasks. Taking instance discrimination as an example, we show that a pretext task itself could be, in fact, a semi-supervised learning task. It might be harmful to think of the pretext task as a simple pure supervised task by assuming the unknown labels are negatives. In addition, this chapter relaxes the stereotype restriction that pretext task labels should always be known and clean. We hope this relaxation can give rise to novel pretext tasks which exploit noisy labels or partially-available labels, making a better usage of the data without human annotation.

# Chapter 6

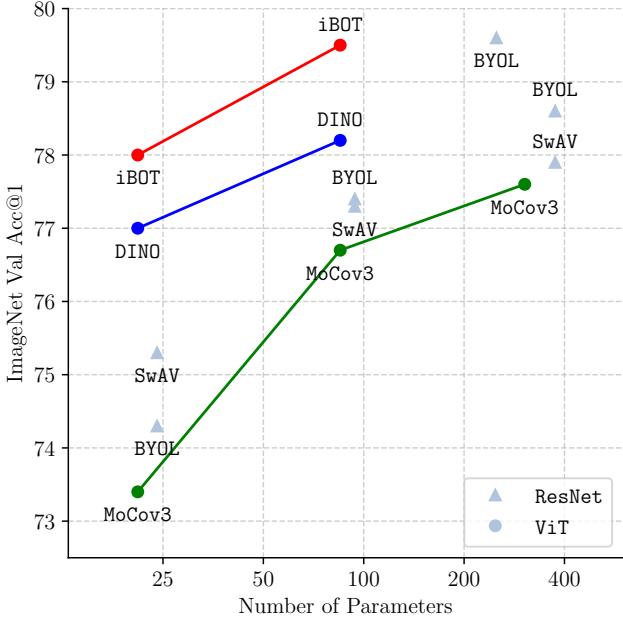
## iBOT: Image BERT Pre-Training with Online Tokenizer

This chapter describes a method that learns semantic representations by performing masked image modeling with transformers.

### 6.1 Introduction

Masked Language Modeling (MLM), which first randomly masks and then reconstructs a set of input tokens, is a popular pre-training paradigm for language models. The MLM pre-trained Transformers [6] have demonstrated their scalability to large-capacity models and datasets, becoming a de-facto standard for lingual tasks. However, its potential for Vision Transformer (ViT), which recently started to revolutionize computer vision research [227], [228], has been largely underexplored. Most popular unsupervised pre-training schemes in vision deal with the global views [229], [230], neglecting images' internal structures, as opposed to MLM modeling local tokens. In this chapter, we seek to continue the success of MLM and explore Masked Image Modeling (MIM) for training better Vision Transformers such that it can serve as a standard component, as it does for NLP.

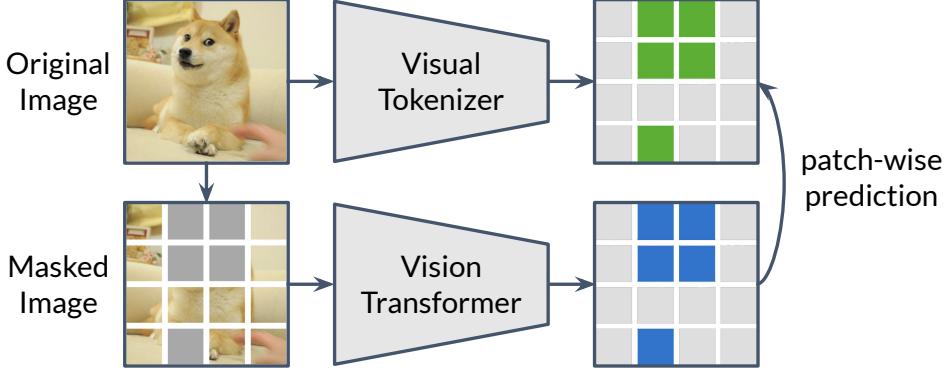
One of the most crucial components in MLM is the *lingual tokenizer* which splits language into semantically meaningful tokens, *e.g.*, WordPiece [62] in BERT. Similarly, the crux



**Figure 6.1. Linear probing accuracy on ImageNet.** We compare iBOT with other unsupervised baselines.

of MIM lies in a proper design of *visual tokenizer*, which transforms the masked patches to supervisory signals for the target model, as shown in Figure 6.2. However, unlike lingual semantics arising naturally from the statistical analysis of word frequency [231], visual semantics cannot be extracted such easily due to the continuous property of images. Empirically, visual semantics emerges progressively by bootstrapping online representation that enforces a similarity of distorted image views [173], [204], [206]. This property intuitively indicates a multi-stage training pipeline, where we need to first train an off-the-shelf semantic-rich tokenizer before training the target model. However, since acquiring visual semantics is a common end for both the tokenizer and target model, a single-stage training pipeline where the tokenizer and target model can be jointly optimized awaits further exploration.

Previous works partially tackle the above challenges. Several works use identity mapping as the visual tokenizer, *i.e.*, predicting the raw pixel values [215], [232]. Such



**Figure 6.2. Masked image modeling.** Given a masked image, the vision transformer learns to predict the output of a visual tokenizer.

paradigm struggles in semantic abstraction and wastes the capacity at modeling high-frequency details, yielding less competitive performance in semantic understanding [233]. Recently, BEiT [234] proposes to use a pre-trained discrete VAE [235] as the tokenizer. Though providing some level of abstraction, the discrete VAE is still found only to capture low-level semantics within local details (as observed by Table 6.9). Moreover, the tokenizer needs to be offline pre-trained with fixed model architectures and extra dataset [235], which potentially limits its adaptability to perform MIM using data from different domains.

To this end, we present iBOT, short for image BERT pre-training with Online Tokenizer, a new framework that performs MIM with a tokenizer handling above-mentioned challenges favorably. We motivate iBOT by formulating the MIM as knowledge distillation (KD), which learns to distill knowledge from the tokenizer, and further propose to perform self-distillation for MIM with the help of twin teacher as online tokenizer. The target network is fed with a masked image while the online tokenizer with the original image. The goal is to let the target network recover each masked patch token to its corresponding tokenizer output. Our online tokenizer naturally resolves two major challenges. On the one hand, our tokenizer captures high-level visual semantics progressively learned by enforcing the similarity of cross-view images on class tokens. On the other hand, our tokenizer needs no extra stages of training as pre-processing setup since it is jointly optimized

with MIM via momentum update.

The online tokenizer enables iBOT to achieve excellent performance for feature representation. Specifically, iBOT advances ImageNet-1K classification benchmark under  $k$ -NN, linear probing and fine-tuning protocols to 77.1%, 79.5%, 84.0% with ViT-Base/16 respectively, which is 1.0%, 1.3%, 0.4% higher than previous best results. When pre-trained with ImageNet-22K, iBOT with ViT-L/16 achieves a linear probing accuracy of 81.7% and a fine-tuning accuracy of 86.6%, which is 0.4% and 0.6% higher than previous best results. Beyond that, the advancement is also valid when transferring to other datasets or under semi-supervised and unsupervised classification settings. Of particular interest, we have identified an emerging part-level semantics that can help the model with image recognition both on global and local scales. We identify that the semantic patterns learned in patch tokens, which sufficiently lack in the off-line tokenizer as in BEiT [234], helps the model to be advanced in linear classification and robustness against common image corruptions. When it is transferred to downstream tasks, we show that in downstream tasks related to image classification, object detection, instance segmentation, and semantic segmentation, iBOT surpasses previous methods with nontrivial margins. All of the evidence demonstrates that iBOT has largely closed the gap of masked modeling pre-training between language and vision Transformers.

## 6.2 Preliminaries

### 6.2.1 Masked Image Modeling as Knowledge Distillation

Masked image modeling (MIM), which takes a similar formulation as MLM in BERT, has been proposed in several recent works [234], [236]. Specifically, for an image token sequence  $\mathbf{x} = \{x_i\}_{i=1}^N$ , MIM first samples a random mask  $\mathbf{m} \in \{0, 1\}^N$  according to a prediction ratio  $r$ , where  $N$  is the number of tokens. The patch token  $x_i$  where  $m_i$  being 1, denoted as  $\tilde{\mathbf{x}} \triangleq \{x_i \mid m_i = 1\}_{i=1}^N$ , are then replaced with a mask token  $e_{[\text{MASK}]}$ ,

yielding a corrupted image  $\hat{\mathbf{x}} \triangleq \{\hat{\mathbf{x}}_i = (1 - m_i)\mathbf{x}_i + m_i \mathbf{e}_{[\text{MASK}]} \}_{i=1}^N$ . MIM is to recover the masked tokens  $\tilde{\mathbf{x}}$  from the corrupted image  $\hat{\mathbf{x}}$ , *i.e.*, to maximize:  $\log q(\tilde{\mathbf{x}}|\hat{\mathbf{x}}) \approx \sum_{i=1}^N m_i \cdot \log q(\mathbf{x}_i|\hat{\mathbf{x}})$ , where  $\approx$  holds with an independence assumption that each masked token can be reconstructed separately. In BEiT [234],  $q$  is modelled as a categorical distribution over  $K$  categories and the task is to

$$\min_{\theta} \sum_{i=1}^N m_i H(P(c_i|\mathbf{x}; \phi), P(c_i|\hat{\mathbf{x}}; \theta)), \quad (6.1)$$

where a random variable  $c_i \in \{1, 2, \dots, K\}$  denotes the category id of the  $i$ -th patch token,  $P(\cdot)$  is a probability distribution of  $c_i$  over the  $K$  categories.  $H$  computes the cross entropy between the two distributions  $H(P_1(c_i), P_2(c_i)) = -\sum_{k=1}^K P_1(c_i=k) \log P_2(c_i=k)$ . And  $\phi$  is the parameter of a discrete VAE [235] that clusters image patches into  $K$  categories and assigns each patch token a one-hot encoding identifying its category. We note this loss is formulated similarly to knowledge distillation [237], where knowledge is distilled from a pre-fixed tokenizer parameterized by  $\phi$  to current model parameterized by  $\theta$ .

### 6.2.2 Self-Distillation

Self-distillation, proposed recently in DINO [230], distills knowledge not from posterior distributions  $P(c|\mathbf{x}; \phi)$  but past iterations of model itself  $P(c|\mathbf{x}; \theta')$  and is cast as a *discriminative* self-supervised objective. Given the training set  $\mathcal{I}$ , an image  $\mathbf{x} \sim \mathcal{I}$  is sampled uniformly, over which two random augmentations are applied, yielding two distorted views  $\mathbf{u}$  and  $\mathbf{v}$ . The two distorted views are then put through a teacher-student framework to get the predictive categorical distributions from the [CLS] token:  $v_t^{[\text{CLS}]} = P(c_{[\text{CLS}]}|\mathbf{v}; \theta')$  and  $u_s^{[\text{CLS}]} = P(c_{[\text{CLS}]}|\mathbf{u}; \theta)$ . The knowledge is distilled from teacher to student by minimizing their cross-entropy, formulated as

$$\mathcal{L}_{[\text{CLS}]} = H(P(c^{[\text{CLS}]}|\mathbf{v}; \theta'), P(c^{[\text{CLS}]}|\mathbf{u}; \theta)). \quad (6.2)$$

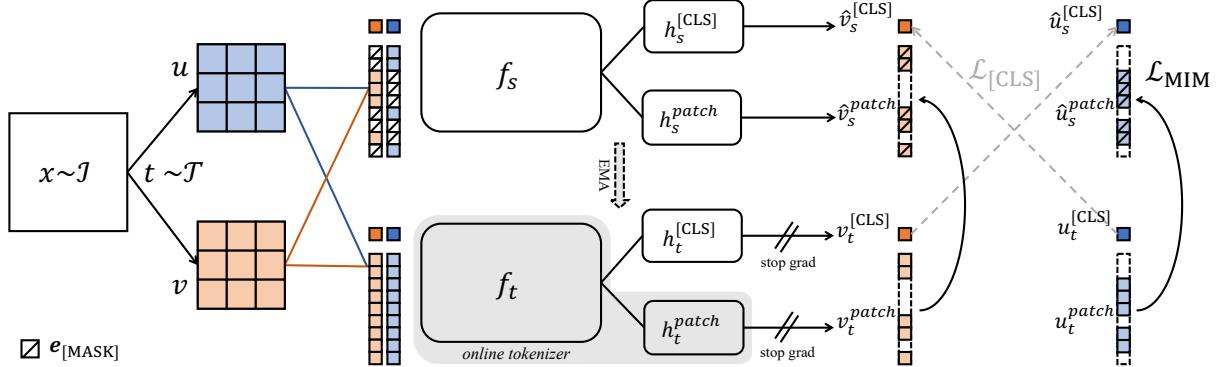
The teacher and the student share the same architecture consisting of a backbone  $f$  (*e.g.*, ViT) and a projection head  $h^{[\text{CLS}]}$ . The parameters of the student network  $\theta$  are Exponentially

Moving Averaged (EMA) to the parameters of teacher network  $\theta'$ . The loss is symmetrized by averaging with another cross-entropy term between  $v_s^{[\text{CLS}]}$  and  $u_t^{[\text{CLS}]}$ .

## 6.3 iBOT

We motivate our method by identifying the similar formulation of Eq. (6.1) and Eq. (6.2). A visual tokenizer parameterized by online  $\theta'$  instead of pre-fixed  $\phi$  thus arises naturally. In this section, we present iBOT, casting self-distillation as a *token-generation* self-supervised objective and perform MIM via self-distillation. We illustrate the framework of iBOT in Figure 6.3 and demonstrate the pseudo-code in Appendix E.1. In Section 6.3.2, we briefly introduce the architecture and pre-training setup.

### 6.3.1 Framework



**Figure 6.3. Overview of iBOT framework, performing masked image modeling with an *online tokenizer*.** Given two views  $u$  and  $v$  of an image  $x$ , each view is passed through a teacher network  $h_t \circ f_t$  and a student network  $h_s \circ f_s$ . iBOT minimizes two losses. The first loss  $\mathcal{L}_{[\text{CLS}]}$  is self-distillation between cross-view [CLS] tokens. The second loss  $\mathcal{L}_{\text{MIM}}$  is self-distillation between in-view patch tokens, with some tokens masked and replaced by  $e_{[\text{MASK}]}$  for the student network. The objective is to reconstruct the masked tokens with the teacher networks' outputs as supervision.

First, we perform blockwise masking [234] on the two augmented views  $u$  and  $v$  and

obtain their masked views  $\hat{\mathbf{u}}$  and  $\hat{\mathbf{v}}$ . Taking  $\hat{\mathbf{u}}$  as an example for simplicity, the student network outputs for the masked view  $\hat{\mathbf{u}}$  projections of its patch tokens  $\hat{\mathbf{u}}_s^{\text{patch}} = P(c_i|\hat{\mathbf{u}}; \theta)$  and the teacher network outputs for the non-masked view  $\mathbf{u}$  projections of its patch tokens  $\mathbf{u}_t^{\text{patch}} = P(c_i|\mathbf{u}; \theta')$ . We here define the training objective of MIM in iBOT as

$$\mathcal{L}_{\text{MIM}} = \sum_{i=1}^N m_i H(P(c_i|\mathbf{u}; \theta'), P(c_i|\hat{\mathbf{u}}; \theta)) . \quad (6.3)$$

We symmetrize the loss by averaging with another CE term between  $\hat{\mathbf{v}}_s^{\text{patch}}$  and  $\mathbf{v}_t^{\text{patch}}$ .

The backbone together with the projection head of teacher network  $h_t^{\text{patch}} \circ f_t$  is, therefore, a visual tokenizer that generates online token distributions for each masked patch token. The tokenizer used in iBOT is jointly learnable to MIM objective without a need of being pre-trained in an extra stage, a bonus feature of which is now its domain knowledge can be distilled from the current dataset rather than fixed to the specified dataset.

To ensure that the online tokenizer is semantically-meaningful, we perform self-distillation on [CLS] token of cross-view images such that visual semantics can be obtained via bootstrapping, as achieved by the majority of the self-supervised methods [173], [206], [230]. In practice, iBOT works with  $\mathcal{L}_{[\text{CLS}]}$  in Eq. (6.2) proposed in DINO [230], except that now we have  $\hat{\mathbf{u}}_s^{[\text{CLS}]}$  instead of  $\mathbf{u}_s^{[\text{CLS}]}$  as input for the student network. To further borrow the capability of semantics abstraction acquired from self-distillation on [CLS] token, we share the parameters of projection heads for [CLS] token and patch tokens, *i.e.*,  $h_s^{[\text{CLS}]} = h_s^{\text{patch}}$ ,  $h_t^{[\text{CLS}]} = h_t^{\text{patch}}$ . We empirically find that it produces better results than using separate heads.

Unlike tokenized words whose semantics are almost certain, image patch is ambiguous in its semantic meaning. Therefore, tokenization as one-hot discretization can be sub-optimal for images. In iBOT, we use the token distribution after softmax instead of the one-hot token id as a supervisory signal, which plays an important role in iBOT pre-training as shown in Table E.12.

### 6.3.2 Implementation

**Architecture.** We use the Vision Transformers [228] and Swin Transformers [238] with different amounts of parameters, ViT-S/16, ViT-B/16, ViT-L/16, and Swin-T/{7,14} as the backbone  $f$ . For ViTs, /16 denotes the patch size being 16. For Swins, /{7,14} denotes the window size being 7 or 14. We pre-train and fine-tune the Transformers with 224-size images, so the total number of patch tokens is 196. The projection head  $h$  is a 3-layer MLPs with  $l_2$ -normalized bottleneck following DINO [230]. Towards a better design to acquire visual semantics, we studied different sharing strategies between projection heads  $h^{[\text{CLS}]}$  and  $h^{\text{patch}}$ , considering that semantics obtained in distillation on [CLS] token helps the training of MIM on patch tokens. We empirically find that sharing the entire head prompts the best performance. We set the output dimension of the shared head to 8192.

**Pre-Training Setup.** We by default pre-train iBOT on ImageNet-1K [197] training set with AdamW [239] optimizer and a batch size of 1024. We pre-train iBOT using ViT-S/16 with 800 epochs, ViT-B/16 with 400 epochs, and Swin-T/{7,14} with 300 epochs. We also pre-train on ImageNet-22K training set with ViT-B/16 for 80 epochs and ViT-L/16 for 50 epochs. The learning rate is linearly ramped up during the first 10 epochs to its base value scaled with the total batch size:  $\text{lr} = 5e^{-4} \times \text{batch\_size}/256$ . We use random MIM, with prediction ratio  $r$  set as 0 with a probability of 0.5 and uniformly sampled from range [0.1, 0.5] with a probability of 0.5. We sum  $\mathcal{L}_{[\text{CLS}]}$  and  $\mathcal{L}_{\text{MIM}}$  up without scaling. More ablations on parameter setup are given in Appendix E.5.

## 6.4 Experiment

We first transfer iBOT to downstream tasks, following the standard evaluation protocols adopted in prior arts, the details of which are in Appendix E.3. We then study several interesting properties of Transformers pre-trained with iBOT. Finally, we give a brief ablation study on the crucial composing of iBOT.

### 6.4.1 Classification on ImageNet-1K

We consider five classification protocols on ImageNet-1K:  $k$ -NN, linear probing, fine-tuning, semi-supervised learning, and unsupervised learning.

**Table 6.1. ImageNet-1K  $k$ -NN and linear probing accuracy.** <sup>†</sup> denotes using selective kernel.  
<sup>‡</sup> denotes pre-training on ImageNet-22K.

Method	Arch.	Par.	im/s	Epo. <sup>1</sup>	$k$ -NN	Lin.
<i>SSL CNNs</i>						
MoCov3	RN50	23	1237	1600	-	74.6
SwAV	RN50	23	1237	2400	65.7	75.3
DINO	RN50	23	1237	3200	67.5	75.3
BYOL	RN200w2	250	123	2000	73.9	79.6
SCLRv2	RN152w3 <sup>†</sup>	794	46	2000	73.1	79.8
<i>SSL Transformers</i>						
MoCov3	ViT-S/16	21	1007	1200	-	73.4
MoCov3	ViT-B/16	85	312	1200	-	76.7
SwAV	ViT-S/16	21	1007	2400	66.3	73.5
DINO	ViT-S/16	21	1007	3200	74.5	77.0
DINO	ViT-B/16	85	312	1600	76.1	78.2
EsViT	Swin-T/7	28	726	1200	75.7	78.1
EsViT	Swin-T/14	28	593	1200	77.0	78.7
iBOT	ViT-S/16	21	1007	3200	75.2	77.9
iBOT	Swin-T/7	28	726	1200	75.3	78.6
iBOT	Swin-T/14	28	593	1200	76.2	79.3
iBOT	ViT-B/16	85	312	1600	77.1	79.5
iBOT	ViT-L/16	307	102	1200	<b>78.0</b>	81.0
iBOT <sup>‡</sup>	ViT-L/16	307	102	200	70.6	<b>81.7</b>

**Table 6.2. ImageNet-1K fine-tuning.**

Method	Arch.	Epo. <sup>1</sup>	Acc.
Rand.	ViT-S/16	-	79.9
MoCov3	ViT-S/16	600	81.4
BEiT	ViT-S/16	800	81.4
DINO	ViT-S/16	3200	82.0
iBOT	ViT-S/16	3200	<b>82.3</b>
Rand.	ViT-B/16	-	81.8
MoCov3	ViT-B/16	600	83.2
BEiT	ViT-B/16	800	83.4
DINO	ViT-B/16	1600	83.6
iBOT	ViT-B/16	1600	<b>84.0</b>
MoCov3	ViT-L/16	600	84.1
iBOT	ViT-L/16	1200	<b>84.7</b>
BEiT	ViT-L/16	800	<b>85.2</b>

**Table 6.3. ImageNet-1K fine-tuning.**

Pre-trained on ImageNet-22K.

Method	Arch.	Epo. <sup>1</sup>	Acc.
BEiT	ViT-B/16	150	83.7
iBOT	ViT-B/16	320	<b>84.4</b>
BEiT	ViT-L/16	150	86.0
iBOT	ViT-L/16	200	<b>86.6</b>

**$k$ -NN and Linear Probing.** To evaluate the quality of pre-trained features, we either use a  $k$ -nearest neighbor classifier ( $k$ -NN) or a linear classifier on the frozen representation. We follow the evaluation protocols in DINO [230]. For  $k$ -NN evaluation, we sweep over different numbers of nearest neighbors. For linear evaluation, we sweep over different learning rates. In Table 6.1, our method reaches a  $k$ -NN accuracy of 75.2% and linear probing accuracy 77.9% with ViT-S/16, and a  $k$ -NN accuracy 77.1% and linear probing

<sup>1</sup>Effective pre-training epochs accounting for actual trained images/views. See Appendix E.2 for details.

accuracy 79.5% with ViT-B/16, achieving state-of-the-art performance. With Swin-T/{7,14}, iBOT achieves a linear probing accuracy of 78.6% and 79.3% respectively, surpassing previous best result, 78.1% and 78.7% by EsViT [240]. With ViT-L/16 and ImageNet-22K as pre-training data, iBOT further achieves a linear probing accuracy 81.7%, surpassing previous state of the art, 81.3% with Swin-B/14 by EsViT. A linear probing accuracy of 79.5% with ViT-B/16 is comparable to 79.8% by SimCLRv2 with RN152 ( $3\times$ )<sup>†</sup> but with  $10\times$  less parameters. We underline that the performance gain over DINO gets larger (0.9% w/ ViT-S versus 1.3% w/ ViT-B) with more parameters, suggesting iBOT is more scalable to larger models.

**Fine-Tuning.** We study the fine-tuning on ImageNet-1K and pre-training on ImageNet-1K or ImageNet-22K. We focus on the comparison with self-supervised methods for Transformers and its supervised baseline. *Rand.* denotes the supervised baselines reported in [227]. As shown in Table 6.2, iBOT achieves an 82.3% and 84.0% top-1 accuracy with ViT-S/16 and ViT-B/16, respectively, yielding a performance gain of 0.3% and 0.4% compared to the previous state of the art, DINO. As shown in Table 6.3, iBOT pre-trained with ImageNet-22K achieves 84.4% and 86.6% top-1 accuracy with ViT-B/16 and ViT-L/16, respectively, outperforming ImageNet-22K pre-trained BEiT by 0.7% and 0.6%.

**Table 6.4. Semi-supervised learning on ImageNet-1K.** 1% and 10% denotes label fraction. SD denotes self-distillation.

Method	Arch.	1%	10%
SimCLRv2	RN50	57.9	68.1
BYOL	RN50	53.2	68.8
SwAV	RN50	53.9	70.2
SimCLRv2+SD	RN50	60.0	70.5
DINO	ViT-S/16	60.3	74.3
iBOT	ViT-S/16	<b>61.9</b>	<b>75.1</b>

**Table 6.5. Unsupervised learning on ImageNet-1K.** <sup>†</sup> denotes  $k$ -means clustering on frozen features.

Method	Arch.	ACC	ARI	NMI	FMI
Self-label <sup>†</sup>	RN50	30.5	16.2	75.4	-
InfoMin <sup>†</sup>	RN50	33.2	14.7	68.8	-
SCAN	RN50	39.9	27.5	72.0	-
DINO	ViT-S/16	41.4	29.8	76.8	32.8
iBOT	ViT-S/16	<b>43.4</b>	<b>32.8</b>	<b>78.6</b>	<b>35.6</b>

**Semi-Supervised and Unsupervised Learning.** For semi-supervised learning, we focus our comparison with methods following the *unsupervised pre-train, supervised fine-tune* paradigm. As shown in Table 6.4, iBOT advances DINO by 1.6% and 0.8% using 1% and 10% data, respectively, suggesting a higher label efficiency. For unsupervised learning, we use standard evaluation metrics, including accuracy (ACC), adjusted random index (ARI), normalized mutual information (NMI), and Fowlkes-Mallows index (FMI). We compare our methods to SimCLRv2 [241], Self-label [242], InfoMin [243], and SCAN [244]. As shown in Table 6.5. iBOT outperforms the previous state of the art by 2.0% in Acc and 1.8 in NMI, suggesting MIM helps the model learn stronger visual semantics on a global scale.

#### 6.4.2 Downstream Tasks

**Table 6.6. Object detection (Det.) & instance segmentation (ISeg.) on COCO and Semantic segmentation (Seg.) on ADE20K.** We report the results of ViT-S/16 (left) and ViT-B/16 (right). Seg.<sup>†</sup> denotes using a linear head for semantic segmentation.

Method	Arch.	Param.	Det.			Method	Det.			Seg. <sup>†</sup>	Seg.
			AP <sup>b</sup>	ISeg. AP <sup>m</sup>	Seg. mIoU		AP <sup>b</sup>	ISeg. AP <sup>m</sup>	mIoU		
Sup.	Swin-T	29	48.1	41.7	44.5	Sup.	49.8	43.2	35.4	46.6	
MoBY	Swin-T	29	48.1	41.5	44.1	BEiT	50.1	43.5	27.4	45.8	
Sup.	ViT-S/16	21	46.2	40.1	44.5	DINO	50.1	43.4	34.5	46.8	
iBOT	ViT-S/16	21	<b>49.4</b>	<b>42.6</b>	<b>45.4</b>	iBOT	<b>51.2</b>	<b>44.2</b>	<b>38.3</b>	<b>50.0</b>	

**Object Detection and Instance Segmentation on COCO.** Object detection and instance segmentation require simultaneous object location and classification. We consider Cascade Mask R-CNN [211], [245] that produces bounding boxes and instance masks simultaneously on COCO dataset [15]. Several recent works [238], [246] proposes Vision Transformers that suit dense downstream tasks. To compare, we include the results of supervised Swin-T [238] which shares approximate parameter numbers with ViT-S/16 and its self-supervised counterpart MoBY [247] in Table 6.6. iBOT improves ViT-S’s AP<sup>b</sup> from 46.2 to

49.4 and AP<sup>m</sup> from 40.1 to 42.6, surpassing both supervised Swin-T and its self-supervised counterpart by a nontrivial margin. With ViT-B/16, iBOT achieves an AP<sup>b</sup> of 51.2 and an AP<sup>m</sup> of 44.2, surpassing previous best results by a large margin.

**Semantic Segmentation on ADE20K.** Semantic segmentation can be seen as a pixel-level classification problem. We mainly consider two segmentation settings on ADE20K dataset [248]. First, similar to linear evaluation protocol in image classification, we evaluate the models on the fixed patch features from the Transformer encoder and only fine-tune a linear layer. This gives us a more explicit comparison of the quality of learned representations. Second, we use the task layer in UPerNet [249] and fine-tune the entire network. We report the mean intersection over union (mIoU) averaged over all semantic categories. From Table 6.6, we can see that iBOT advances its supervised baseline with ViT-S/16 with a large margin of 0.9 on mIoU, surpassing Swin-T. With ViT-B/16, iBOT advances previous best methods DINO by 3.2 on mIoU with UperNet. We notice a performance drop of BEiT using linear head, indicating BEiT’s features lack local semantics. As analyzed later, the property of strong local semantics induces a 2.9 mIoU gain compared to the supervised baseline with a linear head.

**Table 6.7. Transfer learning by fine-tuning pre-trained models on different datasets.** We report Top-1 accuracy of ViT-S/16 (left) and ViT-B/16 (right).

Method	Cif <sub>10</sub>	Cif <sub>100</sub>	iNa <sub>18</sub>	iNa <sub>19</sub>	Flwrs	Cars	Method	Cif <sub>10</sub>	Cif <sub>100</sub>	iNa <sub>18</sub>	iNa <sub>19</sub>	Flwrs	Cars
Rand.	99.0	89.5	70.7	76.6	98.2	92.1	Rand.	99.0	90.8	73.2	77.7	98.4	92.1
BEiT	98.6	87.4	68.5	76.5	96.4	92.1	BEiT	99.0	90.1	72.3	79.2	98.0	94.2
DINO	99.0	90.5	72.0	78.2	98.5	93.0	DINO	99.1	91.7	72.6	78.6	98.8	93.0
iBOT	<b>99.1</b>	<b>90.7</b>	<b>73.7</b>	<b>78.5</b>	<b>98.6</b>	<b>94.0</b>	iBOT	<b>99.2</b>	<b>92.2</b>	<b>74.6</b>	<b>79.6</b>	<b>98.9</b>	<b>94.3</b>

**Transfer Learning.** We study transfer learning where we pre-train on ImageNet-1K and fine-tune on several smaller datasets. We follow the training recipe and protocol used in [228]. The results are demonstrated in Table 6.7. While the results on several datasets (*e.g.*, CIFAR10, CIFAR100, Flowers, and Cars) have almost plateaued, iBOT consistently

performs favorably against other SSL frameworks, achieving state-of-the-art transfer results. We observe greater performance gain over DINO in larger datasets like iNaturalist18 and iNaturalist19, indicating the results are still far from saturation. We also find that with larger models, we typically get larger performance gain compared with DINO (*e.g.*, 1.7% with ViT-S-16 versus 2.0% with ViT-B/16 on iNaturalist18, and 0.3% with ViT/S-16 versus 1.0% with ViT-B/16 on iNaturalist19).

#### 6.4.3 Properties of ViT Trained with MIM

In the previous sections, we have shown the priority of iBOT on various tasks and datasets. To reveal the strengths of iBOT pre-trained Vision Transformers, we analyze its property from several aspects.



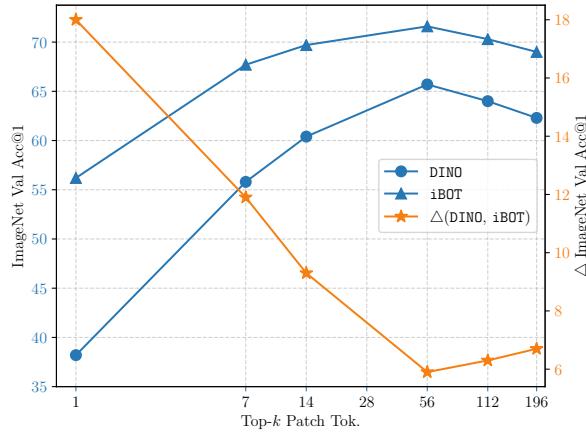
**Figure 6.4. Pattern layout of patch tokens.** Two left figures showcase patterns, *headlight of the vehicle* and *ear of the dog*, that share part semantics. Two right figures showcase patterns, *striped* and *curly surface*, that share part textures.

**What Patterns Does MIM Learn?** The output from the projection head used for self-distillation depicts for patch token a probabilistic distribution. To help understand what patterns MIM induces to learn, we visualize several pattern layouts. We use 800-epoch pre-trained ViT-S/16 and visualize the top-36 patches with the highest confidence on ImageNet-1K validation set. We visualize a  $5 \times$  context for each  $16 \times 16$  patch (colored orange). We observe the emergence of both high-level semantics and low-level details. As shown in Figure 6.4, several patches are grouped with clear semantic meaning, *e.g.*, *headlight*

and *dog’s ear*. Such behavior stands a distinct contrast with the offline tokenizer used in BEiT [234], which encapsulates mostly low-level details as shown in Figure E.10. Apart from patch patterns that share high-level semantics, we also observe clusters accounting for low-level textures, indicating the diversity of learned part patterns. The comparison with previous work [230], [234] and the visualization of more pattern layouts are provided in Appendix E.7.1.

**How Does MIM Help Image Recognition?** To illustrate how the property of better part semantics can help image recognition, we use *part-wise linear classification* to study the relationship between representations of patch tokens and [CLS] token. Specifically, we average  $k$  patch tokens with the top- $k$  highest self-attention scores. The results are demonstrated in Figure 6.5. While the performance gap between DINO and iBOT is only 0.9% in the standard setting (77.9% v.s. 77.0%) with [CLS] token, we observe that iBOT outperforms DINO when using the patch representations directly. We observe that using top-56 patch tokens yields an optimal result, and iBOT is 5.9% higher than DINO. The performance gap becomes more prominent when using fewer patch tokens. When using only the patch token with the highest self-attention score, iBOT advances by 17.9%. These results reveal much semantic information in iBOT representations for patch tokens, which helps the model to be more robust to the loss of local details and further boosts its performance on image-level recognition.

**Discriminative Parts in Self-Attention Map.** To analyze, we visualize the self-attention map with ViT-S/16. We choose [CLS] token as the query and visualize attention maps from different heads of the last layer with different colors, as shown in Figure 6.6. Of particular interest, we indicate that iBOT shows a solid ability to separate different objects or different parts of one object apart. For example, in the leftmost figure, we observe iBOT fairly distinct the bird from the tree branch. Also, iBOT focuses mainly on the discriminative parts of the object (*e.g.*, *the wheel of the car*, *the beak of the bird*). These properties are crucial



**Figure 6.5. Part-wise linear probing accuracy.** Top- $k$  tokens with the highest attention scores are averaged for classification.



**Figure 6.6. Visualization for self-attention map.** Self-attention map from multiple heads are visualized with different color.

for iBOT to excel at image recognition, especially in complicated scenarios with object occlusion or distracting instances. While these properties are not unique strengths brought by MIM and we observe similar behaviors in DINO, we show in Appendix E.7.2 that iBOT generally gives better visualized results.

#### 6.4.4 Robustness

**Table 6.8. Robustness evaluation of pre-trained models against background change, occlusion, and out-of-distribution examples.**

Method	Background Change						Clean IN-9	Occlusion		Out-of-Dist.		Clean IN
	O.F.	M.S.	M.R.	M.N.	N.F.	O.BB.	O.BT.	S. <sub>.5</sub>	NS. <sub>.5</sub>	IN-A	IN-C ↓	
DINO	89.2	89.2	80.4	78.3	52.0	21.9	<b>18.4</b>	96.4	64.7	42.0	12.3	51.7
iBOT	<b>90.9</b>	<b>89.7</b>	<b>81.7</b>	<b>80.3</b>	<b>53.5</b>	<b>22.7</b>	17.4	<b>96.8</b>	<b>65.9</b>	<b>43.4</b>	<b>13.8</b>	<b>48.1</b>

The above-mentioned properties brought by MIM objective can improve the model’s robustness to uncommon examples. We quantitatively benchmark robustness in terms of 3 aspects: background change, occlusion, and out-of-distribution examples, with a ViT-S/16 pre-trained for 800 epochs and then linearly evaluated for 100 epochs. Results are shown

in Table 6.8. For background change, we study images under 7 types of change, detailed in Appendix E.4. iBOT is more robust against background changes except for O.BT.. For occlusion, we study the linear accuracy with salient and non-salient patch dropping following [250] with an information loss ratio of 0.5. iBOT has a smaller performance drop under both settings. For out-of-distribution examples, we study natural adversarial examples in ImageNet-A [251] and image corruptions in ImageNet-C [252]. iBOT has higher accuracy on the ImageNet-A and a smaller mean corruptions error (mCE) on the ImageNet-C.

#### 6.4.5 Ablation Study on Tokenizer

In this section, we ablate the importance of using a semantically meaningful tokenizer using a 300-epoch pre-trained ViT-S/16 with a prediction ratio  $r = 0.3$  and without multi-crop augmentation. Additional ablations are given in Appendix E.5. iBOT works with self-distillation on [CLS] token with cross-view images ( $\mathcal{L}_{[CLS]}$ ) to acquire visual semantics. To verify, we conduct experiments to perform MIM without  $\mathcal{L}_{[CLS]}$  or with alternative models as visual tokenizer. Specifically,  $\circ$  denotes a standalone DINO and  $\triangle$  denotes a pre-trained DALL-E encoder [235]. We find that performing MIM without  $\mathcal{L}_{[CLS]}$  leads

**Table 6.9. Effect of design choices of semantically meaningful tokenization.**

Method	$\mathcal{L}_{MIM}$	$\mathcal{L}_{[CLS]}$	SH	k-NN	Linear	Fine-tuned
iBOT	✓	✓	✓	69.1	74.2	81.5
	✓	✓	✗	69.0	73.8	81.5
	✓	✗	-	9.5	29.8	79.4
	○	✗	-	44.3	60.0	81.7
	△	✗	-	6.9	23.5	81.4
BEiT	△	✗	-	67.9	72.5	80.6
DINO	✗	✓	-	48.0	62.7	81.2
BEiT + DINO	△	✓	-			

○: standalone DINO (w/o mcrop, 300-epoch)

△: pre-trained DALL-E encoder

to undesirable results of 9.5%  $k$ -NN accuracy and 29.8% linear accuracy, indicating that

visual semantics can hardly be obtained with only MIM. While semantics emerges with a standalone DINO as a visual tokenizer, it is still far from reaching a decent result (44.3% versus 69.1% in  $k$ -NN accuracy). Comparing iBOT with multi-tasking of DINO and BEiT (DINO+BEiT), we see the strengths of merging the semantics acquired by self-distillation with the visual tokenizer with an 11.5% advance in linear probing and 0.3% in fine-tuning. Moreover, we empirically observe a performance improvement using a Shared projection Head (SH) for [CLS] token and patch tokens, which shares the semantics acquired in [CLS] token to MIM.

## 6.5 Related Work

**Visual Representation Learning.** Most self-supervised methods assume an augmentation invariance of images and achieve so by enforcing similarity over distorted views of one image while avoiding model collapse. Avoiding collapse can be achieved by noise-contrastive estimation with negative samples [169], [172], [173], introducing asymmetric network [206], [207], or explicitly enforcing the distribution of image distribution over the channel to be uniform as well as one-hot [204], [230], [253]. In fact, the idea of simultaneously enforcing distribution uniform and one-hot is hidden from earlier studies performing representation learning via clustering [204], [254], [255], where the cluster assignment naturally meets these two requirements. Other methods rely on handcrafted pretext tasks and assume the image representation should instead be aware of image augmentation by solving image jigsaw puzzle [196], [256], predicting rotation [257] or relative position [258].

**Masked Prediction in Images.** Predicting masked images parts is a popular self-supervised pretext task drawing on the idea of auto-encoding and has been previously achieved by either recovering raw pixels [215], [232], [259] or mask contrastive learning [186], [260]. Recently, it is formulated into MIM [234], [236] with a discrete VAE [235], [261] as visual tokenizer. As a counterpart of MLM in NLP, MIM eases masked prediction into

a classification problem supervised by labels output from the tokenizer, mitigating the problem of excessive focus on high-frequency details. Concurrently, masked image prediction has been explored in the field of multi-modality, *i.e.*, vision-language representation learning. These methods operate on local regions instead of global images thus reply on pre-trained detection models, *i.e.*, Faster-RCNN [139] to propose regions of interest. [262]–[264] perform masked region classification tasking the category distribution output from the detection model as the ground-truth.

## 6.6 Conclusion

In this chapter, we study BERT-like pre-training for Vision Transformers and underline the significance of a semantically meaningful visual tokenizer. We present a self-supervised framework iBOT that performs masked image modeling via self-distillation with an online tokenizer, achieving state-of-the-art results on downstream tasks related to classification, object detection, instance segmentation, and semantic segmentation. Of particular interest, we identify an emerging part-level semantics for models trained with MIM that helps for not only recognition accuracy but also robustness against common image corruptions. In the future, we plan to scale up iBOT to a larger dataset (*e.g.*, ImageNet-22K) or larger model size (*e.g.*, ViT-L/16 and ViT-H/16) and investigate whether MIM can help Vision Transformers more scalable to unlabelled data in the wild.

# Chapter 7

## Conclusion

### 7.1 Summary

In this dissertation, I have presented a total of 5 research projects. All of them aim at providing a path towards modeling long-range dependencies for 2D visual perception.

Part I describes neural architectures that are capable of modeling long-range relations effectively and efficiently. Chapter 2 improves convolutional neural networks with dynamic scaling policies, augmenting all the layers with an Elastic module and demonstrating model robustness to scale variations. Chapter 3 pushes the idea further by enabling global receptive fields in all the layers, replacing all spatial convolutions with axial-attention layers and capturing context with precise position. Chapter 4 further extends the attention mechanism beyond pixel grids to segmentation masks by introducing a mask transformer between pixels and masks. The transformer captures relations not only between two pixels, but also between two object masks and between pixels and masks.

Part II is focused on self-supervised training of long-range models, mitigating the data efficiency problem of learning long-range dependencies. Chapter 5 improves the contrastive learning framework by formulating it as a semi-supervised learning problem and introducing a consistency term. Chapter 6 studies the pretext task, masked image modeling, which is particularly suitable and beneficial to long-range models or

transformers.

## 7.2 Future Directions

In this section, I will discuss a few future directions that I find interesting and promising given the current progress of long-range architectures and training pipelines.

### 7.2.1 Dynamic Computation

Convolution is previously the default operator for 2D image models and is very efficient to compute on full 2D grids. However, the information density on a 2D image is never a uniform distribution, which leads to an intuitive idea of dropping computation on less informative regions such as a white wall. This idea is not new but is less studied due to its incompatibility with convolution which is efficient only on full 2D grids. But this is not a problem any more with vision transformers that process images as a flattened 1D sequence of tokens or patches. Processing half of the tokens or patches is still efficient with transformers. In this way, we could either accelerate models or improve models with a fixed computation budget, but how to do it efficiently in model training and inference remains an open problem.

### 7.2.2 Long-Range Modeling for Videos

Video recognition, compared with 2D image recognition, involves an extra temporal dimension beside the spatial axes. Mining information from the temporal axis is so challenging that many current methods overfit a lot to spatial dimensions. In this case, modeling long-range dependencies and extracting sparse but useful signals in the temporal axis is a critical step towards overcoming such overfitting challenges. Apart from these challenges and opportunities, videos also provide us with unique signals such as natural correspondence between objects and frames that is otherwise difficult to obtain in 2D

natural images. Such correspondence could potentially be better leveraged by long-range models, for example with masked video modeling tasks.

### 7.2.3 Relation Between Vision and Language

Modeling relations between image pixels, object masks, or video frames could eventually saturate when the training signals for longer-range models become more and more sparse and long-tail. A complementary source of signal lies in natural languages which are naturally (or artificially) compositional. This source of signal is particularly scalable due to the massive amount of image-text correspondence existing on the Internet, and the large scale language models trained in a self-supervised manner. In this case, by bridging vision and language modalities, for example with visual grounding, we might be able to exploit the compositionality of natural languages to learn a more decoupled and generalizable representation for future vision models.

# Appendix A

## ELASTIC: Improving CNNs with Dynamic Scaling Policies

### A.1 sElastic (simple Elastic)

A simple way of augmenting current models with Elastic is directly replacing bottlenecks by Elastic bottlenecks. This leads to models with less FLOPs and exactly the same number of parameters, which we refer to as sElastic (simple Elastic). This is in comparison to Elastic models that maintain the number of FLOPs and parameters. As shown in Table A.1, sElastic already outperforms some of the original models, with less FLOPs. Note that DLA-X60+sElastic in Table A.1 is equivalent to DLA-X60+Elastic (in Table 2.2 in the original chapter), i.e. we do not add/remove layers in different scales.

### A.2 Elastic Architecture Details

SElastic already outperforms original models. However, only applying downsamplings equivalently shifts computation from low level to higher level, which could cause lack of low level features to support high level processing. Also, sElastic reduces FLOPs so that its accuracy is not fairly comparable with the original model. For these two reasons, we rearrange computation distribution in each resolution, and this leads to our final Elastic

Model	# Params	FLOPs	Top-1	Top-5
ResNext50	25.0M	4.2B	22.2	-
ResNext50 <sup>*</sup>	25.0M	4.2B	22.23	6.25
ResNext50+sElastic	25.0M	3.4B	<b>22.03</b>	<b>6.07</b>
ResNeXt50+Elastic	25.2M	4.2B	21.56	5.83
DLA-X60	17.6M	3.6B	21.8	-
DLA-X60 <sup>*</sup>	17.6M	3.6B	21.92	6.03
DLA-X60+sElastic	17.6M	3.2B	<b>21.25</b>	<b>5.71</b>
DLA-X60+Elastic	17.6M	3.2B	21.25	5.71
DLA-X102	26.8M	6.0B	21.5	-
DLA-X102+sElastic	26.8M	5.0B	<b>21.0</b>	<b>5.66</b>
DLA-X102+Elastic	24.9M	6.0B	20.71	5.38

**Table A.1. Error rates for sElastic on the ImageNet validation set.** sElastic models with reduced FLOPs already perform better than some of the original models. We also provide the Elastic versions from the original chapter as a reference.

model.

Consider ResNeXt-50 as an example. The original model assigns [3, 4, 6, 3] blocks respectively to [56, 28, 14, 7] four scales. As shown in Table A.3, sElastic simply replaces original bottlenecks with Elastic bottlenecks. In Elastic, we roughly match the scale distribution of the original model by assigning [6, 8, 5, 3] blocks to those resolutions, as shown in Table A.3. Note that half of each block processes information at a higher level. This modification also leads to matched number of parameters, and matched number of FLOPs. For ResNeXt101, we use a block design of [12, 14, 20, 3]. DenseNet+Elastic and DLA+Elastic architectures are shown respectively in Table A.4 and Table A.2. Note that these block designs were picked to match the original number of parameters and FLOPs, so we didn't tune them as hyper-parameters. Tuning them could probably lead to even lower error rates.

**Table A.2. DLA model architectures.** Following DLA, we show our DLA classification architectures in the table. Split32 means a ResNeXt bottleneck with 32 paths while Split50 means a ResNeXt bottleneck with 50 paths. Stages 3 to 6 show d-n where d is the aggregation depth and n is the number of channels.

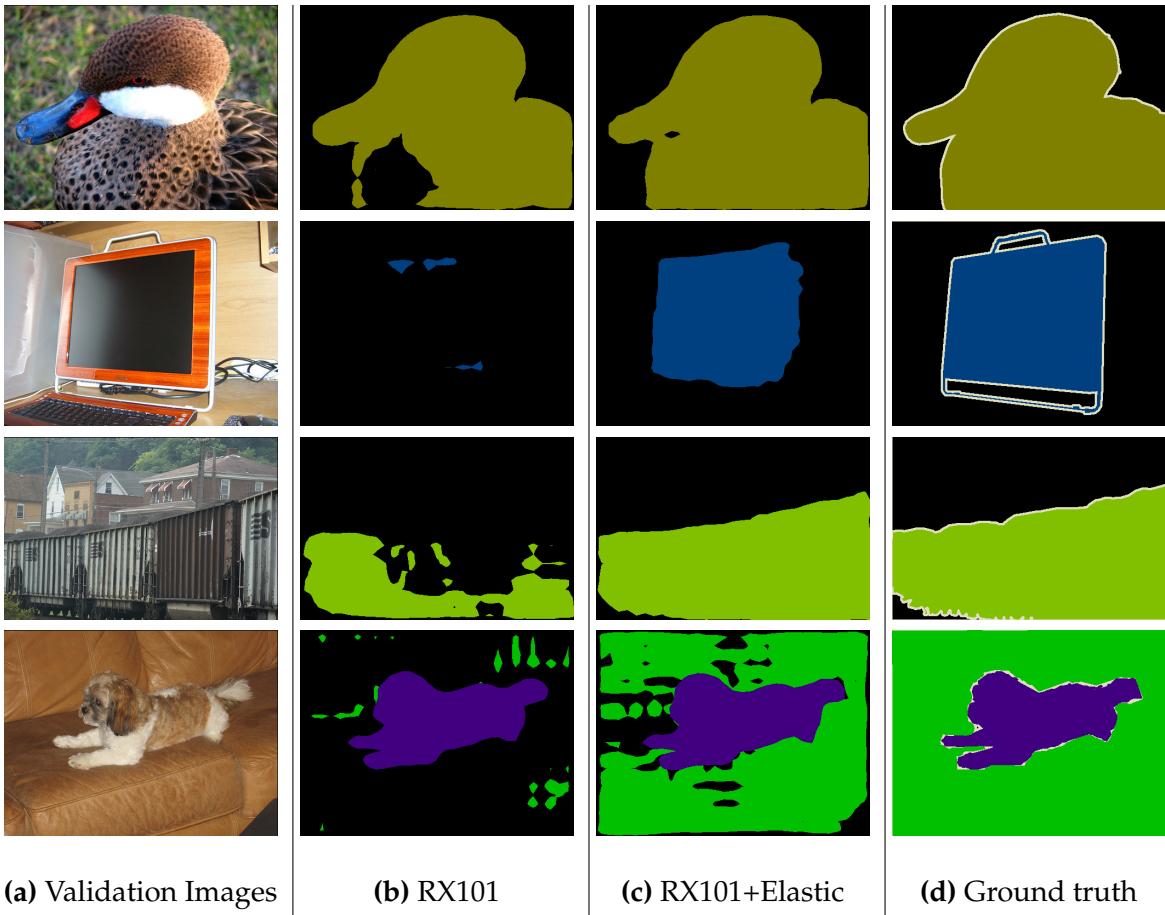
Name	Block	Stage 3	Stage 4	Stage 5	Stage 6	Params.	FLOPs
DLA-X60	Split32	1-128	2-256	3-512	1-1024	$17.7 \times 10^6$	$3.6 \times 10^9$
DLA-X60+Elastic	Split32+Elastic	1-128	2-256	3-512	1-1024	$17.7 \times 10^6$	$3.2 \times 10^9$
DLA-X102	Split32	1-128	3-256	4-512	1-1024	$26.8 \times 10^6$	$6.0 \times 10^9$
DLA-X102+sElastic	Split32+Elastic	1-128	3-256	4-512	1-1024	$26.8 \times 10^6$	$5.0 \times 10^9$
DLA-X102+Elastic	Split50+Elastic	3-128	3-256	3-512	1-1024	$24.9 \times 10^6$	$6.0 \times 10^9$

stage	ResNeXt50	ResNeXt50+sElastic			ResNeXt50+Elastic		
	7×7, 64, stride 2, 3×3 max pool, stride 2						
conv1							
conv2 56×56	$\begin{bmatrix} 1\times 1, 128 \\ 3\times 3, 128, C=32 \\ 1\times 1, 256 \end{bmatrix} \times 3$	$1\times 1, 64$ $3\times 3, 64, C=16 + 1\times 1, 256$ $2\times up, 56\times 56$	$2\times down, 28\times 28$ $1\times 1, 64$ $3\times 3, 64, C=16 + 1\times 1, 256$ $2\times up, 56\times 56$	$\times 3$	$1\times 1, 64$ $3\times 3, 64, C=16 + 1\times 1, 256$ $2\times up, 56\times 56$	$2\times down, 28\times 28$ $1\times 1, 64$ $3\times 3, 64, C=16 + 1\times 1, 256$ $2\times up, 56\times 56$	$\times 6$
conv3 28×28	$\begin{bmatrix} 1\times 1, 256 \\ 3\times 3, 256, C=32 \\ 1\times 1, 512 \end{bmatrix} \times 4$	$1\times 1, 128$ $3\times 3, 128, C=16 + 1\times 1, 512$ $2\times up, 28\times 28$	$2\times down, 14\times 14$ $1\times 1, 128$ $3\times 3, 128, C=16 + 1\times 1, 512$ $2\times up, 28\times 28$	$\times 4$	$1\times 1, 128$ $3\times 3, 128, C=16 + 1\times 1, 512$ $2\times up, 28\times 28$	$2\times down, 14\times 14$ $1\times 1, 128$ $3\times 3, 128, C=16 + 1\times 1, 512$ $2\times up, 28\times 28$	$\times 8$
conv4 14×14	$\begin{bmatrix} 1\times 1, 512 \\ 3\times 3, 512, C=32 \\ 1\times 1, 1024 \end{bmatrix} \times 6$	$1\times 1, 256$ $3\times 3, 256, C=16 + 1\times 1, 1024$ $2\times up, 14\times 14$	$2\times down, 7\times 7$ $1\times 1, 256$ $3\times 3, 256, C=16 + 1\times 1, 1024$ $2\times up, 14\times 14$	$\times 6$	$1\times 1, 256$ $3\times 3, 256, C=16 + 1\times 1, 1024$ $2\times up, 14\times 14$	$2\times down, 7\times 7$ $1\times 1, 256$ $3\times 3, 256, C=16 + 1\times 1, 1024$ $2\times up, 14\times 14$	$\times 5$
conv5 7×7			$1\times 1, 1024$ $3\times 3, 1024, C=32$ $1\times 1, 2048$	$\times 3$			
1×1	global average pool, 1000-d fc, softmax						
Params.	$25.0 \times 10^6$	$25.0 \times 10^6$				$25.2 \times 10^6$	
FLOPs	$4.2 \times 10^9$	$3.4 \times 10^9$				$4.2 \times 10^9$	

**Table A.3. ResNeXt50 vs. ResNeXt50+sElastic vs. ResNeXt50+Elastic.** ResNeXt50+Elastic employs two resolutions in each block, and keeps output resolution high for more blocks, compared with ResNeXt50.

stage	DenseNet201	DenseNet201+Elastic
conv1	$7 \times 7, 64$ , stride 2, $3 \times 3$ max pool, stride 2	
conv2 $56 \times 56$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 32 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 32 \end{bmatrix} + \begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 32 \end{bmatrix} \times 10$ $2 \times \text{down}, 28 \times 28$ $2 \times \text{up}, 56 \times 56$
trans1	$1 \times 1$ conv, $2 \times 2$ average pool, stride 2	
conv3 $28 \times 28$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 32 \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 32 \end{bmatrix} + \begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 32 \end{bmatrix} \times 20$ $2 \times \text{down}, 14 \times 14$ $2 \times \text{up}, 28 \times 28$
trans2	$1 \times 1$ conv, $2 \times 2$ average pool, stride 2	
conv4 $14 \times 14$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 32 \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 32 \end{bmatrix} + \begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 32 \end{bmatrix} \times 40$ $2 \times \text{down}, 7 \times 7$ $2 \times \text{up}, 14 \times 14$
trans3	$1 \times 1$ conv, $2 \times 2$ average pool, stride 2	
conv5 $7 \times 7$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 32 \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 32 \end{bmatrix} \times 30$
$1 \times 1$	global average pool, 1000-d fc, softmax	
Params.	$20.0 \times 10^6$	$19.5 \times 10^6$
FLOPs	$4.4 \times 10^9$	$4.2 \times 10^9$

**Table A.4. DenseNet201 vs. DenseNet201+Elastic.** DenseNet+Elastic follows a similar modification as ResNeXt+Elastic, i.e. two resolutions in each block and more blocks in high resolutions.



**Figure A.1. Semantic segmentation results on PASCAL VOC.** Elastic improves most on scale-challenging images.

### A.3 Semantic Segmentation Results

Some visualizations of our semantic segmentation results are shown in Figure A.1, demonstrating that Elastic segments scale-challenging objects well on PASCAL VOC.

# Appendix B

## Axial-DeepLab: Stand-Alone Axial-Attention for Panoptic Segmentation

### B.1 Runtime

In this section, we profile our Conv-Stem Axial-ResNet-L in a common setting: 224x224 feed-forward with batch size 1, on a V100 GPU, averaged over 5 runs. The time includes input standardization, and the last projection to 1000 logits. Our model takes 16.54 ms. For comparison, we list our TensorFlow runs of some popular models at hand (with comparable flops). To provide more context, we take entries from [265] for reference (A Titan X Pascal is used in [265], but the PyTorch code is more optimized). Our runtime is roughly at the same level of ResNeXt-101 (32x4d), SE-ResNet-101, ResNet-152, and DenseNet-201 (k=32).

Note that we directly benchmark with our code optimized for TPU execution, with channels being the last dimension. Empirically, the generated graph involves transposing between NCHW and NHWC, before and after almost every conv2d operation. (This effect also puts Xception-71 at a disadvantage because of its separable conv design.) Further optimizing this could lead to faster inference.

**Table B.1.** Runtime of Axial-ResNet-L on a  $224 \times 224$  image

Model	Our Profile (ms)	[265] (ms)
Axial-ResNet-L	16.54	-
Stand-Alone-L [71]	18.05	-
Xception-71 [101], [127]	24.85	-
ResNet-101 [10]	10.08	8.9
ResNet-152 [10]	14.43	14.31
ResNeXt-101 (32x4d) [16]	-	17.05
SE-ResNet-101 [266]	-	15.10
SE-ResNeXt-101 (32x4d) [266]	-	24.96
DenseNet-201 (k=32) [11]	-	17.15

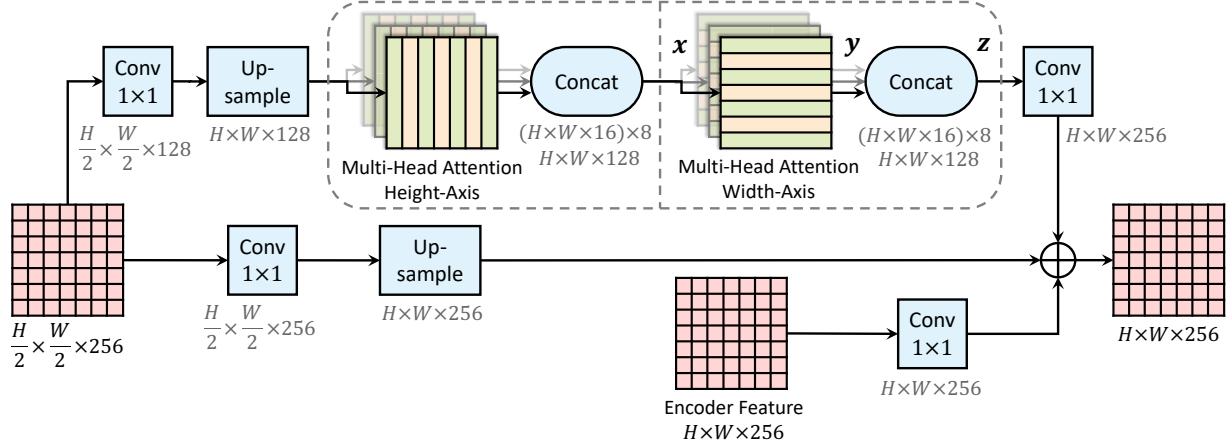
We observe that our Conv-Stem Axial-ResNet-L runs faster than Conv-Stem Stand-Alone-L [71], although we split one layer into two. This is because our axial-attention makes better use of existing kernels:

- The width-axis attention is parallelizable over height-axis, i.e. this is a large batch of 1d row operations (the batch size is the height of the input).
- Axial attention avoids extracting 2d memory blocks with pads, splits and concatenations, which are not efficient on accelerators.

## B.2 Axial-Decoder

Axial-DeepLab employs dual convolutional decoders [78]. In this section, we explore a setting with a *single* axial-decoder instead. In the axial-decoder module, we apply one axial-attention block at each upsampling stage. In Figure B.1, we show an example axial-decoder in Axial-DeepLab-L from output stride 8 to output stride 4. We apply three such blocks, analogous to the three  $5 \times 5$  convolutions in Panoptic-DeepLab [78].

**Importance of Output Stride and Axial-Decoder:** In Table B.2, we experiment with the effect of output stride and axial-decoder (*i.e.*, replacing dual decoders with axial-attention



**Figure B.1.** An axial-decoder block. We augment an axial-attention block with up-samplings, and encoder features

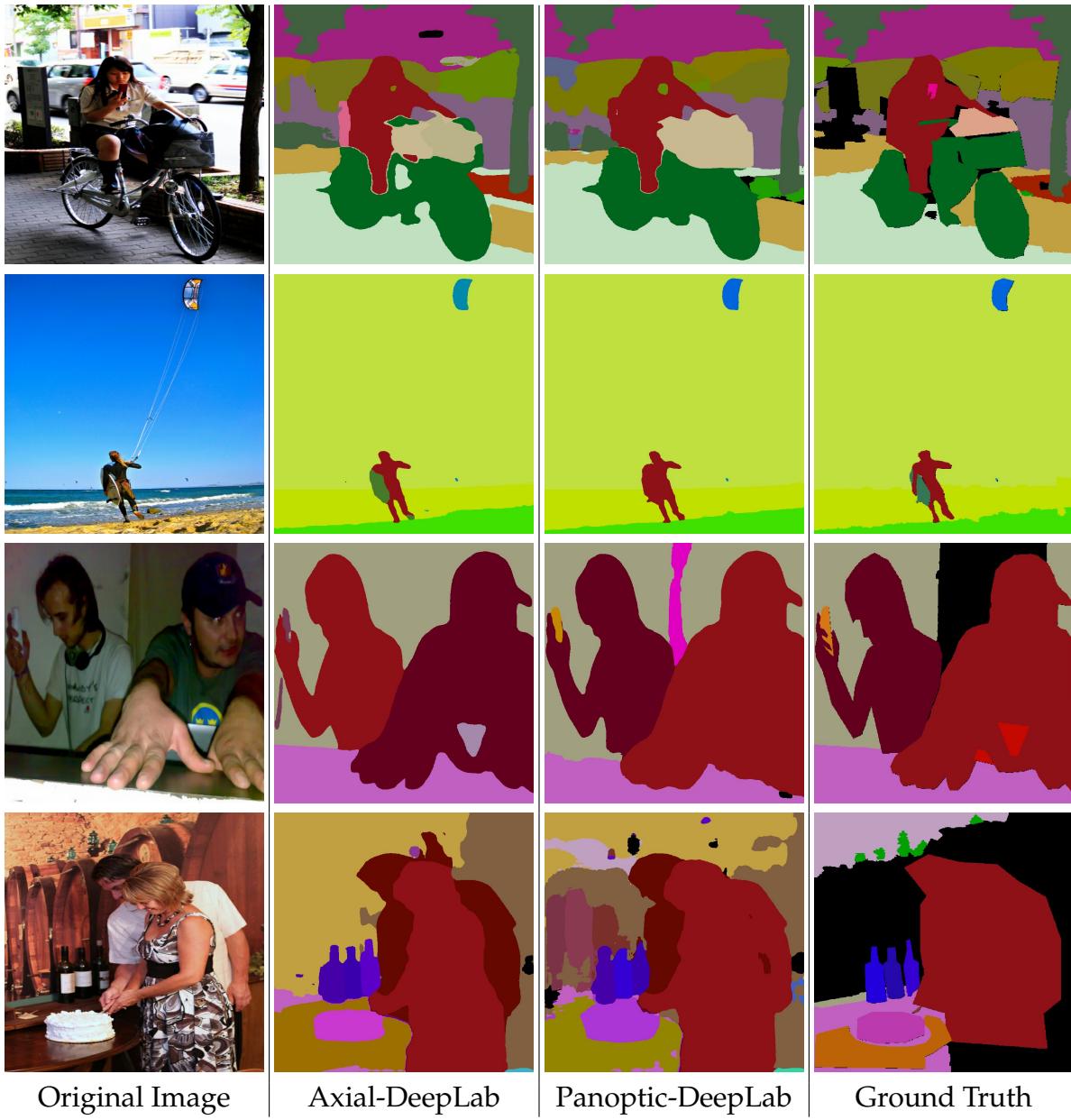
blocks). As shown in the table, our models are robust to output stride, and using axial-decoder is able to yield similar results. Our simple axial-decoder design works as well as dual convolutional decoders.

**Table B.2.** Ablating output strides and decoder types on Cityscapes val set. **ASPP:** Atrous spatial pyramid pooling. **OS:** Output stride (*i.e.*, the ratio of image resolution to final feature resolution in backbone). **AD:** Use axial-decoder in Axial-DeepLab

Backbone	ASPP	OS	AD	Params	M-Adds	PQ	AP	mIoU
Xception-71	✓	16		46.7M	547.7B	63.2	35.0	80.2
Axial-ResNet-L		16		44.9M	687.4B	63.9	35.8	81.0
Axial-ResNet-L		32		45.2M	525.2B	63.9	36.3	80.9
Axial-ResNet-L		16	✓	45.4M	722.7B	63.7	36.9	80.7
Axial-ResNet-L		32	✓	45.9M	577.8B	64.0	37.1	81.0

### B.3 COCO Visualization

In Figure B.2, we visualize some panoptic segmentation results on COCO val set. Our Axial-DeepLab-L demonstrates robustness to occlusion, compared with Panoptic-DeepLab (Xception-71).



**Figure B.2.** Visualization on COCO val set. Axial-DeepLab shows robustness to occlusion. In row 1 and row 4, Axial-DeepLab captures the occluded left leg and the remote control cable respectively, which are not even present in ground truth labels. In the last row, Axial-DeepLab distinguishes one person occluding another correctly, whereas the ground truth treats them as one instance

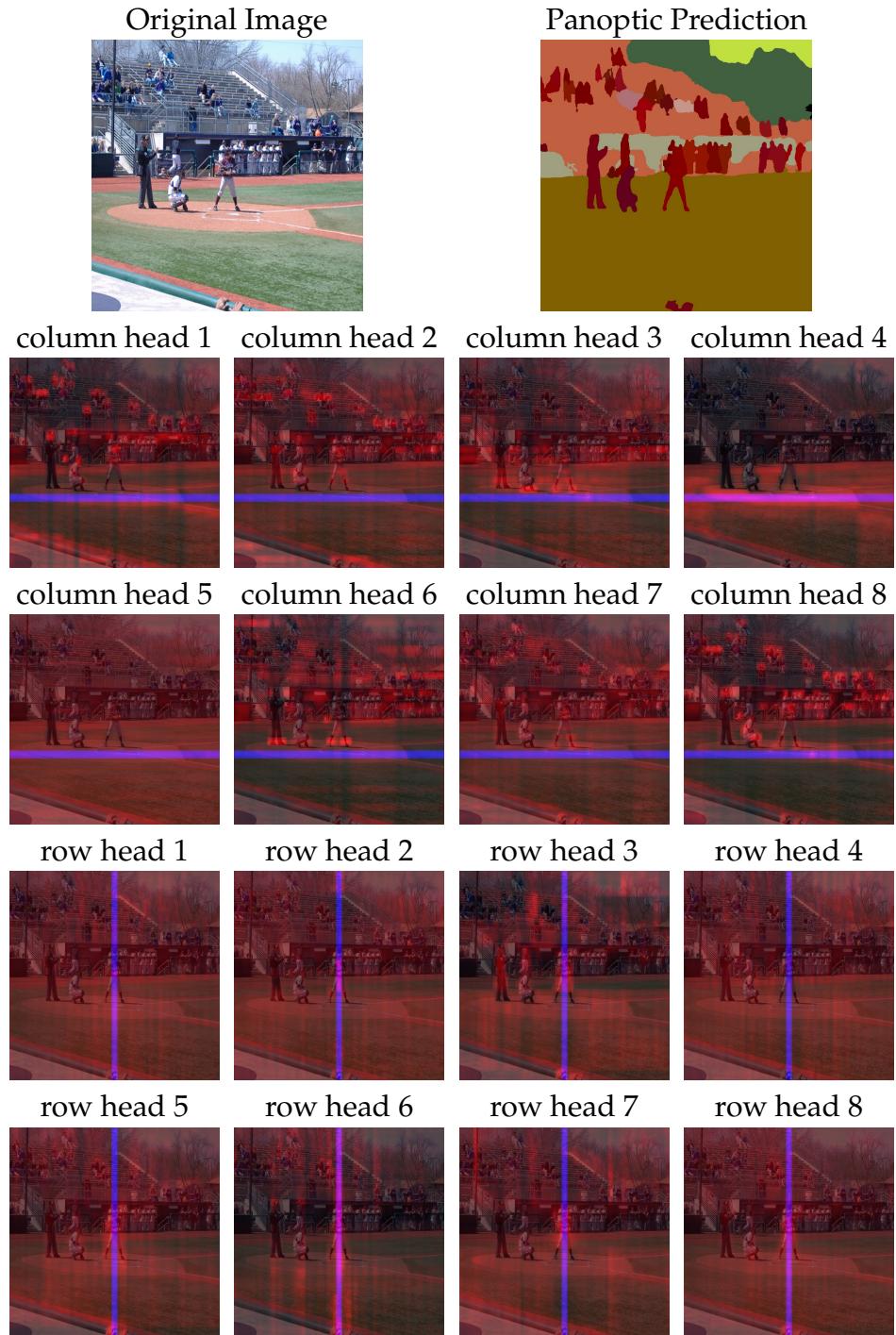
In Figure B.3 and Figure B.4, we visualize the attention maps of our Axial-DeepLab-L on COCO val set. We take a row of pixels, and visualize their column (height-axis) attention in all 8 heads. Then, we take a column, and visualize their row attention. We visualize a low level block (stage 3 block 2) and a high level block (stage 4 block 3), which are respectively the first block and the last block with resolution  $65 \times 65$ , in the setting of output stride 16. We notice that in our multi-head axial-attention, some heads learn to focus on local details while some others focus on long range context. Additionally, we find that some heads are able to capture positional information and some others learn to correlate with semantic concepts

In Figure B.5, we compare Axial-DeepLab with Panoptic-DeepLab [78], in terms of the three training loss functions, defined in Panoptic-DeepLab [78]. We observe that Axial-DeepLab is able to fit data better, especially on the offset prediction task. This also demonstrates the effectiveness of our position-sensitive attention design, and the long range modeling ability of axial-attention.

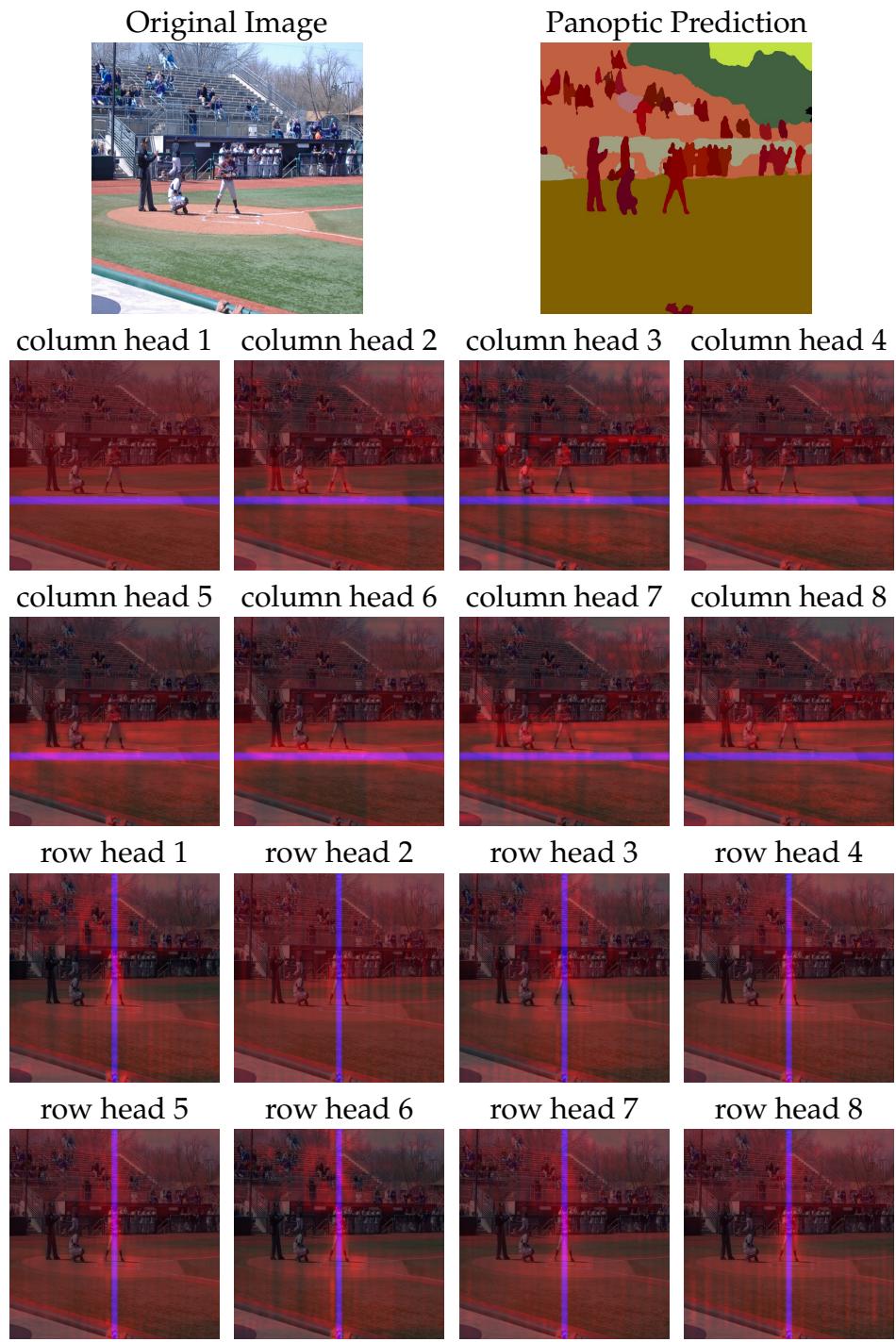
## B.4 Raw Data

In companion to Figure 3.3 of the main chapter where we compare parameters and M-Adds against accuracy on ImageNet classification, we also show the performance of our models in Table B.3.

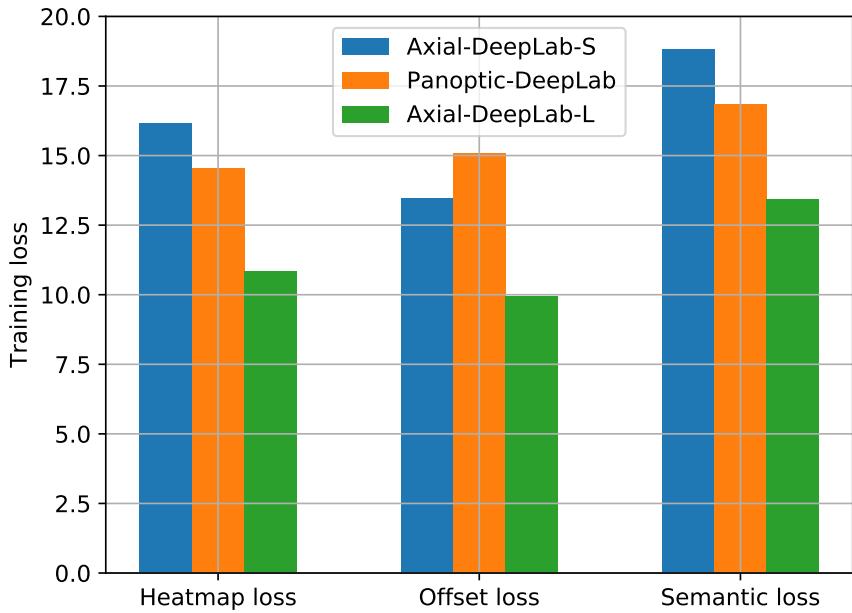
In companion to Figure 3.4 of the main chapter where we demonstrate the relative improvements of Axial-DeepLab-L over Panoptic-DeepLab (Xception-71) in our scale stress test on COCO, we also show the raw performance of both models in Figure B.6.



**Figure B.3.** Attention maps in block 2 of stage 3. Blue pixels are queries that we take, and red pixels indicate the corresponding attention weights. We notice that column head 1 focuses on human heads, while column head 4 correlates with the field. Row head 6 focuses more on local regions whereas column head 5 pools all over the whole image



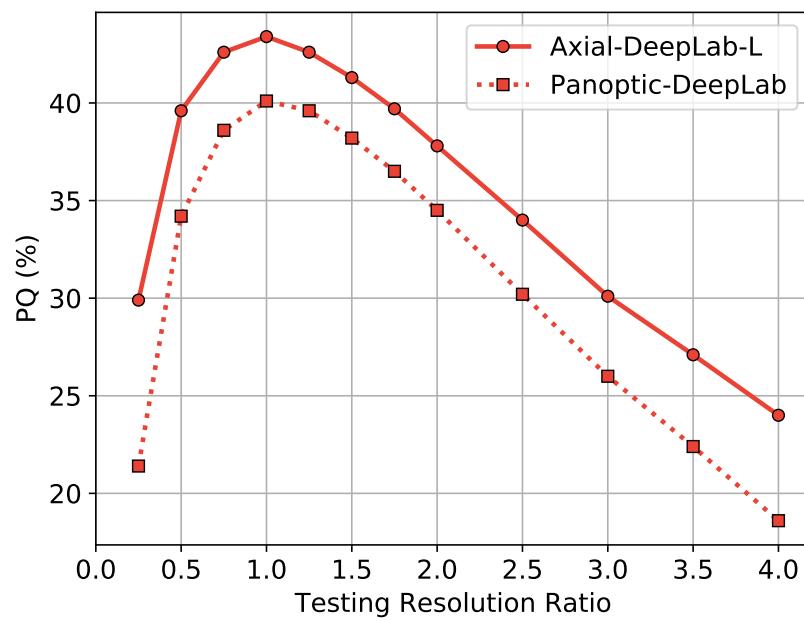
**Figure B.4.** Attention maps in block 3 of stage 4. They focus more on long range context than those in Figure B.3, although all of them have a global receptive field



**Figure B.5.** Training loss on COCO. Equipped with position-sensitive axial-attention, our Axial-DeepLab fits data distribution better than Panoptic-DeepLab [78], especially on the task of predicting the offset to the object center, which requires precise and long range positional information

**Table B.3.** ImageNet validation set results. **Width:** the width multiplier that scales the models up. **Full:** Stand-alone self-attention models without spatial convolutions

Method	Width	Full	Params	M-Adds	Top-1
Conv-Stem + PS-Attention	0.5		5.1M	1.2B	75.5
Conv-Stem + PS-Attention	0.75		10.5M	2.3B	77.4
Conv-Stem + PS-Attention	1.0		18.0M	3.7B	78.1
Conv-Stem + PS-Attention	1.25		27.5M	5.6B	78.5
Conv-Stem + PS-Attention	1.5		39.0M	7.8B	79.0
Conv-Stem + Axial-Attention	0.375		7.4M	1.8B	76.4
Conv-Stem + Axial-Attention	0.5		12.4M	2.8B	77.5
Conv-Stem + Axial-Attention	0.75		26.4M	5.7B	78.6
Conv-Stem + Axial-Attention	1.0		45.6M	9.6B	79.0
Full Axial-Attention	0.5	✓	<b>12.5M</b>	<b>3.3B</b>	<b>78.1</b>
Full Axial-Attention	0.75	✓	<b>26.5M</b>	<b>6.8B</b>	<b>79.2</b>
Full Axial-Attention	1.0	✓	<b>45.8M</b>	<b>11.6B</b>	<b>79.3</b>



**Figure B.6.** Scale stress test on COCO val set

# Appendix C

## MaX-DeepLab: End-to-End Panoptic Segmentation with Mask Transformers

### C.1 Panoptic Segmentation Results

Similar to the case study in Figure 4.2, we provide more panoptic segmentation results of our MaX-DeepLab-L and compare them to the state-of-the-art *box-free* method, Axial-DeepLab [2], the state-of-the-art *box-based* method, DetectoRS [138], and the first Detection Transformer, DETR [142] in Figure C.1 and Figure C.2. MaX-DeepLab demonstrates robustness to the challenging cases of similar object bounding boxes and nearby objects with close centers, while other methods make systematic mistakes because of their individual surrogate sub-task design. MaX-DeepLab also shows exceptional mask quality, and performs well in the cases of many small objects. Similar to DETR [142], MaX-DeepLab fails typically when there are too many object masks.

### C.2 Runtime

In Table C.1, we report the end-to-end runtime (i.e., inference time from an input image to final panoptic segmentation) of MaX-DeepLab on a V100 GPU. All results are obtained

---

<sup>1</sup><https://github.com/facebookresearch/detr>

Original Image	MaX-DeepLab-L	Axial-DeepLab [2]	DetecToRS [138]	DETR [142]	Ground Truth
	Mask Transformer 51.1% PQ	Box-Free 43.4% PQ	Box-Based 48.6% PQ	Box Transformer 45.1% PQ	
MaX-DeepLab segments the baby with its occluded leg correctly. DetecToRS and DETR merge the two people into one instance, probably because the two people have similar bounding boxes. In addition, DETR introduces artifacts around the head of the horse.					
MaX-DeepLab correctly segments all the boards, the zebras, and the people. All other methods fail in these challenging cases of similar bounding boxes and nearby object centers.					
MaX-DeepLab generates a high quality mask for the cat, arguably better than the ground truth. Axial-DeepLab predicts cat pixels on the right of the image, as the center of the cat is close to the center of the bike. And DETR misses the cat and introduces artifacts.					
MaX-DeepLab also performs well in the presence of many small instances.					

**Figure C.1.** Comparing MaX-DeepLab with other representative methods on the COCO *val* set. (Colors modified for better visualization).

Method	Backbone	Input Size	Runtime (ms)	PQ [val]	PQ [test]
Fast Regime					
Panoptic-DeepLab [78]	X-71 [127]	$641 \times 641$	74	38.9	38.8
MaX-DeepLab-S	MaX-S	$641 \times 641$	67	46.4	46.7
Slow Regime					
DETR [142]	RN-101	$1333 \times 800$	128 <sup>1</sup>	45.1	46.0
Panoptic-DeepLab [78]	X-71 [127]	$1025 \times 1025$	132	39.7	39.6
MaX-DeepLab-S	MaX-S	$1025 \times 1025$	131	48.4	49.0

**Table C.1.** End-to-end runtime. **PQ [val]:** PQ (%) on COCO val set. **PQ [test]:** PQ (%) on COCO test-dev set.

by (1) a single-scale input without flipping, and (2) built-in TensorFlow library without extra inference optimization. In the fast regime, MaX-DeepLab-S takes 67 ms with a typical  $641 \times 641$  input. This runtime includes 5 ms of postprocessing and 15 ms of batch normalization that can be easily optimized. This fast MaX-DeepLab-S does not only outperform DETR-R101 [142], but is also around 2x faster. In the slow regime, the standard MaX-DeepLab-S takes 131 ms with a  $1025 \times 1025$  input, similar to Panoptic-DeepLab-X71 [78]. This runtime is also similar to our run of the official DETR-R101 which takes 128 ms on a V100, including 63 ms for box detection and 65 ms for the heavy mask decoding.

### C.3 Mask Output Slot Analysis

In this section, we analyze the statistics of all  $N = 128$  mask prediction slots using MaX-DeepLab-L. In Figure C.3, we visualize the joint distribution of mask slot firings and the classes they predict. We observe that the mask slots have imbalanced numbers of predictions and they specialize on ‘thing’ classes and ‘stuff’ classes. Similar to this Mask-Class joint distribution, we visualize the Mask-Pixel joint distribution by extracting an average mask for each mask slot, as shown in Figure C.4. Specifically, we resize all COCO [15] validation set panoptic segmentation results to a unit square and take an

average of masks that are predicted by each mask slot. We split all mask slots into three categories according to their total firings and visualize mask slots in each category. We observe that besides the class-level specialization, our mask slots also specialize on certain regions of an input image. This observation is similar to DETR [142], but we do not see the pattern that almost all slots have a mode of predicting large image-wide masks.

## C.4 Mask Head Visualization

In Figure 4.5, we visualize how the mask head works by training a MaX-DeepLab with only  $D = 3$  decoder feature channels (for visualization purpose only). Although this extreme setting degrades the performance from 45.7% PQ to 37.8% PQ, it enables us to directly visualize the decoder features as RGB colors. Here in Figure C.5 we show more examples using this model, together with the corresponding panoptic sementation results. We see a similar clustering effect of instance colors, which enables our simple mask extraction with just a matrix multiplication (a.k.a. dynamic convolution [165]–[168]).

## C.5 Transformer Attention Visualization

We also visualize the  $M2P$  attention that connects the transformer to the CNN. Specifically, given an input image from COCO validation set, we first select four output masks of interest from the MaX-DeepLab-L panoptic prediction. Then, we probe the attention weights between the four masks and all the pixels, in the last dual-path transformer block. Finally, we colorize the four attention maps with four colors and visualize them in one figure. This process is repeated for two images and all eight attention heads as shown in Figure C.6. We omit our results for the first transformer block since it is mostly flat. This is expected because the memory feature in the first transformer block is unaware of the pixel-path input image at all. Unlike DETR [142] which focuses on object extreme points for detecting bounding boxes, our MaX-DeepLab attends to individual object (or stuff)

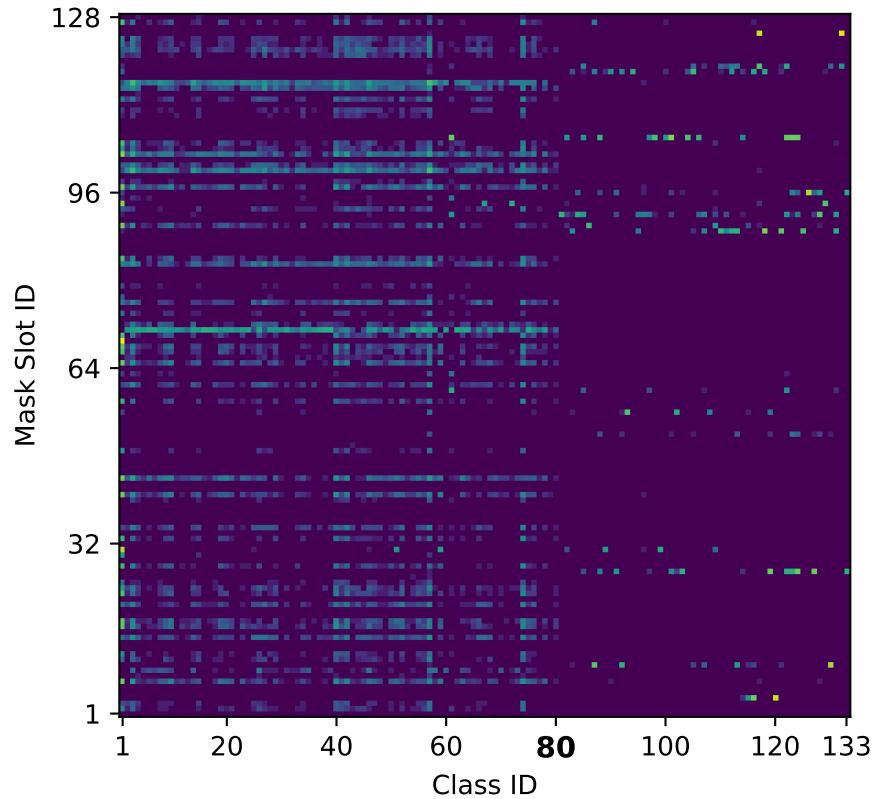
masks. This mask-attending property makes MaX-DeepLab relatively robust to nearby objects with similar bounding boxes or close mass centers.

## C.6 More Technical Details

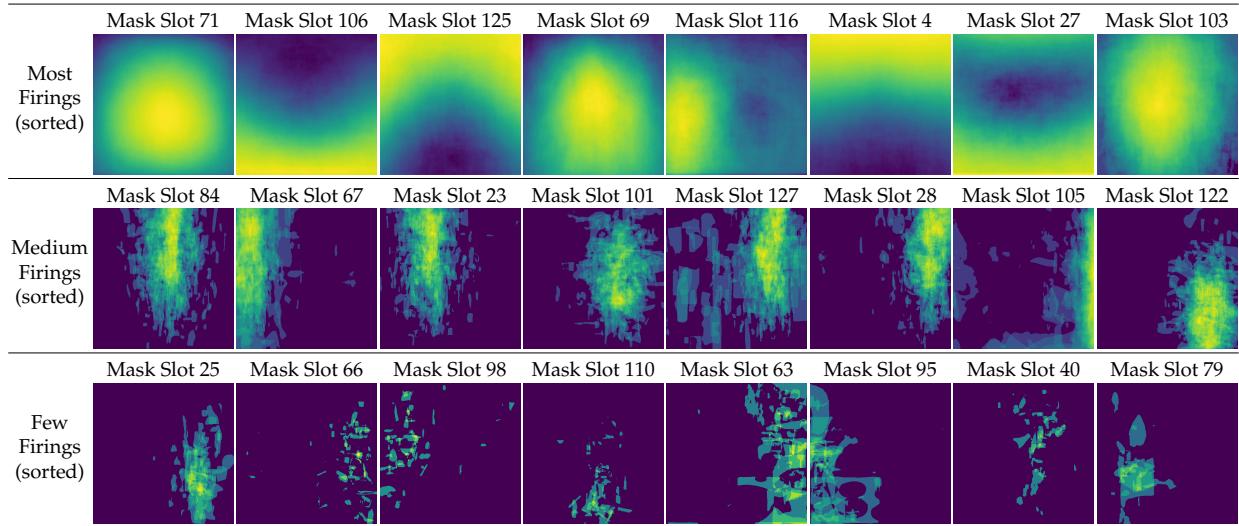
In Figure C.7, Figure C.8, and Figure C.9, we include more details of our MaX-DeepLab architectures. As marked in the figure, we pretrain our model on ImageNet [8]. The pretraining model uses only *P2P* attention (could be a convolutional residual block or an axial-attention block), without the other three types of attention, the feed-forward network (FFN), or the memory. We directly pretrain with an average pooling followed by a linear layer. This pretrained model is used as a backbone for panoptic segmentation, and it uses the backbone learning rate multiplier we mentioned in Section 4.4. After pretraining the CNN path, we apply (with random initialization) our proposed memory path, including the memory, the three types of attention, the FFNs, the decoding layers, and the output heads for panoptic segmentation. In addition, we employ multi-head attention with 8 heads for all attention operations. In MaX-DeepLab-L, we use shortcuts in the stacked decoder. Specifically, each decoding stage (resolution) is connected to the nearest two previous decoding stage outputs of the same resolution.

Original Image	MaX-DeepLab-L	Axial-DeepLab [2]	DetectoRS [138]	DETR [142]	Ground Truth
	Mask Transformer 51.1% PQ	Box-Free 43.4% PQ	Box-Based 48.6% PQ	Box Transformer 45.1% PQ	
					
Similar to DETR [142], MaX-DeepLab fails typically when there are too many masks to segment in an image. This example contains more than 200 masks that should be predicted, mostly people and ties.					
					
In this failure case, MaX-DeepLab mistakes the birds for kites in the sky, probably because the birds are too small.					

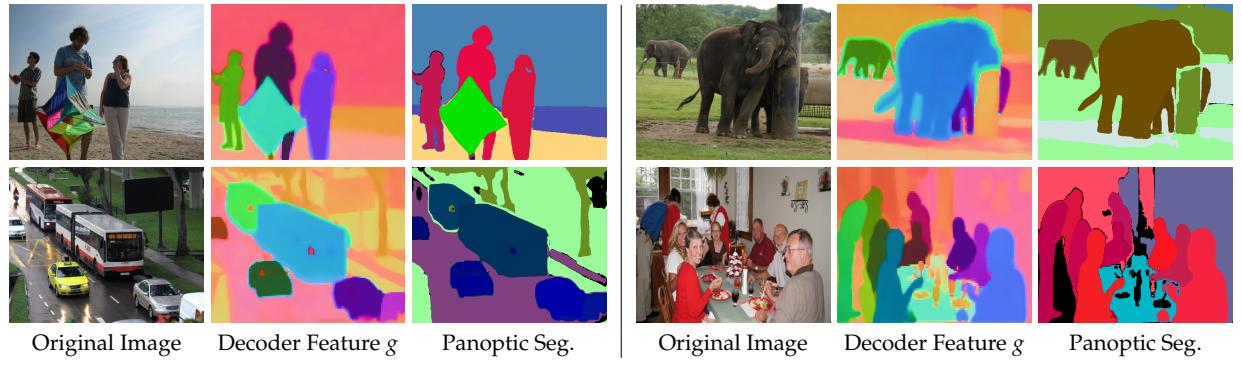
**Figure C.2.** Failure cases of MaX-DeepLab on the COCO *val* set.



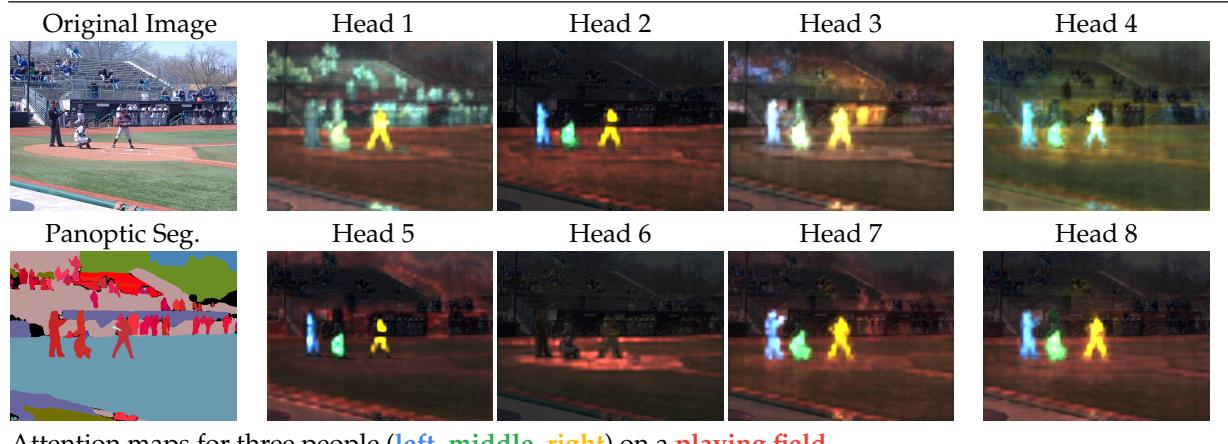
**Figure C.3.** The joint distribution for our  $N = 128$  mask slots and 133 classes with 80 ‘thing’ classes on the left and 53 ‘stuff’ classes on the right. We observe that a few mask slots predict a lot of the masks. Some mask slots are used less frequently, probably only when there are a lot of objects in one image. Some other slots do not fire at all. In addition, we see automatic functional segregation between ‘thing’ mask slots and ‘stuff’ mask slots, with a few exceptions that can predict both thing and stuff masks.



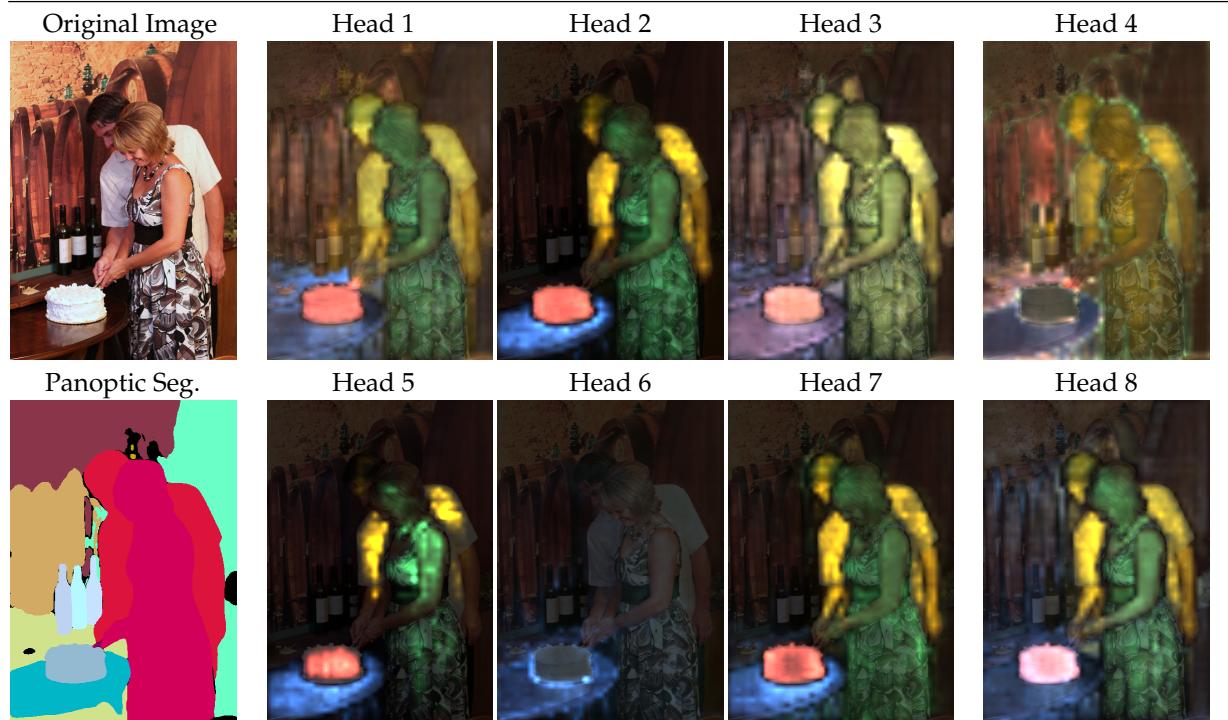
**Figure C.4.** The average masks that each mask slot predicts, normalized by image shape. Mask slots are categorized by their total number of firings and sorted from most firings to few firings. We observe spatial clustered patterns, meaning that the mask slots specialize on certain regions of an input image. For example, the most firing mask slot 71, focusing on the center of an image, predicts almost all 80 ‘thing’ classes but ignores ‘stuff’ classes (Figure C.3). The top three categories are tennis rackets, cats, and dogs. The second firing mask slot 106 segments 14 classes of masks on the bottom of an image, such as road, floor, or dining-tables. The third firing mask slot 125 concentrates 99.9% on walls or trees that are usually on the top of an image. The fourth firing mask slot 69 focuses entirely on the person class and predicts 2663 people in the 5000 validation images.



**Figure C.5.** More visualizations of the decoder feature  $g$  with  $D = 3$ . Similar to Figure 4.5, we observe a clustering effect of instance colors, *i.e.*, pixels of the same instance have similar colors (features) while pixels of different instances have distinct colors. Note that in this extreme case of  $D = 3$  (that achieves 37.8% PQ), there are not enough colors for all masks, which causes missing objects or artifacts at object boundaries, but these artifacts do not present in our normal setting of  $D = 128$  (that achieves 45.7% PQ).

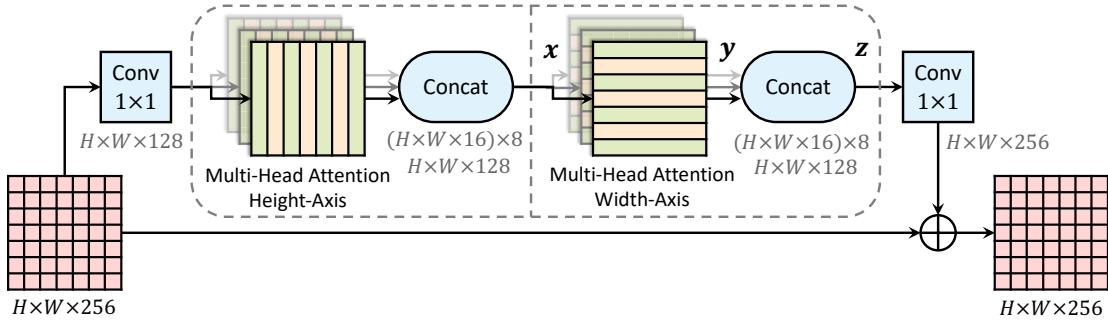


Attention maps for three people (**left, middle, right**) on a **playing field**.

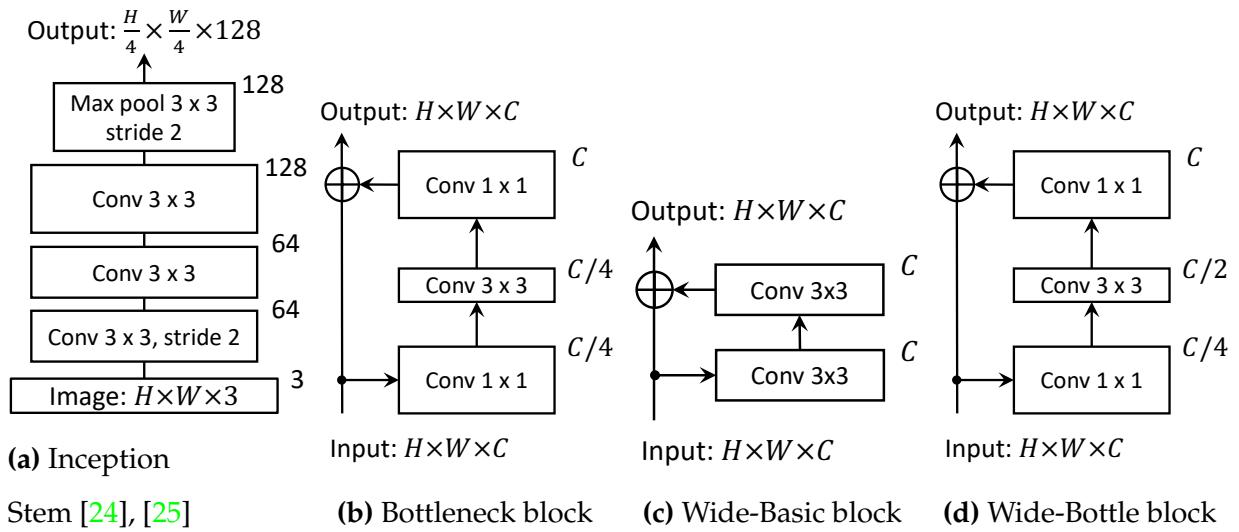


Attention maps for two people (**woman, man**) cutting a **cake** on a **table**.

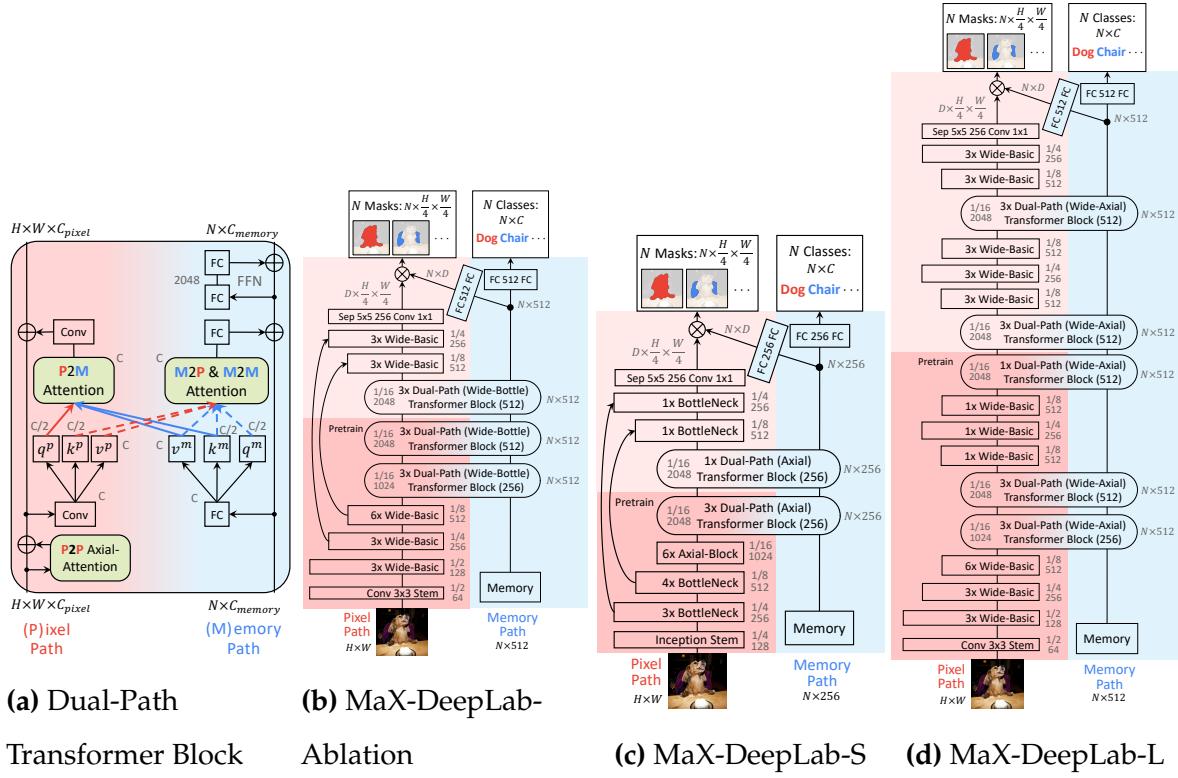
**Figure C.6.** Visualizing the transformer  $M2P$  attention maps for selected predicted masks. We observe that head 2, together with head 5, 7, and 8, mainly attends to the output mask regions. Head 1, 3, and 4 gather more context from broader regions, such as semantically-similar instances (scene 1 head 1) or mask boundaries (scene 2 head 4). In addition, we see that head 6 does not pay much attention to the pixel-path, except for some minor firings on the playing field and on the table. Instead, it focuses more on  $M2M$  self-attention which shares the same softmax with  $M2P$  attention (Equation (4.14)).



**Figure C.7.** An example Axial-Block from Axial-DeepLab [2]. This axial-attention bottleneck block consists of two axial-attention layers operating along height- and width-axis sequentially.



**Figure C.8.** Building blocks for our MaX-DeepLab architectures.



**Figure C.9.** More detailed MaX-DeepLab architectures. **Pretrain** labels where we use a classification head to pretrain our models on ImageNet [8]. **(a)** A dual-path transformer block with  $C$  intermediate bottleneck channels. **(b)** The baseline architecture for our ablation studies in Section 4.4.2. **(c)** MaX-DeepLab-S that matches the number of parameters and M-Adds of DETR-R101-Panoptic [142]. **Axial-Block** (Figure C.7) is an axial-attention bottleneck block borrowed from Axial-DeepLab-L [2]. **(d)** MaX-DeepLab-L that achieves the state-of-the-art performance on COCO [15]. **Wide-Axial** is a wide version of Axial-Block with doubled intermediate bottleneck channels, similar to the one used in Axial-DeepLab-XL [2]. (The residual connections are dropped for neatness).

## Appendix D

# CO2: Consistent Contrast for Unsupervised Visual Representation Learning

### D.1 Implementation Details of Contrastive Pre-Training

We evaluate our approach based on MoCo [173]. MoCo has two different encoders to encode queries and keys respectively. The query encoder is updated with respect to the loss function, while the key encoder is an exponential moving average of the query encoder. The keys are stored in a dynamic memory bank, whose entries are updated at every training step with the current mini-batch enqueued and the oldest mini-batch dequeued. The backbone is a standard ResNet-50 [267], and features after the global average pooling layer are projected to 128-D vectors [169], normalized by  $\ell_2$  norm. The size of the memory bank (i.e., the number of negative samples) is 65,536 and the momentum to update the key encoder is 0.999.  $\tau_{ins}$  is 0.07 for MoCo variants and 0.2 for MoCo v2 variants, which are the default settings of these two methods.

We use momentum SGD with momentum 0.9 and weight decay 1e-4. The batch size is 256 on 4 GPUs. To prevent potential information leak with Batch Normalization (BN) [268], shuffling BN [173] is performed. The model is trained for 200 epochs with the initial

learning rate of 0.03. The learning rate is multiplied by 0.1 after 120 and 160 epochs for MoCo v1, while cosine decayed [269] for MoCo v2. We keep aligned all training details with MoCo except the number of GPUs. This could be problematic since it changes the per-worker minibatch size, which is related to potential information leaks pointed by [173]. However, we do not notice much difference when reproducing MoCo with 4 GPUs. Our reproduced MoCo v2 with 4 GPUs reaches the accuracy of 67.6% on the linear classification protocol, 0.1% higher than 67.5% reported in its paper. For the hyper-parameters of the proposed consistency term, we set  $\tau_{cons}$  as 0.04 and  $\alpha$  as 10 for the MoCo v1-based CO2, and  $\tau_{con}$  as 0.05,  $\alpha$  as 0.3 for the MoCo v2-based variant.

## D.2 Implementation Details of Downstream Tasks

**Linear Classification.** We freeze the backbone network including the batch normalization parameters, and train a linear classifier consisting of a fully-connected layer followed by softmax on the 2048-D features following the global average pooling layer. We train for 100 epochs. The learning rate is initialized as 15 and decayed by 0.1 every 20 epoch after the first 60 epochs. We set weight decay as 0 and momentum as 0.9. Only random cropping with random horizontal flipping is used as data augmentation.

**Semi-Supervised Learning.** We finetune the pre-trained model for 20 epochs with learning rate starting from 0.01 for the base model and 1.0 for the randomly initialized classification head, decayed by 0.2 after 12 and 16 epochs. Momentum is set to 0.9. Weight decay is 5e-4 for MoCo v1 and 1e-4 for MoCo v2. Only random cropping with random horizontal flipping is used as data augmentation.

**Classification on PASCAL VOC.** Following the evaluation setup in [209], we train a linearSVM [210] on the frozen 2048-D features extracted after the global average pooling layer. The models are trained on `trainval2007` split and tested on `test2007`. The hyper-parameters are selected based on a held-out subset of the training set.

**Detection on PASCAL VOC.** Following the detection benchmark set up in [173], we use FasterR-CNN [139] object detector and ResNet-50 C4 [211] backbone, implemented in Detectron2 [270]. We finetune all the layers including the batch normalization parameters for 24k iterations on the trainval07+12 split and test on test2007 set. The hyper-parameters are the same as the counterpart with supervised ImageNet initialization and MoCo. To calibrate the small feature magnitude due to the output normalization in the unsupervised pre-training stage, two extra batch normalization layers are introduced, one is followed by the regional proposal head whose gradients are divided by 10 and the other is followed by the box prediction head.

**Segmentation on PASCAL VOC.** Following the setup in [173], an FCN-based [271] architecture with atrous convolutions [272] is used and ResNet-50 is the backbone. The training set is train\_aug2012 [273] and the testing set is val2012. Initialized with CO2 models, we finetune all layers for 50 epochs ( 33k iterations) with batch size 16, initial learning rate 0.003, weight decay 1e-4 and momentum 0.9.

# Appendix E

## iBOT: Image BERT Pre-Training with Online Tokenizer

### E.1 Pseudocode

Algorithm 1 shows the PyTorch-like pseudocode for iBOT w/o multi-crop augmentation.

### E.2 Multi-Crop

The advanced performance of several recent state-of-the-art methods [204], [230] relies on multi-crop augmentation, as well as iBOT. In our early experiments, we find the direct usage of multi-crop augmentation leads to instability issues that degrade accuracy. We reveal that these results can be attributed to the distribution mismatch between masked images and non-masked images and can be resolved by minimal changes in iBOT framework.

**Stability of MIM Pre-Trained with Multi-Crop.** We first showcase several practices where training instability occurs, shown in Figure E.1. To reveal the instability, we monitor the NMI curves during training for each epoch as shown in Figure E.2. The most intuitive ideas are to compute as (b) or (c). In (b), MIM is only performed on global crops. This pipeline is unstable during training, and we observe a dip in the NMI training curve. We hypothesize that it can be caused by the distribution mismatch of masked global crops

---

**Algorithm 1:** iBOT PyTorch-like Pseudocode w/o multi-crop augmentation

---

**Input:**

```

 $g_s, g_t;$  // student and teacher network
 $C, C';$  // center on [CLS] token and patch tokens
 $\tau_s, \tau_t;$  // temperature on [CLS] token for student and teacher network
 $\tau'_s, \tau'_t;$  // temperature on patch tokens for student and teacher network
 $l;$  // momentum rate for network
 $m, m';$  // momentum rates for center on [CLS] token and patch tokens

```

$g_t.\text{params} = g_s.\text{params}$

**for**  $x$  in loader **do**

```

 $u, v = \text{augment}(x), \text{augment}(x);$  // random views

```

```

 $\hat{u}, m_u = \text{blockwise\_mask}(u);$  // random block-wise masking
 $\hat{v}, m_v = \text{blockwise\_mask}(v);$  // random block-wise masking

```

```

 $\hat{u}_s^{[\text{CLS}]}, \hat{u}_s^{\text{patch}} = g_s(\hat{u}, \text{return\_all\_tok=true});$  // [n, K], [n, S2, K]
 $\hat{v}_s^{[\text{CLS}]}, \hat{v}_s^{\text{patch}} = g_s(\hat{v}, \text{return\_all\_tok=true});$  // [n, K], [n, S2, K]

```

```

 $u_t^{[\text{CLS}]}, u_t^{\text{patch}} = g_t(u, \text{return\_all\_tok=true});$  // [n, K], [n, S2, K]
 $v_t^{[\text{CLS}]}, v_t^{\text{patch}} = g_t(v, \text{return\_all\_tok=true});$  // [n, K], [n, S2, K]

```

```

 $\mathcal{L}_{[\text{CLS}]} = H(\hat{u}_s^{[\text{CLS}]}, v_t^{[\text{CLS}]}, C, \tau_s, \tau_t) / 2 + H(\hat{v}_s^{[\text{CLS}]}, u_t^{[\text{CLS}]}, C, \tau_s, \tau_t) / 2$ 

```

```

 $\mathcal{L}_{\text{MIM}} = (m_u \cdot H(\hat{u}_s^{\text{patch}}, u_t^{\text{patch}}, C', \tau'_s, \tau'_t).sum(dim=1) / m_u.sum(dim=1)) / 2$ 
 $+ (m_v \cdot H(\hat{v}_s^{\text{patch}}, v_t^{\text{patch}}, C', \tau'_s, \tau'_t).sum(dim=1) / m_v.sum(dim=1)) / 2$ 

```

```

( $\mathcal{L}_{[\text{CLS}]}$ .mean() +  $\mathcal{L}_{\text{MIM}}$ .mean()).backward()

```

```

update( $g_s$ ); // student, teacher and center update

```

```

 $g_t.\text{params} = l \cdot g_t.\text{params} + (1 - l) \cdot g_s.\text{params}$ 

```

```

 $C = m \cdot C + (1 - m) \cdot \text{cat}([u_t^{[\text{CLS}]}, v_t^{[\text{CLS}]}]).\text{mean(dim=0)}$ 

```

```

 $C' = m' \cdot C' + (1 - m') \cdot \text{cat}([u_t^{\text{patch}}, v_t^{\text{patch}}]).\text{mean(dim=(0, 1))}$ 

```

**end**

**def**  $H(s, t, c, \tau_s, \tau_t):$

```

 $t = t.\text{detach}();$  // stop gradient

```

```

 $s = \text{softmax}(s / \tau_s, \text{dim}=1)$ 

```

```

 $t = \text{softmax}((t - c) / \tau_t, \text{dim}=1);$  // center + sharpen

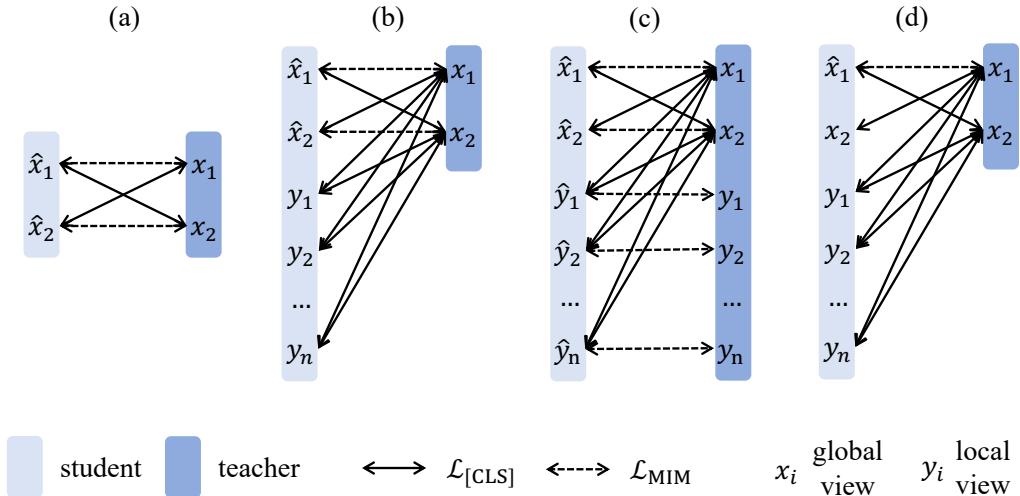
```

```

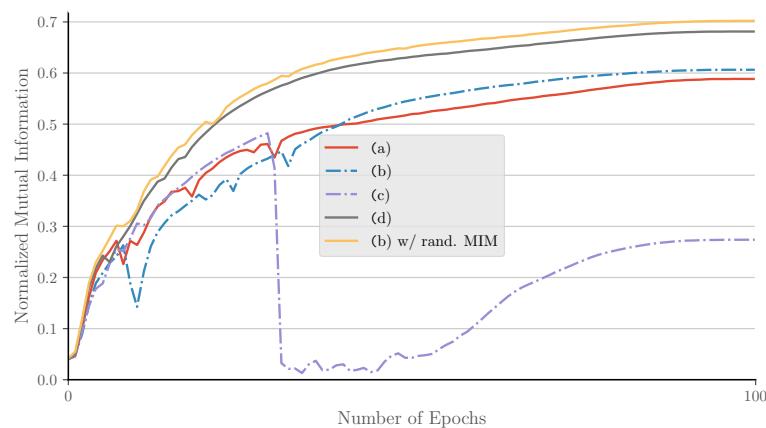
return  $-(t \cdot \log(s)).\text{sum(dim=-1)}$ ;

```

---



**Figure E.1. Computation pipelines for iBOT with or without multi-crop augmentation.**  
 (a) iBOT w/o multi-crop augmentation. (b), (c), and (d) are three pipelines w/ multi-crop augmentation. (b) does not perform MIM for local crops, whereas (c) performs MIM for all crops. (d) only performs MIM for one of the two global crops. iBOT uses (b) with random MIM.



**Figure E.2.** Training curves of different multi-crop strategy.

**Table E.1.**  $k$ -NN performance of iBOT variants.

ViT-S/16, 100 epochs	(a)	(b)	(c)	(d)	(e)	(b) w/ rand. MIM
$k$ -NN	62.1	62.0	31.9	69.8	56.6	71.5

and non-masked local crops. To alleviate this, a straightforward solution is to also perform MIM on local crops with an extra computation cost as (c). However, we do not observe this circumvents training instability. We hypothesize that the regions corresponding to patch tokens of the local crops are small in size, in which there exist few meaningful contents to predict. This hypothesis can be supported by the experiments that when we set the local crop scale in (c) from  $(0.05, 0.4)$  to  $(0.2, 0.4)$ , denoted as (e), the performance drop is mitigated (Table E.1).

**Stabilizing the Training with Non-Masked Global Crops.** Another solution to alleviate the distribution mismatch between masked global crops and non-masked local crops is to train with non-masked global crops, as shown in (d). In other words, we perform *random MIM* when training ViT with multi-crop augmentation. This computation pipeline is stable and achieves a substantial performance gain. In practice, to include non-masked global crops in training, we use (b) and randomly choose a prediction ratio between  $[0, r (r > 0)]$  for each image. When the ratio 0 is chosen, the whole framework excludes MIM and can be seen as DINO. When the ratio  $r (r > 0)$  is chosen, MIM is performed for both of the two global crops. We observe the latter practice performs slightly better since it is more flexible in task composition and data in a batch is mutually independent.

**Range of Scales in Multi-Crop.** In Table E.2, we further study the performance with different local and global scale. Following DINO [230], we conduct the experiments by tweaking  $s$ , where  $s$  is the scale dividing the local and global crops. The local crops are sampled from  $(0.05, s)$  and the global crops are sampled from  $(s, 1)$ .

**Table E.2. Varying multi-crop scale  $s$ .**

ViT-S/16, 300 epochs	0.25	0.4	0.32	ViT-B/16, 50 epochs	0.25	0.4	0.32
$k$ -NN	74	74.3	74.6	$k$ -NN	70	70.1	70.4

We empirically find that  $s = 0.32$  yields optimal performance for both small-size and base-size models. Therefore, we use an  $s$  of 0.32 by default.

**Table E.3.** *k*-NN and linear probing accuracy on ImageNet-1K without multi-crop augmentation (left) and with multi-crop augmentation (right) multi-crop augmentation. We split the table into results without or with multi-crop augmentation.

Method	Arch	Param.	Epo.	<i>k</i> -NN	Linear	Method	Arch	Param.	Epo.	<i>k</i> -NN	Linear
MoCov3	RN50	23	800	-	74.6	SwAV	RN50	23	800	65.7	75.3
	ViT-S/16	21	600	-	73.4		ViT-S/16	21	800	66.3	73.5
	ViT-B/16	85	600	-	76.7		RN50	23	800	67.5	75.3
DINO	ViT-S/16	21	800	70.0	73.7	DINO	ViT-S/16	21	800	74.5	77.0
	ViT-B/16	85	400	68.9	72.8		ViT-B/16	85	400	76.1	78.2
iBOT	ViT-S/16	21	800	72.4	76.2	iBOT	ViT-S/16	21	800	75.2	77.9
	ViT-B/16	85	400	71.2	76.0		ViT-B/16	85	400	76.8	79.4

**State-of-the-Art Comparison w/o and w/ Multi-Crop.** Including iBOT, several recent state-of-the-art works [204], [230] rely heavily on multi-crop augmentation during pre-training. Except for several specific self-supervised methods [206], multi-crop works well on most of the self-supervised methods and consistently yields performance gain [230]. While a more fair comparison with our methods without multi-crop augmentation can be conducted, we believe it is a unique strength of iBOT to work well with multi-crop. In Table E.3, we categorize the state-of-the-art comparison into two parts where one for methods without multi-crop and the other with multi-crop. For the former, we mainly compare our method without multi-crop with MoCov3 [229] and DINO without multi-crop. We observe that our method achieves state-of-the-art performance with ViT-S/16 even without multi-crop and comparable performance with ViT-B/16 compared with MoCov3. For the latter, we mainly compare our method with SwAV [204] and DINO with multi-crop augmentation. We observe that iBOT achieves higher performance with 79.4% of linear probing accuracy when using ViT-S/16.

**Effective Training Epochs.** Due to extra computation costs brought by multi-crop augmentation, different methods with the same pre-training epochs actually see different total numbers of images. To mitigate, we propose to measure the effective training epochs, defined as actual pre-training epochs multiplied with a scaling factor accounting for extra

trained images of different resolutions induced by multi-crop augmentation. DINO and iBOT are by default trained with 2 global crops of size  $224 \times 224$  and 10 local crops of size  $96 \times 96$ . Thus  $r = 2 + (\frac{96}{224})^2 \times 10 = 3.84 \approx 4$  for DINO and iBOT.  $r \approx 3$  for SwAV or DINO with RN50 as the backbone and pre-trained with 2 global crops and 6 local crops.  $r = 2$  for contrastive methods without multi-crop augmentation (*e.g.*, MoCo, SimCLR, BYOL, *etc.*) and  $r = 1$  for non-contrastive methods (*e.g.*, BEiT, Jigsaw, *etc.*).

### E.3 Additional Implementations

**Table E.4. Different fine-tuning recipes.**

LD: layerwise learning rate decay. DS: mixed-precision training with DeepSpeed.

Epo.	LD	DS	BEiT	DINO	iBOT
<i>ViT-S/16</i>					
1	300	1.0	<b>X</b>	81.5	81.1
2	300	0.75	<b>✓</b>	<b>81.7</b>	82.0
3	200	0.65	<b>X</b>	80.7	-
4	200	0.75	<b>X</b>	81.4	81.9
5	200	0.75	<b>✓</b>	81.4	<b>82.3</b>
6	200	0.85	<b>X</b>	81.2	-
<i>ViT-B/16</i>					
7	300	1.0	<b>X</b>	82.1	82.8
8	200	0.65	<b>✓</b>	82.7	83.1
9	100	0.65	<b>X</b>	<b>83.4</b>	83.5
10	100	0.65	<b>✓</b>	83.2	<b>84.0</b>

**Table E.5. Evaluation protocols for semi-supervised learning.**

*Proj.*: fine-tuning from the middle layer of the projection head. LR: logistic regression.

Method	<i>Proj.</i>	1%	10%
<i>frozen features</i>			
1 DINO + k-NN	-	61.3	69.1
2 iBOT + k-NN	-	62.3	70.1
3 DINO + Lin.	-	60.5	71.0
4 iBOT + Lin.	-	62.5	72.2
5 DINO + LR	-	64.5	72.2
6 iBOT + LR	-	<b>65.9</b>	73.4
<i>end-to-end fine-tuning</i>			
7 DINO	<b>X</b>	50.6	73.2
8 iBOT	<b>X</b>	55.0	74.0
9 DINO	<b>✓</b>	60.3	74.3
10 iBOT	<b>✓</b>	61.9	<b>75.1</b>

**Fine-Tuning Recipes of Classification on ImageNet-1K.** By default, we follow the fine-tuning protocol in BEiT [234] to use a layer-wise learning rate decay, weight decay and AdamW optimizer and train small-, base-size models with 200, 100, and 50 epochs respectively. We sweep over four learning rates  $\{8e^{-4}, 9e^{-4}, 1e^{-3}, 2e^{-3}\}$ . Comparatively,

traditional fine-tuning recipe is to fine-tune the network for 300 epochs with a learning rate  $5e^{-4}$ , no weight decay, and SGD optimizer [227] (row 1 versus 8). For a fair comparison, we compare the impact of different fine-tuning recipes with different methods, shown in Table E.4. We empirically find that fine-tuning protocol used in BEiT consistently yields better fine-tuning results and greatly reduces the training epochs. By default, we use a layerwise decay of 0.75 with a training epoch of 200 for ViT-S/16, a layerwise decay of 0.65 with a training epoch of 100 for ViT-B/16, and a layerwise decay of 0.75 with a training epoch of 50 for ViT-L/16. We report the higher results between using or not using DS since we find it brings different impacts to different methods.

**Evaluation Protocols of Semi-Supervised Learning on ImageNet-1K.** We study the impact of different evaluation protocols for semi-supervised learning. Under conventional semi-supervised evaluation protocol, pre-trained models are end-to-end fine-tuned with a linear classification head. SimCLRV2 [241] found that keeping the first layer of the projection head can improve accuracy, especially under the low-shot setting. We fine-tune the pre-trained model from the first layer of the projection head and verify this conclusion holds true for Vision Transformers. We empirically find that Vision Transformer performs better with a frozen backbone with 1% of training data (62.5% in row 4 versus 61.9 % in row 7). In DINO, a logistic regressor built upon the frozen features is found to perform better compared with the multi-class linear classifier upon the frozen features, especially with 1% data (65.9% in row 6 versus 62.5% in row 4). When using 10% data, we empirically find that *end-to-end fine-tuning* from the first layer of the projection layer yields the best performance (75.1% in row 10 versus 73.4% in row 6).

**Fine-Tuning Recipes of Object Detection and Instance Segmentation on COCO.** For both small- and base-size models, we utilize multi-scale training (resizing image with shorter size between 480 and 800 while the longer side no larger than 1333), a learning rate  $1e^{-4}$ , a weight decay of 0.05, and fine-tune the entire network for  $1 \times$  schedule (12 epochs

with the learning rate decayed by  $10 \times$  at epochs 9 and 11). We sweep a layer decay rate of  $\{0.65, 0.75, 0.8, 0.9\}$ . Note that a layer decay rate of 1.0 denotes no layer is decayed. To produce hierarchical feature maps, we use the features output from layer 4, 6, 8, and 12, with 2 deconvolutions, 1 deconvolution, identity mapping, and max-pooling appended after, respectively. We do not use multi-scale testing.

**Fine-Tuning Recipes of Semantic Segmentation on ADE20K.** For semantic segmentation, we follow the configurations in BEiT [234], fine-tuning 160k iterations with  $512 \times 512$  images and a layer decay rate of 0.65. We do not use multi-scale training and testing. We sweep the learning rate  $\{3e^{-5}, 8e^{-5}, 1e^{-4}, 3e^{-4}, 8e^{-4}\}$ . Similar to object detection and instance segmentation, to produce hierarchical feature maps, we add additional deconvolution layers after ViT. As shown in Table E.6, when using linear (Lin.) as the task layer, we

**Table E.6. Linear probing on ADE20K semantic segmentation with and without the last LayerNorm [LN].**

DINO, w/o [LN]	DINO, w/ [LN]	iBOT, w/o [LN]	iBOT, w/ [LN]
33.7	34.5	37.8	<b>38.3</b>

find that appending the last LayerNorm [LN] for [CLS] token to each patch tokens before the decoder consistently yields better performance, while we do not spot the substantial gain when with UperNet as the task layer. By default, we report the segmentation result with [LN] for both linear head for UperNet head.

**Part-Wise Linear Probing.** We use the average of the last-layer self-attention map with [CLS] as the query from multiple heads to rank all the patch tokens. We remove the extra LayerNorm (LN) after the final block following MoCov3 [229].

## E.4 Additional Results

In this section, we provide detailed results for dense downstream tasks, *i.e.*, object detection, instance segmentation, and semantic segmentation. We give the complete figures for occlusion robustness analysis. We also provide extra experiments of nearest neighbor retrieval, robustness analysis against occlusion and shuffle.

**Table E.7. Additional object detection, instance segmentation, and semantic segmentation results with small-size models.** We pre-train iBOT with ViT-S/16 for 800 epochs.

Method	Arch.	Param.	Cascade Mask R-CNN						UperNet Seg.	
			AP <sup>b</sup>	AP <sup>b</sup> <sub>50</sub>	AP <sup>b</sup> <sub>75</sub>	AP <sup>m</sup>	AP <sup>m</sup> <sub>50</sub>	AP <sup>m</sup> <sub>75</sub>	mIoU	mAcc
Sup.	Swin-T	29	48.1	67.1	52.5	41.7	64.4	45.0	44.5	-
MoBY	Swin-T	29	48.1	67.1	52.1	41.5	64.0	44.7	44.1	-
Sup.	ViT-S/16	21	46.2	65.9	49.6	40.1	62.9	42.8	44.5	55.5
iBOT	ViT-S/16	21	<b>49.4</b>	<b>68.7</b>	<b>53.3</b>	<b>42.6</b>	<b>65.6</b>	<b>45.8</b>	<b>45.4</b>	<b>56.2</b>

**Table E.8. Additional object detection, instance segmentation, and semantic segmentation results with base-size models.** We pre-train iBOT with ViT-B/16 for 400 epochs.

Method	Cascade Mask R-CNN						Linear Seg.		UperNet Seg.	
	AP <sup>b</sup>	AP <sup>b</sup> <sub>50</sub>	AP <sup>b</sup> <sub>75</sub>	AP <sup>m</sup>	AP <sup>m</sup> <sub>50</sub>	AP <sup>m</sup> <sub>75</sub>	mIoU	mAcc	mIoU	mAcc
Sup.	49.8	69.6	53.8	43.2	66.6	46.5	35.4	44.6	46.6	57.0
BEiT	50.1	68.5	54.6	43.5	66.2	47.1	27.4	35.5	45.8	55.9
DINO	50.1	69.5	54.3	43.4	66.8	47.0	34.5	43.7	46.8	57.1
iBOT	<b>51.2</b>	<b>70.8</b>	<b>55.5</b>	<b>44.2</b>	<b>67.8</b>	<b>47.7</b>	<b>38.3</b>	<b>48.0</b>	<b>50.0</b>	<b>60.3</b>

**Object Detection, Instance Segmentation, and Semantic Segmentation.** We here provide more detailed results on object detection, instance segmentation, and semantic segmentation with small- and base-size models, shown in Table E.7 and Table E.8 respectively. Specifically, we include AP<sup>b</sup><sub>50</sub> and AP<sup>b</sup><sub>75</sub> for object detection, AP<sup>m</sup><sub>50</sub> and AP<sup>m</sup><sub>75</sub> for instance segmentation, mAcc for semantic segmentation. For object detection (Det.) and instance segmentation (Inst. Seg.), we consider Cascade Mask R-CNN as the task layer. For seman-

tic segmentation (Seg.), we consider two evaluation settings where a linear head (Lin.) and UPerNet are taken as the task layer.

**Table E.9.  $k$ -NN and linear probing on ImageNet-1K with different pre-training datasets.**

Arch.	Pre-Train Data	Param.	Epoch	$k$ -NN	Linear
ViT-S/16	ImageNet-1K	21	800	75.2	77.9
ViT-S/16	ImageNet-22K	21	160	69.3	76.5
ViT-B/16	ImageNet-1K	85	400	<b>77.1</b>	79.5
ViT-B/16	ImageNet-22K	85	80	71.1	79.0
ViT-L/16	ImageNet-1K	307	300	78.0	81.0
ViT-L/16	ImageNet-22K	307	50	70.6	<b>81.7</b>

**$k$ -NN and Linear Probing with ImageNet-22K.** We further report  $k$ -NN and linear probing accuracy on ImageNet-1K with models pre-trained on ImageNet-22K dataset. We empirically observe that ImageNet-1K pre-training incurs better ImageNet-1K  $k$ -NN and linear probing performance, which is opposite to the fine-tuning performance observed in Table 6.2 and Table 6.3. We hypothesize that the data distribution plays a more crucial rule under evaluation protocols based on frozen features, such that models pre-trained with smaller ImageNet-1K dataset consistently achieve better results.

**Table E.10. Effectiveness of pre-trained features on nearest neighbor retrieval.** We report the results on different downstream tasks whose evaluation is based on nearest neighbor retrieval.

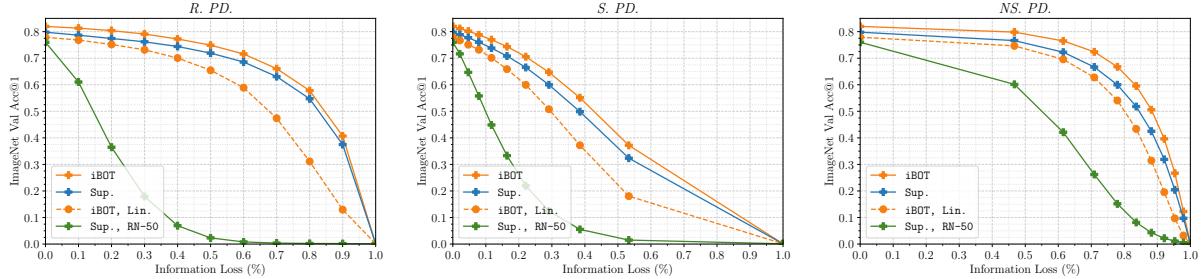
Method	Image Retrieval				Vid. Obj. Segment.		
	$\mathcal{ROx}$		$\mathcal{RPar}$				
	M	H	M	H	$(\mathcal{J} \& \mathcal{F})_m$	$\mathcal{J}_m$	$\mathcal{F}_m$
DINO	<b>37.2</b>	<b>13.9</b>	<b>63.1</b>	<b>34.4</b>	61.8	60.2	<b>63.4</b>
iBOT	36.6	13.0	61.5	34.1	<b>61.8</b>	<b>60.4</b>	63.2

**Nearest Neighbor Retrieval.** Nearest neighbor retrieval is considered using the frozen pre-trained features following the evaluation protocol as in DINO [230]. DINO has demonstrated the strong potential of pre-trained ViT features to be directly used for retrieval. To

validate, DINO designed several downstream tasks, including image retrieval and video object segmentation, where video object segmentation can be seen as a dense retrieval task by finding the nearest neighbor between consecutive frames to propagate masks. We compare iBOT with DINO on these benchmarks with the same evaluation settings. As demonstrated in Table E.10, iBOT has comparable results with DINO. While iBOT has higher  $k$ -NN results on Imagenet-1K, the performance is not better for iBOT in image retrieval. We empirically find that the results on image retrieval are sensitive to image resolution, multi-scale features, *etc.*, and the performance varies using pre-trained models with minimal differences on hyper-parameter setup. For this reason, we do not further push iBOT for better results.

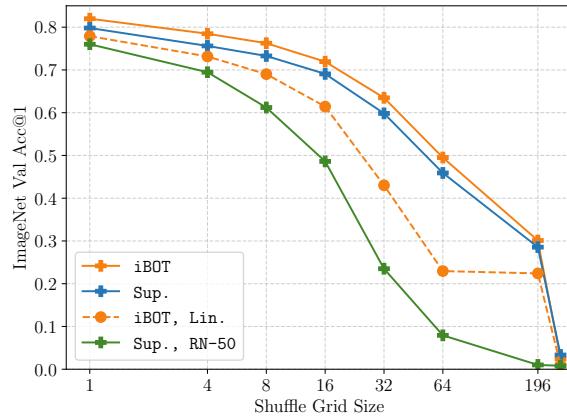
**Robustness against Background Change.** Deep models rely on both foreground objects and backgrounds. Robust models should be tolerant to background changes and able to locate discriminative foreground parts. We evaluate this property on ImageNet-9 (IN-9) dataset [274]. IN-9 includes 9 coarse-grained classes and 7 variants by mixing up the foreground and background from different images. *Only-FG* (*O.F.*), *Mixed-Same* (*M.S.*), *Mixed-Rand* (*M.R.*), and *Mixed-Next* (*M.N.*) are 4 variant datasets where the original foreground is present but the background is modified, whereas *No-FG* (*N.F.*), *Only-BG-B* (*O.BB.*), and *Only-BG-T* (*O.BT.*) are 3 variants where the foreground is masked. As shown in Table 6.8, we observe a performance gain except for *O.BT.*, indicating iBOT’s robustness against background changes. We note in *O.BT.* neither foreground nor foreground mask is visible, contradicting the pre-training objective of MIM.

**Robustness against Occlusion.** Masked prediction has a natural strength in cases where parts of the image are masked out since the models are trained to predict their original contents. We here provide the detailed results of occlusion with different information loss ratios in Figure E.3 under three dropping settings: random, salient, and non-salient. We showcase the results of iBOT end-to-end fine-tuned or with a linear head over the



**Figure E.3. Robustness against occlusion.** Model’s robustness against occlusion with different information loss ratios is studied. 3 patch dropping settings: Random Patch Dropping (left), Salient Patch Dropping (middle), and Non-Salient Patch Dropping (right) are considered.

pre-trained backbone. We include the results of supervised results with both ViT-S/16 and ResNet-50 for comparison. ViT shows higher robustness compared to its CNN counterpart, *i.e.*, ResNet-50, given that Transformers’ dynamic receptive field makes it less dependent on images’ spatial structure. We empirically find iBOT has stronger robustness against occlusion compared to its supervised baseline, consolidating that MIM helps to model the interaction between the sequence of image patches using self-attention such that discarding proportion of elements does not degrade the performance significantly.



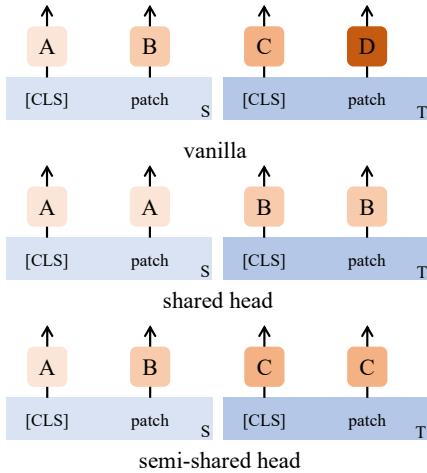
**Figure E.4. Robustness against shuffle.** Model’s robustness against shuffle with different grid shuffle sizes is studied.

**Robustness against Shuffle.** We study the model’s sensitivity to the spatial structure by shuffling on input image patches. Specifically, we shuffle the image patches with different grid sizes following [250]. We showcase the results of iBOT end-to-end fine-tuned or with a linear head over the pre-trained backbone. We include the results of supervised results with both ViT-S/16 and ResNet-50 for comparison. Note that a shuffle grid size of 1 means no shuffle, and a shuffle grid size of 196 means all patch tokens are shuffled. Figure E.4 suggests that iBOT retain accuracy better than its supervised baseline and ResNet-50. It also indicates that iBOT relies less on positional embedding to preserve the global image context for right classification decisions.

## E.5 Additional Ablations

In this section, we study the impact of other parameters that we have conducted experiments on. Without extra illustrations, we use 300-epoch pre-trained ViT-S/16, a prediction ratio  $r = 0.3$  and without multi-crop augmentation for the ablative study.

**Table E.11. Different head sharing strategy.**



**Table E.12. Hard label versus soft label.**

*Cen.:* centering.  $^\dagger$ : smaller temperature for teacher output.

Method	Cen.	Post Proc.	$k$ -NN	Linear
x	softmax	49.8	63.5	
	hardmax	64.8	71.9	
	softmax $^\dagger$	69.4	73.9	
✓	softmax	67.8	72.9	
	hardmax	68.1	73.3	
	softmax $^\dagger$	69.1	74.2	
DINO	-	-	67.9	72.5

**Architecture of Projection Head.** As mentioned earlier, a shared head can transfer the semantics acquired in [CLS] token to patch tokens, slightly improving the performance.

We notice that the head for patch tokens in the student network only see the masked tokens throughout the training, the distribution of which mismatches tokens with natural textures. Therefore, we conduct an experiment using a non-shared head for the student network but a shared head for the teacher network denoted as *semi-shared head*. Their differences are demonstrated in Figure E.11, where  $S$  and  $T$  denotes student and teacher network respectively. The heads with the same index and color denotes they have shared parameters.  $^{\dagger}$  denotes only the first 2 layers out of the 3-layer MLP share the parameters.

**Table E.13. Varying iBOT projection head design.**

Arch.	vanilla	shared $^{\dagger}$	sm. shared	sm. shared $^{\dagger}$	shared
$k$ -NN .	68.9	68.0	68.4	68.4	69.1
Lin.	73.9	73.7	73.7	73.8	74.2

However, we do not observe that semi-shared head is better than shared head. By default, we share the entire projection head for [CLS] token and patch tokens.

**Comparing MIM with Dense Self-Distillation.** To identify the superiority of MIM to model internal structure using over its alternatives, we conduct experiments performing self-distillation on original patch tokens along with the [CLS] token. We consider two matching strategies to construct patch token pairs for self-distillation. Specifically, *pos.*

**Table E.14. Comparing MIM with dense self-distillation.**

Arch.	DINO	DINO + pos.	DINO + feat.	iBOT
$k$ -NN	67.9	67.1 ( $-0.8$ )	68.5 ( $+0.6$ )	69.1 ( $+1.2$ )
Lin.	72.5	72.5 ( $+0.0$ )	73.4 ( $+0.9$ )	74.2 ( $+1.7$ )

denotes matching according to the absolute position of two views. Similar to [275].  $j$  is defined as  $\arg \min_j dist(p_i, p'_j)$ , where  $p$  is the position in the original image space and  $dist(u, v)$  is euclidean distance. The losses are only computed for the overlapped regions of two views. We do not observe substantial gain brought by matching via patches' absolute position. *feat.* denotes matching according to the similarity of the backbone similarity of

two views. Similar to [276], we match for each patch token  $f_i$  the most similar patch token from another view  $f'_j$ , where  $j = \arg \max_j sim(f_i, f'_j)$ .  $sim(u, v)$  is cosine distance. Such practice brings a 0.6% performance gain in terms of linear probing accuracy, which is also observed by a concurrent work, EsViT [240]. Comparatively, iBOT prompts an 1.2% gain on linear probing, verifying the necessity and advancement of MIM.

**Hard Label versus Soft Label.** We study the importance of using a continuous token distribution (softmax $^\dagger$ ) instead of a discretized id (hardmax) when performing MIM. Results in Table E.12 indicate continuous tokenization plays a crucial part. We empirically find the improvement brought by centering, whose roles are less important compared to centering in self-distillation on [CLS] token. Only sharpening can produce a  $k$ -NN accuracy of 69.4 and a linear probing accuracy of 73.9.

**Centering and Sharpening.** Different from the [CLS] token, patch tokens do not have certain semantic cluster and vary more widely from each others. We study the impact of several critical parameters that decide the distillation process and customize them for distillation over the patch tokens. Specifically, the smoothing momentum for online

**Table E.15. Varying the smoothing momentum for online centering  $m'$  and sharpening temperature  $\tau'_t$  for the patch tokens.**

$m'$	.8	.99	.999	.9	.9	.9
$\tau'_t$	.04 → .07	.04 → .07	.04 → .07	.04 → .06	.05 → .08	.04 → .07
$k$ -NN	68.7	68.8	68.9	68.5	68.7	69.1
Lin.	74.0	73.8	73.8	73.5	73.9	74.2

centering  $m'$  and sharpening temperature  $\tau'_t$  are studied. Note we keep the parameters for [CLS] token the same as DINO and only study for parameters for the patch tokens.

**Loss Ratio.** We study the impact of different ratio between  $\mathcal{L}_{[CLS]}$  and  $\mathcal{L}_{\text{MIM}}$ . We keep the base of  $\mathcal{L}_{[CLS]}$  to 1 and scale  $\mathcal{L}_{\text{MIM}}$  with different ratios. We observe that directly adding two losses up without scaling yields the best performance in terms of linear probing

**Table E.16. Varying the loss ratio between  $\mathcal{L}_{[\text{CLS}]}$  and  $\mathcal{L}_{\text{MIM}}$ .**

$\mathcal{L}_{[\text{CLS}]} / \mathcal{L}_{\text{MIM}}$	0.5	2	1
k-NN	68.7	69.4	69.1
Lin.	73.8	74.1	74.2

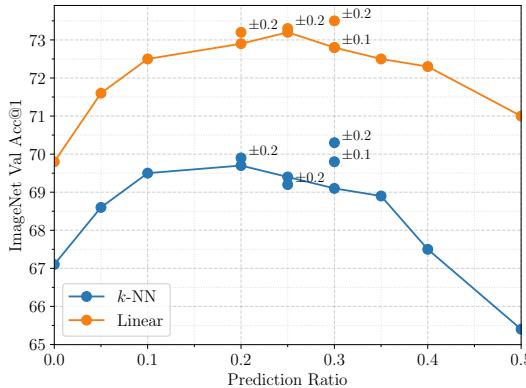
accuracy.

**Output Dimension.** We follow the structure of projection head in DINO with l2-normalized bottleneck and without batch normalization. We study the impact of output dimension  $K$  of the last layer. While our method excludes large output dimensionality since each patch

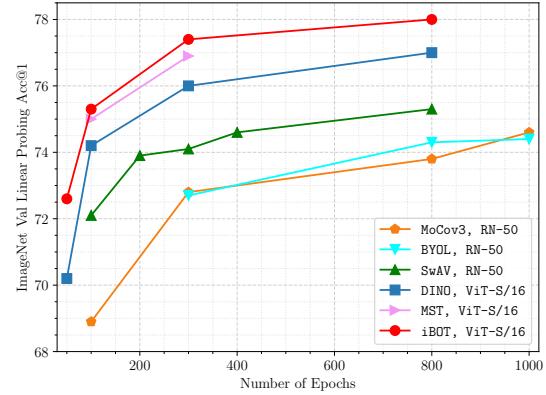
**Table E.17. Varying the projection head output dimension.**

$K$	4096	16384	8192
k-NN	68.3	68.8	69.1
Lin.	74.5	74.0	74.2

token has an output distribution, we do not observe substantial performance gain brought by larger output dimensions. Therefore, we choose  $K = 8192$  by default.



**Figure E.5. Impact of the prediction ratio.**  
 $\pm$  denotes to randomly sample from a region.



**Figure E.6. Impact of the training epochs.**  
Models are ViT-S/16 with multi-crop augmentation.

**Prediction Ratios.** Masked modeling is based on a formulation of partial prediction, the objective of which is to maximize the log-likelihood of the target tokens conditioned on the non-target tokens. We experiment with different prediction ratios for masked image modeling. The results are shown in Figure E.5. We observe that the performance is not sensitive to variant prediction ratios between 0.05 and 0.4. Adding a variance upon the fixed value can also consistently bring a performance gain, which can be explained as stronger data augmentation. The teacher output of non-masked images is now pulled together with the student output of masked images with different ratios. By default, we use 0.3 ( $\pm 0.2$ ) as the prediction ratio. For models with multi-crop augmentation, following the above discussions, we randomly choose a prediction of 0 or 0.3 ( $\pm 0.2$ ) for each image.

**Training Epochs.** We provide the linear probing top-1 accuracy with ViT-S/16 pre-trained for different epochs. For comparison, we also include the accuracy curve of other methods with comparable numbers of parameters, *i.e.*, ResNet-50. From Figure E.6, we observe that longer training for 800 epochs can improve the model’s performance. It’s noteworthy that iBOT can achieve a Top-1 accuracy of SwAV [204] pre-trained with 800 epochs in less than 100 epochs. iBOT pre-trained with 800 epochs brings a 0.9% improvement over previous state-of-the-art method.

**Table E.18. Time and Memory Requirements.** We detail the actual training time (T) and GPU memory (Mem.) of different methods, together with their respective linear probing (Lin.) and fine-tuning (Fin.) accuracy. All methods are trained on two 8-GPU V100 machines with a batch size of 1024.

Method	Number of Crops	T <sub>100</sub>	T <sub>300</sub>	T <sub>800</sub>	Mem.	Lin. <sub>300</sub>	Lin. <sub>800</sub>	Fin. <sub>800</sub>
BEiT	1 × 224 <sup>2</sup>	11.3h	33.7h	90.1h	5.6G	20.7	24.2	81.4
DINO	2 × 224 <sup>2</sup>	15.1h	44.7h	111.6h	9.3G	72.5	73.7	81.6
iBOT	2 × 224 <sup>2</sup>	15.6h	47.0h	126.4h	13.1G	74.8	76.2	82.0
DINO	2 × 224 <sup>2</sup> + 10 × 96 <sup>2</sup>	24.2h	72.6h	180.0h	15.4G	76.2	77.0	82.0
iBOT	2 × 224 <sup>2</sup> + 10 × 96 <sup>2</sup>	24.3h	73.3h	193.4h	19.5G	77.4	77.9	82.3

**Time and Memory Requirements.** BEiT is trained with a non-contrastive objective and without multi-crop augmentation, thus it consumes only a memory of 5.6G and takes 90.1h for 800 epochs. Comparing iBOT and DINO with multi-crop augmentation, iBOT with MIM induces 25% more memory requirements and 7.4% more actual training time. Considering pre-training efficiency (accuracy versus time), 800-epochs pre-trained DINO requiring for 180.0h, while 300-epochs iBOT only requires 73.3h with 0.4% higher linear probing accuracy (77.0 versus 77.4).

## E.6 Alternative Tokenizers

**Table E.19. Methodology comparison over different approaches to tokenize the patches.** We report ImageNet-1K  $k$ -NN, linear and fine-tuning validation accuracy. Models are pre-trained with ViT-S/16 and 300 epochs.

Method	$k$ -NN	Linear	Fine-Tune
Rand.	-	-	79.9
MPP [228]	16.4	37.2	80.8
Patch Clustering	19.2	40.1	81.3
BEiT [234]	6.9	24.2	81.4
Standalone DINO as tokenizer	44.3	60.0	81.7
iBOT	70.3	74.8	81.5

To investigate how different approaches to tokenize the patches affect MIM, we study several alternatives. In BEiT [234], masked patches are tokenized by a DALL-E encoder. MPP [228] tokenizes the masked patches using their 3-bit mean color. For Patch Clustering, we first perform  $K$ -Means algorithm to the flattened color vector of each  $16 \times 16$  patch ( $d = 768$ ). 10% data of ImageNet-1K training set is sampled and clustered. We set  $K$  to 4096. During pre-training, each patch is tokenized by the index of its closest centroids. Lastly, we use 300-epoch pre-trained DINO as a standalone tokenizer. Each patch can be tokenized by the argmax of its output from the pre-trained DINO. We use average pooling to aggregate

the patch representations. From Table E.19, we see that all methods achieve decent fine-tuning results compared to the supervised baseline, while only methods tokenized by semantically meaningful tokenizer have proper results on  $k$ -NN and linear classification. MPP [228] and patch clustering rely purely on offline statistics without the extra stage of online training. We find patch clustering has slightly better performance in all three protocols compared to MPP, suggesting the benefits brought by visual semantics. While BEiT has poor  $k$ -NN and linear probing accuracy, a good fine-tuning result also suggests relatively low requirements for fine-tuning protocol on high-level semantics.

## E.7 Visualization

In this section, we first give more visualized pattern layouts and self-attention maps. Beyond that, we consider an additional task of mining sparse correspondences between two images and illustrating the superiority of ViTs by showcasing several visualized results.

### E.7.1 Pattern Layout

**Pattern Layout for Patch Tokens.** To illustrate versatile, interesting behaviors iBOT has learned, we organize the visualization of pattern layout in two figures. In Figure E.7, we mainly showcase additional pattern layouts that share high-level semantics. In Figure E.8, we mainly showcase additional pattern layouts that share low-level details like color, texture, shape, *etc*. Top 100 patches with the highest confidence over the validation set are visualized with a  $5 \times 5$  context around each  $16 \times 16$  patch token (colored orange).

**Composing Images with Representative Patterns.** In Figure E.9, we visualize 4 patches with the highest self-attention score (with non-overlapped assigned index) and also show the pattern layout of that assigned index. The visualized results indicate iBOT can only be represented by several representative patches, which helps the model’s robustness

and performance in recognition. This is also validated by our part-wise linear probing experiments.

**Comparison with Other Methods.** We visualize pattern layout for patch tokens using other self-supervised methods [230], [234] in Figure E.10. For BEiT, the DALL-E encoder generates a discrete number for each patch token. For DINO, we directly use the projection head for [CLS] token and generate a 65536-d probability distribution for each patch token. The index with the highest probability is assigned for the token.

**Pattern Layout for [CLS] Token.** We here also provide additional visualization of semantic patterns emerge in [CLS] token, which is obtained via self-distillation on cross-view images. We also observe similar behavior in DINO since it's not a unique property brought by MIM. In fact, semantics are now believed to emerge as long as a similarity between two distorted views of one image is enforced [173], [204], [206], [254].

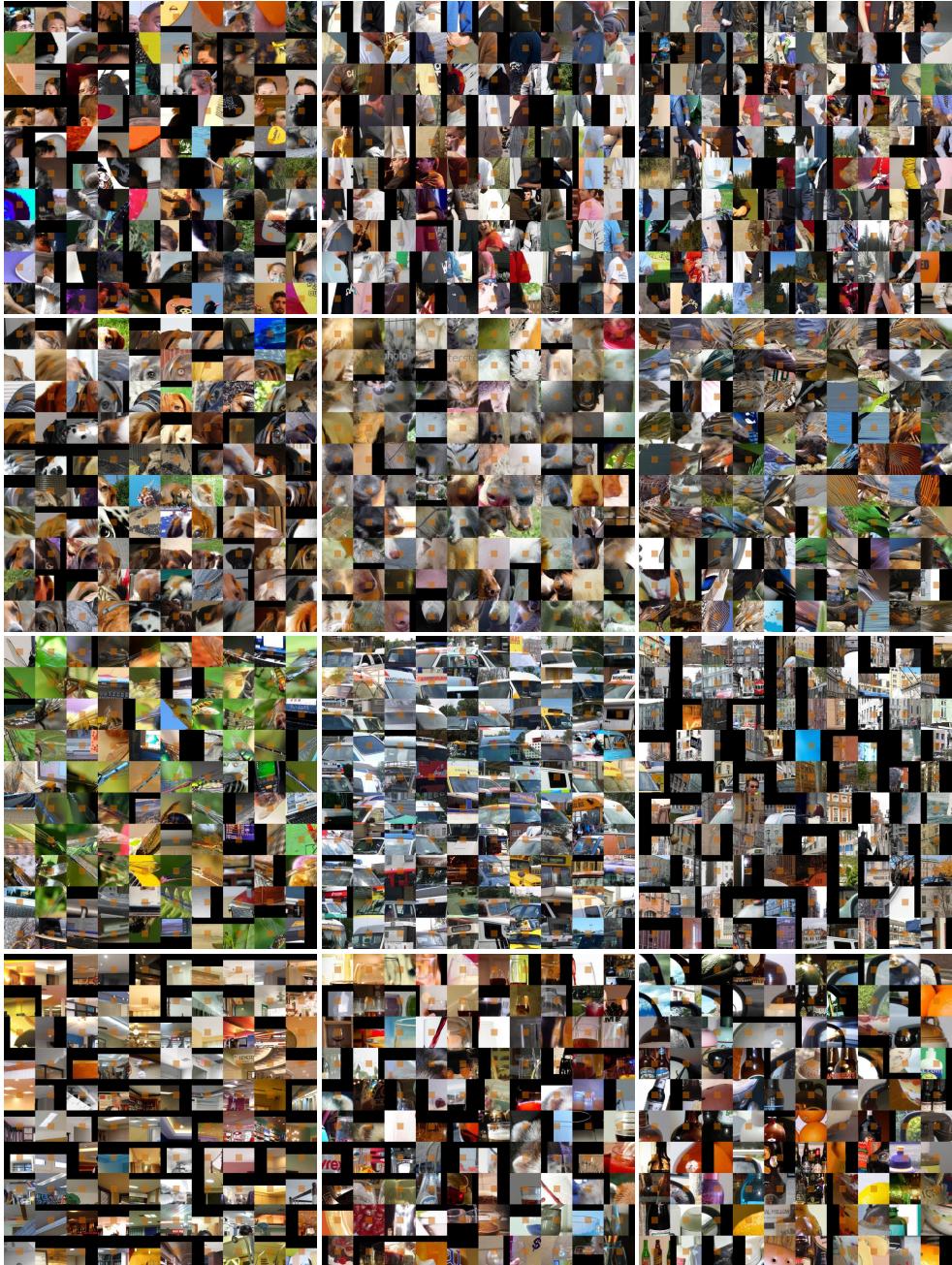
## E.7.2 Self-Attention Visualizations

Similar to the setting of Section 6.4.3, we here provided more self-attention map visualization from multiple heads of the last layer in Figure E.12.

## E.7.3 Sparse Correspondence.

We consider a sparse correspondence task where the overlapped patches from two augmented views of one image, or patches from two images labeled as one class, are required to be matched. The correlation is sparse since at most  $14 \times 14$  matched pairs can be extracted with a ViT-S/16 model. We visualize 12 correspondences with the highest self-attention score extracted from iBOT with ViT-S/16 pre-trained for 800 epochs. The score is averaged between multiple heads of the last layer. Several sampled sets of image pairs are shown in Figure E.13. We observe empirically that iBOT perform well for two views drawn from one image, nearly matched the majority of correspondence correctly. In the second

column, iBOT can match different parts of two instances from the same class (*e.g.*, tiles and windows of two cars) despite their huge differences in texture or color. We observe the DINO also has comparable visualized effects, illustrating the representation pre-trained with self-distillation also suits well for retrieval in a patch-level scale.

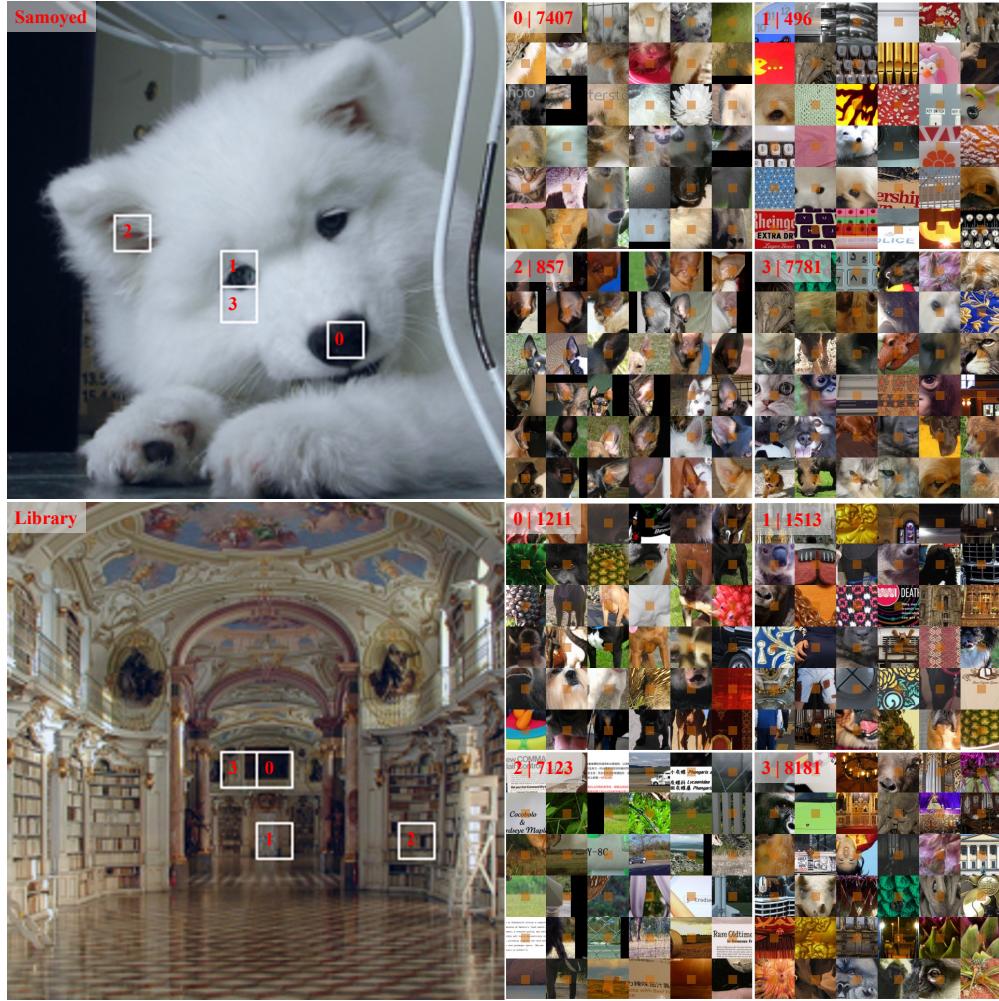


**Figure E.7. Visualization for pattern layout of patch tokens that share high-level semantics.** In the first row, we visualize different human-related semantic parts. We observe patterns for *human hair*, *human shoulder & arm*, and *human elbow* respectively. In other rows of the figure, we show semantic parts related to animals (*dog's ear*, *dog's nose*, *bird's wing*, *dragonfly's wing*), outdoor scenes (*front window of the vehicle*, *window of the architecture*) and indoor objects (*ceiling*, *glass bottle*).



**Figure E.8. Visualization for pattern layout of patch tokens that share low-level patterns.**

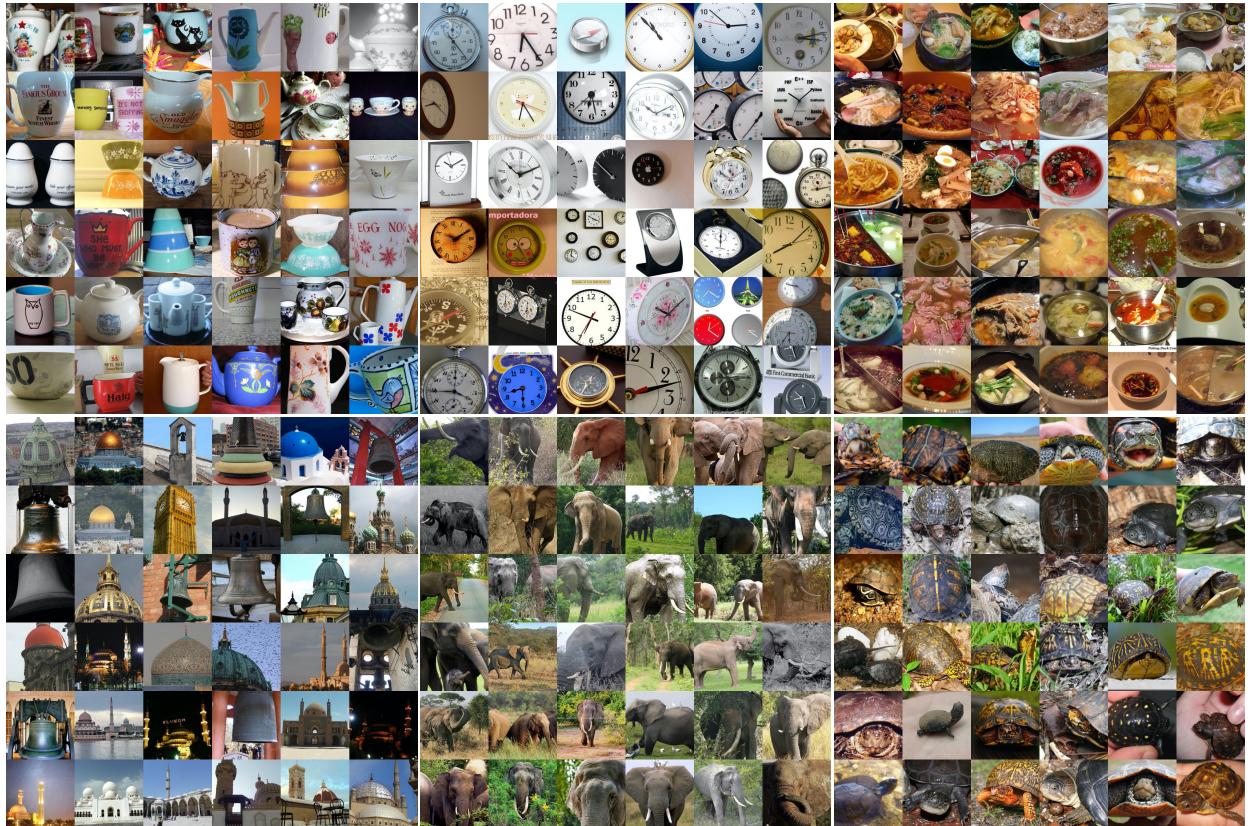
In the first two rows, we visualize patches that share similar textures. In the third row, we show some shape-related patterns, such as those of lines and similar curvatures. We also notice that some tokens capture color information, as shown in the last row.



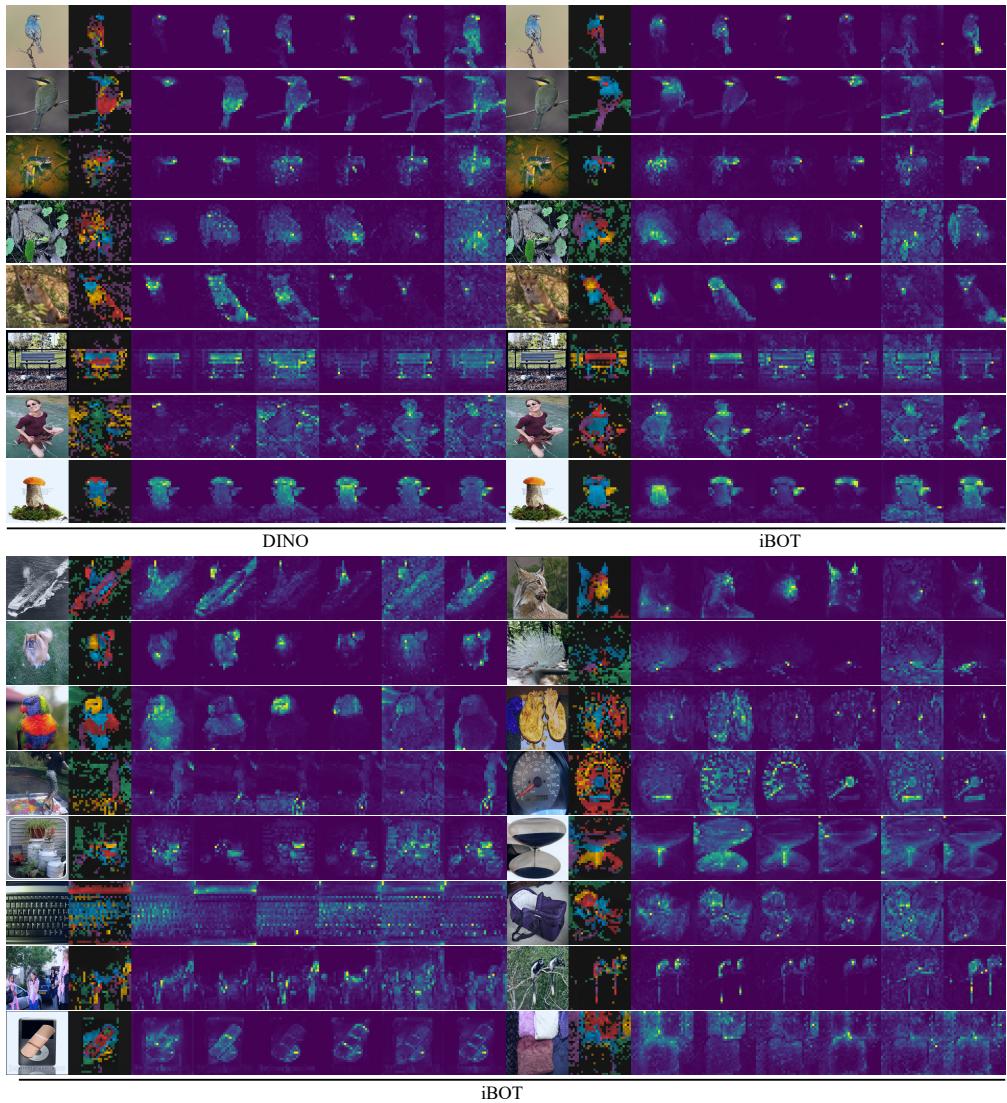
**Figure E.9. Top-4 representative patches with each of their pattern layout.** Order index **0, 1, 2, 3** are ranked according to its self-attention score. In the top-left corner for each pattern layout subfigure, its order index and cluster index are annotated. In the top panel, we can observe that pattern **0,2,3** show explicit semantic information of *nose*, *eyes*, *ears* respectively. Interestingly, patch **1** also locates around the eyes of the *Samoyed* but its corresponding pattern share visual similarity in shape instead of semantics. This illustrates the diverse behavior for each learned pattern. In the bottom panel, a *library* is represented by **0** *two- or multi-color joints*, **1,3** *knurling texture*, **2** *texts*. Similarly, we have patterns **0,1,3** focusing more on texture & color and pattern **2** focusing more on semantics. All of these visualized results illustrate versatile behavior for each index.



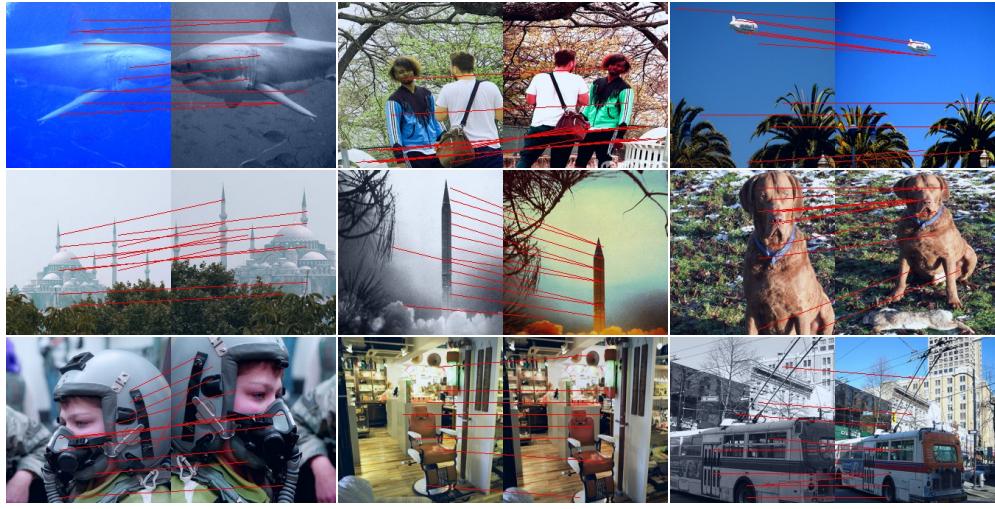
**Figure E.10. Visualization for pattern layout of patch tokens using BEiT (top) and DINO (bottom).** In the layout extracted from the DALL-E encoder, we observe minimal semantic patterns. In most cases, patches with similar color (*e.g.*, *black area* in left figure) or texture (*e.g.*, *line* in right figure) are clustered. In the layout extracted from DINO, while more complex textures are visible, most patches share similar local details instead of high-level semantics. In the right figure, the semantic part *eyes* can be somehow observed, yet it is mixed with plenty of irrelevant semantic parts.



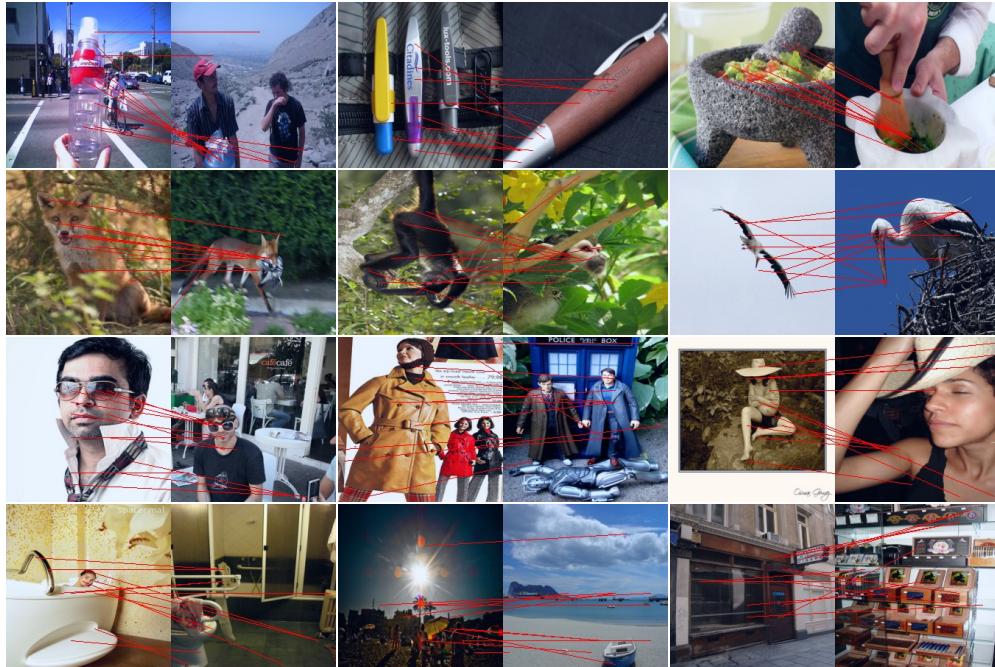
**Figure E.11. Visualization for pattern layout of [CLS] token.** We here indicate the high quality of semantic layout brought by self-distillation of cross-view images on [CLS] token. This property is not brought by MIM and is also prominent in DINO.



**Figure E.12. Visualization for self-attention map from Multiple Heads.** In the first 8 columns, we showcase iBOT’s attention map along with DINO’s. In the last 10 columns, we showcase more attention map from iBOT. We indicate that iBOT shows visually stronger ability to separate different objects or different parts of one object apart by giving more attentive visualized results for each part, compared with DINO. For example, in the fifth column, there is an attention head in iBOT accounting for the *ear of the fox* solely, while in DINO, it emerges with other parts; In the eighth column, iBOT separates the *mushroom* into more semantically meaningful parts.



Correspondence between two views of one image



Correspondence between two images of one class

**Figure E.13. Visualization for sparse correspondence.** The top panel shows images pairs sampled from two views of one image. The extracted correspondence from iBOT is mostly correct despite augmentations on scale and color. The bottom panel shows image pairs sampled from two images of the same class. For example, the second row shows animal images, and we observe that iBOT matches the semantic parts of animals correctly (e.g., *tails of the fox, beak of the bird*). These results demonstrate the capability of iBOT in part retrieval or matching in a local scale.

# Bibliography

- [1] Huiyu Wang, Aniruddha Kembhavi, Ali Farhadi, Alan Yuille, and Mohammad Rastegari, “ELASTIC: Improving cnns with dynamic scaling policies,” in *CVPR*, 2019.
- [2] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen, “Axial-DeepLab: Stand-alone axial-attention for panoptic segmentation,” in *ECCV*, 2020.
- [3] Huiyu Wang, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen, “MaX-DeepLab: End-to-end panoptic segmentation with mask transformers,” in *CVPR*, 2021.
- [4] Chen Wei, Huiyu Wang, Wei Shen, and Alan Yuille, “CO2: Consistent contrast for unsupervised visual representation learning,” in *ICLR*, 2021.
- [5] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong, “iBOT: Image BERT pre-training with online tokenizer,” *arXiv:2111.07832*, 2021.
- [6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in *NAACL*, 2019.
- [7] Tony Lindeberg, “Scale-space theory: A basic tool for analyzing structures at different scales,” *Journal of applied statistics*, 1994.
- [8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, “Imagenet classification with deep convolutional neural networks,” in *NeurIPS*, 2012.
- [9] Karen Simonyan and Andrew Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *ICLR*, 2015.

- [10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.
- [11] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger, "Densely connected convolutional networks.," in *CVPR*, 2017.
- [12] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell, "Deep layer aggregation," in *CVPR*, 2018.
- [13] Barret Zoph and Quoc V Le, "Neural architecture search with reinforcement learning," in *ICLR*, 2017.
- [14] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei, "Imagenet large scale visual recognition challenge," *IJCV*, vol. 115, pp. 211–252, 2015.
- [15] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick, "Microsoft coco: Common objects in context," in *ECCV*, 2014.
- [16] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He, "Aggregated residual transformations for deep neural networks," in *CVPR*, 2017.
- [17] Jie Hu, Li Shen, and Gang Sun, "Squeeze-and-excitation networks," in *CVPR*, 2018.
- [18] Navneet Dalal and Bill Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005.
- [19] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan, "Object detection with discriminatively trained part-based models," *IEEE TPAMI*, 2010.
- [20] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in *ICLR*, 2014.

- [21] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, “Going deeper with convolutions,” in *CVPR*, 2015.
- [22] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg, “Ssd: Single shot multibox detector,” in *ECCV*, 2016.
- [23] Zhaowei Cai, Quanfu Fan, Rogerio S Feris, and Nuno Vasconcelos, “A unified multi-scale deep convolutional neural network for fast object detection,” in *ECCV*, 2016.
- [24] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna, “Rethinking the inception architecture for computer vision,” in *CVPR*, 2016.
- [25] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning.,” in *AAAI*, 2017.
- [26] Sachin Mehta, Mohammad Rastegari, Anat Caspi, Linda Shapiro, and Hannaneh Hajishirzi, “Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation,” in *ECCV*, 2018.
- [27] Fisher Yu and Vladlen Koltun, “Multi-scale context aggregation by dilated convolutions,” *ICLR*, 2016.
- [28] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser, “Dilated residual networks,” *CVPR*, 2017.
- [29] Tianshui Chen, Liang Lin, Wangmeng Zuo, Xiaonan Luo, and Lei Zhang, “Learning a wavelet-like auto-encoder to accelerate deep neural networks,” in *AAAI*, 2018.
- [30] Hong-Yu Zhou, Bin-Bin Gao, and Jianxin Wu, “Adaptive feeding: Achieving fast and accurate detections by adaptively combining object detectors,” in *ICCV*, 2017.
- [31] Jonathan Long, Evan Shelhamer, and Trevor Darrell, “Fully convolutional networks for semantic segmentation,” in *CVPR*, 2015.
- [32] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik, “Hypercolumns for object segmentation and fine-grained localization,” in *CVPR*, 2015.

- [33] Tao Kong, Anbang Yao, Yurong Chen, and Fuchun Sun, "Hypernet: Towards accurate region proposal generation and joint object detection," in *CVPR*, 2016.
- [34] Wei Liu, Andrew Rabinovich, and Alexander C Berg, "Parsenet: Looking wider to see better," in *ICLR Workshop*, 2016.
- [35] Sean Bell, C Lawrence Zitnick, Kavita Bala, and Ross Girshick, "Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks," in *CVPR*, 2016.
- [36] Pedro O Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár, "Learning to refine object segments," in *ECCV*, 2016.
- [37] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015.
- [38] Alejandro Newell, Kaiyu Yang, and Jia Deng, "Stacked hourglass networks for human pose estimation," in *ECCV*, 2016.
- [39] Sina Honari, Jason Yosinski, Pascal Vincent, and Christopher Pal, "Recombinator networks: Learning coarse-to-fine feature aggregation," in *CVPR*, 2016.
- [40] Gao Huang, Shichen Liu, Laurens van der Maaten, and Kilian Q Weinberger, "Condensenet: An efficient densenet using learned group convolutions," in *CVPR*, 2018.
- [41] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie, "Feature pyramid networks for object detection," in *CVPR*, 2017.
- [42] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *ICCV*, 2015.
- [43] Laurens van der Maaten and Geoffrey Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, 2008.
- [44] Feng Zhu, Hongsheng Li, Wanli Ouyang, Nenghai Yu, and Xiaogang Wang, "Learning spatial regularization with image-level supervisions for multi-label image classification," in *CVPR*, 2017.

- [45] Weifeng Ge, Sibei Yang, and Yizhou Yu, "Multi-evidence filtering and fusion for multi-label classification, object detection and semantic segmentation based on weakly supervised learning," in *CVPR*, 2018.
- [46] Junjie Zhang, Qi Wu, Chunhua Shen, Jian Zhang, and Jianfeng Lu, "Multi-label image classification with regional latent semantic dependencies," *IEEE TMM*, 2018.
- [47] Yuncheng Li, Yale Song, and Jiebo Luo, "Improving pairwise ranking for multi-label image classification," in *CVPR*, 2017.
- [48] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv:1706.05587*, 2017.
- [49] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, 1998.
- [50] David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski, "A learning algorithm for boltzmann machines," *Cognitive science*, 1985.
- [51] Richard Zhang, "Making convolutional networks shift-invariant again," in *ICML*, 2019.
- [52] Matthias Holschneider, Richard Kronland-Martinet, Jean Morlet, and Ph Tchamitchian, "A real-time algorithm for signal analysis with the help of the wavelet transform," in *Wavelets*, 1990.
- [53] Mark J Shensa, "The discrete wavelet transform: Wedding the a trous and mallat algorithms," *IEEE Transactions on signal processing*, 1992.
- [54] George Papandreou, Iasonas Kokkinos, and Pierre-Andre Savalle, "Modeling local and global deformations in deep learning: Epitomic convolution, multiple instance learning, and sliding window detection," in *CVPR*, 2015.
- [55] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," in *ICLR*, 2015.
- [56] Chao Peng, Xiangyu Zhang, Gang Yu, Guiming Luo, and Jian Sun, "Large kernel matters—improve semantic segmentation by global convolutional network," in *CVPR*, 2017.

- [57] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia, "Pyramid scene parsing network," in *CVPR*, 2017.
- [58] Barret Zoph and Quoc V. Le, "Neural architecture search with reinforcement learning," in *ICLR*, 2017.
- [59] Liang-Chieh Chen, Maxwell Collins, Yukun Zhu, George Papandreou, Barret Zoph, Florian Schroff, Hartwig Adam, and Jon Shlens, "Searching for efficient multi-scale architectures for dense image prediction," in *NeurIPS*, 2018.
- [60] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan Yuille, and Li Fei-Fei, "Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation," in *CVPR*, 2019.
- [61] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *NeurIPS*, 2017.
- [62] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv:1609.08144*, 2016.
- [63] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio, "Attention-based models for speech recognition," in *NeurIPS*, 2015.
- [64] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *ICASSP*, 2016.
- [65] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *ICML*, 2015.
- [66] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le, "Attention augmented convolutional networks," in *ICCV*, 2019.
- [67] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei, "Relation networks for object detection," in *CVPR*, 2018.

- [68] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu, “Ccnet: Criss-cross attention for semantic segmentation,” in *ICCV*, 2019.
- [69] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He, “Non-local neural networks,” in *CVPR*, 2018.
- [70] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He, “Feature denoising for improving adversarial robustness,” in *CVPR*, 2019.
- [71] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens, “Stand-alone self-attention in vision models,” in *NeurIPS*, 2019.
- [72] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin, “Local relation networks for image recognition,” in *ICCV*, 2019.
- [73] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans, “Axial attention in multidimensional transformers,” *arXiv:1912.12180*, 2019.
- [74] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder, “The mapillary vistas dataset for semantic understanding of street scenes,” in *ICCV*, 2017.
- [75] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele, “The cityscapes dataset for semantic urban scene understanding,” in *CVPR*, 2016.
- [76] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár, “Panoptic segmentation,” in *CVPR*, 2019.
- [77] Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen, “Panoptic-deeplab,” in *ICCV COCO + Mapillary Joint Recognition Challenge Workshop*, 2019.
- [78] ——, “Panoptic-DeepLab: A Simple, Strong, and Fast Baseline for Bottom-Up Panoptic Segmentation,” in *CVPR*, 2020.
- [79] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, “Mask r-cnn,” in *ICCV*, 2017.

- [80] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár, “Panoptic feature pyramid networks,” in *CVPR*, 2019.
- [81] Lorenzo Porzi, Samuel Rota Bulò, Aleksander Colovic, and Peter Kontschieder, “Seamless scene segmentation,” in *CVPR*, 2019.
- [82] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille, “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs,” *IEEE TPAMI*, 2017.
- [83] Jie Li, Allan Raventos, Arjun Bhargava, Takaaki Tagawa, and Adrien Gaidon, “Learning to fuse things and stuff,” *arXiv:1812.01192*, 2018.
- [84] Yanwei Li, Xinze Chen, Zheng Zhu, Lingxi Xie, Guan Huang, Dalong Du, and Xingang Wang, “Attention-guided unified network for panoptic segmentation,” in *CVPR*, 2019.
- [85] Huanyu Liu1, Chao Peng, Changqian Yu, Jingbo Wang, Xu Liu, Gang Yu, and Wei Jiang, “An end-to-end network for panoptic segmentation,” in *CVPR*, 2019.
- [86] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun, “Upsnet: A unified panoptic segmentation network,” in *CVPR*, 2019.
- [87] Qizhu Li, Xiaojuan Qi, and Philip HS Torr, “Unifying training and inference for panoptic segmentation,” in *CVPR*, 2020.
- [88] Konstantin Sofiiuk, Olga Barinova, and Anton Konushin, “Adaptis: Adaptive instance selection network,” in *ICCV*, 2019.
- [89] Tien-Ju Yang, Maxwell D Collins, Yukun Zhu, Jyh-Jing Hwang, Ting Liu, Xiao Zhang, Vivienne Sze, George Papandreou, and Liang-Chieh Chen, “Deeperlab: Single-shot image parser,” *arXiv:1902.05093*, 2019.
- [90] Naiyu Gao, Yanhu Shan, Yupei Wang, Xin Zhao, Yinan Yu, Ming Yang, and Kaiqi Huang, “Ssap: Single-shot instance segmentation with affinity pyramid,” in *ICCV*, 2019.
- [91] Yiding Liu, Siyu Yang, Bin Li, Wengang Zhou, Jizheng Xu, Houqiang Li, and Yan Lu, “Affinity derivation and graph merge for instance segmentation,” in *ECCV*, 2018.

- [92] Margret Keuper, Evgeny Levinkov, Nicolas Bonneel, Guillaume Lavoué, Thomas Brox, and Bjorn Andres, "Efficient decomposition of image and mesh graphs by lifted multicut," in *ICCV*, 2015.
- [93] Ujwal Bonde, Pablo F Alcantarilla, and Stefan Leutenegger, "Towards bounding-box free panoptic segmentation," in *Pattern Recognition*, 2021.
- [94] Luc Vincent and Pierre Soille, "Watersheds in digital spaces: An efficient algorithm based on immersion simulations," *IEEE TPAMI*, 1991.
- [95] Min Bai and Raquel Urtasun, "Deep watershed transform for instance segmentation," in *CVPR*, 2017.
- [96] Dana H Ballard, "Generalizing the hough transform to detect arbitrary shapes," *Pattern Recognition*, 1981.
- [97] Bastian Leibe, Ales Leonardis, and Bernt Schiele, "Combined object categorization and segmentation with an implicit shape model," in *ECCV Workshop*, 2004.
- [98] Alex Kendall, Yarin Gal, and Roberto Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *CVPR*, 2018.
- [99] Jonas Uhrig, Eike Rehder, Björn Fröhlich, Uwe Franke, and Thomas Brox, "Box2pix: Single-shot instance segmentation by assigning pixels to object boxes," in *IEEE Intelligent Vehicles Symposium (IV)*, 2018.
- [100] Davy Neven, Bert De Brabandere, Marc Proesmans, and Luc Van Gool, "Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth," in *CVPR*, 2019.
- [101] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *ECCV*, 2018.
- [102] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR*, 2015.

- [103] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck, “Music transformer: Generating music with long-term structure,” in *ICLR*, 2019.
- [104] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran, “Image transformer,” in *ICML*, 2018.
- [105] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani, “Self-attention with relative position representations,” in *NAACL*, 2018.
- [106] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov, “Transformer-xl: Attentive language models beyond a fixed-length context,” in *ACL*, 2019.
- [107] Yingwei Li, Xiaojie Jin, Jieru Mei, Xiaochen Lian, Linjie Yang, Cihang Xie, Qihang Yu, Yuyin Zhou, Song Bai, and Alan Yuille, “Neural architecture search for lightweight non-local networks,” in *CVPR*, 2020.
- [108] Antoni Buades, Bartomeu Coll, and J-M Morel, “A non-local algorithm for image denoising,” in *CVPR*, 2005.
- [109] Yunpeng Chen, Yannis Kalantidis, Jianshu Li, Shuicheng Yan, and Jiashi Feng, “ $\hat{A^2}$ -nets: Double attention networks,” in *NeurIPS*, 2018.
- [110] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu, “Dual attention network for scene segmentation,” in *CVPR*, 2019.
- [111] Zhen Zhu, Mengde Xu, Song Bai, Tengteng Huang, and Xiang Bai, “Asymmetric non-local neural networks for semantic segmentation,” in *CVPR*, 2019.
- [112] Xizhou Zhu, Dazhi Cheng, Zheng Zhang, Stephen Lin, and Jifeng Dai, “An empirical study of spatial attention mechanisms in deep networks,” in *ICCV*, 2019.
- [113] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li, “Efficient attention: Attention with linear complexities,” in *WACV*, 2021.
- [114] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena, “Self-attention generative adversarial networks,” in *ICML*, 2019.

- [115] Andrew Brock, Jeff Donahue, and Karen Simonyan, "Large scale gan training for high fidelity natural image synthesis," in *ICLR*, 2019.
- [116] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, "Tensorflow: A system for large-scale machine learning," in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, 2016.
- [117] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He, "Accurate, large minibatch sgd: Training imagenet in 1 hour," *arXiv:1706.02677*, 2017.
- [118] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han, "On the variance of the adaptive learning rate and beyond," in *ICLR*, 2020.
- [119] Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton, "Lookahead optimizer: K steps forward, 1 step back," in *NeurIPS*, 2019.
- [120] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, "Mobileneets: Efficient convolutional neural networks for mobile vision applications," *arXiv:1704.04861*, 2017.
- [121] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *CVPR*, 2018.
- [122] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, *et al.*, "Searching for mobilenetv3," in *ICCV*, 2019.
- [123] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.
- [124] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei, "Deformable convolutional networks," in *ICCV*, 2017.

- [125] Qiang Chen, Anda Cheng, Xiangyu He, Peisong Wang, and Jian Cheng, "Spatialflow: Bridging all tasks for panoptic segmentation," *IEEE TCSVT*, 2019.
- [126] Yibo Yang, Hongyang Li, Xia Li, Qijie Zhao, Jianlong Wu, and Zhouchen Lin, "Sognet: Scene overlap graph network for panoptic segmentation," in *AAAI*, 2020.
- [127] François Chollet, "Xception: Deep learning with depthwise separable convolutions," in *CVPR*, 2017.
- [128] Haozhi Qi, Zheng Zhang, Bin Xiao, Han Hu, Bowen Cheng, Yichen Wei, and Jifeng Dai, "Deformable convolutional networks – coco detection and segmentation challenge 2017 entry," *ICCV COCO Challenge Workshop*, 2017.
- [129] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao, "Deep high-resolution representation learning for visual recognition," *TPAMI*, 2021.
- [130] Xiangtai Li, Houlong Zhao, Lei Han, Yunhai Tong, and Kuiyuan Yang, "Gff: Gated fully fusion for semantic segmentation," in *AAAI*, 2020.
- [131] Yi Zhu, Karan Sapra, Fitsum A Reda, Kevin J Shih, Shawn Newsam, Andrew Tao, and Bryan Catanzaro, "Improving semantic segmentation via video propagation and label relaxation," in *CVPR*, 2019.
- [132] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia, "Path aggregation network for instance segmentation," in *CVPR*, 2018.
- [133] Justin Liang, Namdar Homayounfar, Wei-Chiu Ma, Yuwen Xiong, Rui Hu, and Raquel Urtasun, "Polytransform: Deep polygon transformer for instance segmentation," in *CVPR*, 2020.
- [134] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman, "Speeding up convolutional neural networks with low rank expansions," in *BMVC*, 2014.
- [135] Laurent Sifre, "Rigid-motion scattering for image classification," *PhD thesis*, 2014.
- [136] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai, "Deformable convnets v2: More deformable, better results," in *CVPR*, 2019.

- [137] Hang Gao, Xizhou Zhu, Steve Lin, and Jifeng Dai, "Deformable kernels: Adapting effective receptive fields for object deformation," in *ICLR*, 2020.
- [138] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille, "Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution," in *CVPR*, 2021.
- [139] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *NeurIPS*, 2015.
- [140] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z Li, "Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection," in *CVPR*, 2020.
- [141] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S Davis, "Soft-nms—improving object detection with one line of code," in *ICCV*, 2017.
- [142] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko, "End-to-end object detection with transformers," in *ECCV*, 2020.
- [143] Liang-Chieh Chen, Raphael Gontijo Lopes, Bowen Cheng, Maxwell D Collins, Ekin D Cubuk, Barret Zoph, Hartwig Adam, and Jonathon Shlens, "Naive-Student: Leveraging Semi-Supervised Learning in Video Sequences for Urban Scene Segmentation," in *ECCV*, 2020.
- [144] Harold W Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, 1955.
- [145] Russell Stewart, Mykhaylo Andriluka, and Andrew Y Ng, "End-to-end people detection in crowded scenes," in *CVPR*, 2016.
- [146] Bowen Cheng, Liang-Chieh Chen, Yunchao Wei, Yukun Zhu, Zilong Huang, Jinjun Xiong, Thomas S Huang, Wen-Mei Hwu, and Honghui Shi, "Spnet: Semantic prediction guidance for scene parsing," in *ICCV*, 2019.
- [147] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick, *Detectron2*, <https://github.com/facebookresearch/detectron2>, 2019.

- [148] Jianpeng Cheng, Li Dong, and Mirella Lapata, “Long short-term memory-networks for machine reading,” in *EMNLP*, 2016.
- [149] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya, “Reformer: The efficient transformer,” in *ICLR*, 2020.
- [150] Sinong Wang, Belinda Li, Madian Khabsa, Han Fang, and Hao Ma, “Linformer: Self-attention with linear complexity,” *arXiv:2006.04768*, 2020.
- [151] Minh-Thang Luong, Hieu Pham, and Christopher D Manning, “Effective approaches to attention-based neural machine translation,” in *EMNLP*, 2015.
- [152] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever, “Generating long sequences with sparse transformers,” *arXiv:1904.10509*, 2019.
- [153] Iz Beltagy, Matthew E Peters, and Arman Cohan, “Longformer: The long-document transformer,” *arXiv preprint arXiv:2004.05150*, 2020.
- [154] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, *et al.*, “Big bird: Transformers for longer sequences,” in *NeurIPS*, 2020.
- [155] Ankit Gupta and Jonathan Berant, “Gmat: Global memory augmentation for transformers,” *arXiv:2006.03274*, 2020.
- [156] Joshua Ainslie, Santiago Ontanon, Chris Alberti, Philip Pham, Anirudh Ravula, and Sumit Sanghai, “Etc: Encoding long and structured data in transformers,” in *EMNLP*, 2020.
- [157] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” in *ICLR*, 2020.
- [158] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li, “Efficient attention: Attention with linear complexities,” in *WACV*, 2021.
- [159] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai, “Deformable detr: Deformable transformers for end-to-end object detection,” in *ICLR*, 2020.

- [160] Liang-Chieh Chen, Yi Yang, Jiang Wang, Wei Xu, and Alan L Yuille, “Attention to scale: Scale-aware semantic image segmentation,” in *CVPR*, 2016.
- [161] Hengshuang Zhao, Yi Zhang, Shu Liu, Jianping Shi, Chen Change Loy, Dahua Lin, and Jiaya Jia, “Psanet: Point-wise spatial attention network for scene parsing,” in *ECCV*, 2018.
- [162] Yibo Yang, Hongyang Li, Xia Li, Qijie Zhao, Jianlong Wu, and Zhouchen Lin, “Sognet: Scene overlap graph network for panoptic segmentation,” in *AAAI*, 2020.
- [163] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár, “Focal loss for dense object detection,” in *ICCV*, 2017.
- [164] Mingxing Tan, Ruoming Pang, and Quoc V Le, “Efficientdet: Scalable and efficient object detection,” in *CVPR*, 2020.
- [165] Zhi Tian, Chunhua Shen, and Hao Chen, “Conditional convolutions for instance segmentation,” in *ECCV*, 2020.
- [166] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen, “SOLOv2: Dynamic and fast instance segmentation,” in *NeurIPS*, 2020.
- [167] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool, “Dynamic filter networks,” in *NeurIPS*, 2016.
- [168] Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam, “Condconv: Conditionally parameterized convolutions for efficient inference,” in *NeurIPS*, 2019.
- [169] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin, “Unsupervised feature learning via non-parametric instance discrimination,” in *CVPR*, 2018.
- [170] Zhirong Wu, Alexei A Efros, and Stella X Yu, “Improving generalization via scalable neighborhood component analysis,” in *ECCV*, 2018.
- [171] Jyh-Jing Hwang, Stella X Yu, Jianbo Shi, Maxwell D Collins, Tien-Ju Yang, Xiao Zhang, and Liang-Chieh Chen, “SegSort: Segmentation by discriminative sorting of segments,” in *ICCV*, 2019.
- [172] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton, “A simple framework for contrastive learning of visual representations,” in *ICML*, 2020.

- [173] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick, "Momentum contrast for unsupervised visual representation learning," in *CVPR*, 2020.
- [174] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan, "Supervised contrastive learning," in *NeurIPS*, 2020.
- [175] Fabian Isensee, Paul F Jaeger, Simon AA Kohl, Jens Petersen, and Klaus H Maier-Hein, "Nnunet: A self-configuring method for deep learning-based biomedical image segmentation," *Nature methods*, 2021.
- [176] Saeid Asgari Taghanaki, Yefeng Zheng, S Kevin Zhou, Bogdan Georgescu, Puneet Sharma, Daguang Xu, Dorin Comaniciu, and Ghassan Hamarneh, "Combo loss: Handling input and output imbalance in multi-organ segmentation," *Computerized Medical Imaging and Graphics*, 2019.
- [177] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *3DV*, 2016.
- [178] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger, "Deep networks with stochastic depth," in *ECCV*, 2016.
- [179] Sergey Zagoruyko and Nikos Komodakis, "Wide residual networks," in *BMVC*, 2016.
- [180] Zifeng Wu, Chunhua Shen, and Anton Van Den Hengel, "Wider or deeper: Revisiting the ResNet model for visual recognition," *Pattern Recognition*, 2019.
- [181] Yudong Liu, Yongtao Wang, Siwei Wang, TingTing Liang, Qijie Zhao, Zhi Tang, and Haibin Ling, "Cbnet: A novel composite backbone network architecture for object detection.," in *AAAI*, 2020.
- [182] Carl Doersch, Abhinav Gupta, and Alexei A Efros, "Unsupervised visual representation learning by context prediction," in *ECCV*, 2015.
- [183] Spyros Gidaris, Praveer Singh, and Nikos Komodakis, "Unsupervised representation learning by predicting image rotations," in *ICLR*, 2018.

- [184] Richard Zhang, Phillip Isola, and Alexei A Efros, “Colorful image colorization,” in *ECCV*, 2016.
- [185] Philip Bachman, R Devon Hjelm, and William Buchwalter, “Learning representations by maximizing mutual information across views,” in *NeurIPS*, 2019.
- [186] Olivier Henaff, “Data-efficient image recognition with contrastive predictive coding,” in *ICML*, 2020.
- [187] Yonglong Tian, Dilip Krishnan, and Phillip Isola, “Contrastive multiview coding,” in *ECCV*, 2020.
- [188] Ishan Misra and Laurens van der Maaten, “Self-supervised learning of pretext-invariant representations,” in *CVPR*, 2020.
- [189] Raia Hadsell, Sumit Chopra, and Yann LeCun, “Dimensionality reduction by learning an invariant mapping,” in *CVPR*, 2006.
- [190] Nikunj Saunshi, Orestis Plevrakis, Sanjeev Arora, Mikhail Khodak, and Hrishikesh Khan-deparkar, “A theoretical analysis of contrastive unsupervised representation learning,” in *ICML*, 2019.
- [191] Ching-Yao Chuang, Joshua Robinson, Lin Yen-Chen, Antonio Torralba, and Stefanie Jegelka, “Debiased contrastive learning,” in *NeurIPS*, 2020.
- [192] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen, “Regularization with stochastic transformations and perturbations for deep semi-supervised learning,” in *NeurIPS*, 2016.
- [193] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le, “Unsupervised data augmentation for consistency training,” in *NeurIPS*, 2020.
- [194] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel, “MixMatch: A holistic approach to semi-supervised learning,” in *NeurIPS*, 2019.
- [195] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel, “FixMatch: Simplifying semi-supervised learning with consistency and confidence,” in *NeurIPS*, 2020.

- [196] Mehdi Noroozi and Paolo Favaro, “Unsupervised learning of visual representations by solving jigsaw puzzles,” in *ECCV*, 2016.
- [197] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *CVPR*, 2009.
- [198] Mang Ye, Xu Zhang, Pong C Yuen, and Shih-Fu Chang, “Unsupervised embedding learning via invariant and spreading instance feature,” in *CVPR*, 2019.
- [199] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He, “Improved baselines with momentum contrastive learning,” *arXiv preprint arXiv:2003.04297*, 2020.
- [200] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, “Distilling the knowledge in a neural network,” in *NeurIPS Workshop*, 2015.
- [201] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze, “Deep clustering for unsupervised learning of visual features,” in *ECCV*, 2018.
- [202] Chengxu Zhuang, Alex Lin Zhai, and Daniel Yamins, “Local aggregation for unsupervised learning of visual embeddings,” in *CVPR*, 2019.
- [203] Junnan Li, Pan Zhou, Caiming Xiong, and Steven Hoi, “Prototypical contrastive learning of unsupervised representations,” in *ICLR*, 2021.
- [204] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin, “Unsupervised learning of visual features by contrasting cluster assignments,” in *NeurIPS*, 2020.
- [205] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox, “Discriminative unsupervised feature learning with convolutional neural networks,” in *NeurIPS*, 2014.
- [206] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, koray kavukcuoglu koray, Remi Munos, and Michal Valko, in *NeurIPS*, 2020.

- [207] Xinlei Chen and Kaiming He, “Exploring simple siamese representation learning,” in *CVPR*, 2021.
- [208] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman, “The pascal visual object classes challenge: A retrospective,” *IJCV*, 2015.
- [209] Priya Goyal, Dhruv Mahajan, Abhinav Gupta, and Ishan Misra, “Scaling and benchmarking self-supervised visual representation learning,” in *ICCV*, 2019.
- [210] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik, “A training algorithm for optimal margin classifiers,” in *Proceedings of the fifth annual workshop on computational learning theory*, 1992.
- [211] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, “Mask R-CNN,” in *ICCV*, 2017.
- [212] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna, “Rethinking the inception architecture for computer vision,” in *CVPR*, 2016.
- [213] Chen Wei, Lingxi Xie, Xutong Ren, Yingda Xia, Chi Su, Jiaying Liu, Qi Tian, and Alan L Yuille, “Iterative reorganization with weak spatial constraints: Solving arbitrary jigsaw puzzles for unsupervised representation learning,” in *CVPR*, 2019.
- [214] G. Larsson, M. Maire, and G. Shakhnarovich, “Colorization as a proxy task for visual understanding,” in *CVPR*, 2017.
- [215] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros, “Context encoders: Feature learning by inpainting,” in *CVPR*, 2016.
- [216] Mehdi Noroozi, Hamed Pirsiavash, and Paolo Favaro, “Representation learning by learning to count,” in *ICCV*, 2017.
- [217] YM Asano, C Rupprecht, and A Vedaldi, “Self-labelling via simultaneous clustering and representation learning,” in *ICLR*, 2019.
- [218] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell, “Adversarial feature learning,” in *ICLR*, 2016.

- [219] Jeff Donahue and Karen Simonyan, "Large scale adversarial representation learning," in *NeurIPS*, 2019.
- [220] Carl Doersch and Andrew Zisserman, "Multi-task self-supervised visual learning," in *ICCV*, 2017.
- [221] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer, "Revisiting self-supervised visual representation learning," in *CVPR*, 2019.
- [222] R Devon Hjelm, Alefx Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio, "Learning deep representations by mutual information estimation and maximization," in *ICLR*, 2019.
- [223] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii, "Virtual adversarial training: A regularization method for supervised and semi-supervised learning," *TPAMI*, 2018.
- [224] Antti Tarvainen and Harri Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *NeurIPS*, 2017.
- [225] Samuli Laine and Timo Aila, "Temporal ensembling for semi-supervised learning," in *ICLR*, 2017.
- [226] David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel, "ReMixMatch: Semi-supervised learning with distribution matching and augmentation anchoring," in *ICLR*, 2019.
- [227] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou, "Training data-efficient image transformers & distillation through attention," in *ICML*, 2021.
- [228] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *ICLR*, 2021.

- [229] Xinlei Chen, Saining Xie, and Kaiming He, “An empirical study of training self-supervised vision transformers,” in *ICCV*, 2021.
- [230] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin, “Emerging properties in self-supervised vision transformers,” in *ICCV*, 2021.
- [231] Rico Sennrich, Barry Haddow, and Alexandra Birch, “Neural machine translation of rare words with subword units,” in *ACL*, 2016.
- [232] Sara Atito, Muhammad Awais, and Josef Kittler, “SiT: Self-supervised vision transformer,” *arXiv preprint arXiv:2104.03602*, 2021.
- [233] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang, “Self-supervised learning: Generative or contrastive,” *TKDE*, 2021.
- [234] Hangbo Bao, Li Dong, and Furu Wei, “BEiT: BERT pre-training of image transformers,” *arXiv preprint arXiv:2106.08254*, 2021.
- [235] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever, “Zero-shot text-to-image generation,” in *ICML*, 2021.
- [236] Hao Tan, Jie Lei, Thomas Wolf, and Mohit Bansal, “Vimpac: Video pre-training via masked token prediction and contrastive learning,” *arXiv preprint arXiv:2106.11250*, 2021.
- [237] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean, “Distilling the knowledge in a neural network,” in *NeurIPS*, 2015.
- [238] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo, “Swin transformer: Hierarchical vision transformer using shifted windows,” in *ICCV*, 2021.
- [239] Ilya Loshchilov and Frank Hutter, “Decoupled weight decay regularization,” in *ICLR*, 2019.
- [240] Chunyuan Li, Jianwei Yang, Pengchuan Zhang, Mei Gao, Bin Xiao, Xiyang Dai, Lu Yuan, and Jianfeng Gao, “Efficient self-supervised vision transformers for representation learning,” *arXiv preprint arXiv:2106.09785*, 2021.

- [241] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton, “Big self-supervised models are strong semi-supervised learners,” in *NeurIPS*, 2020.
- [242] Yuki M. Asano, Christian Rupprecht, and Andrea Vedaldi, “Self-labelling via simultaneous clustering and representation learning,” in *ICLR*, 2020.
- [243] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola, “What makes for good views for contrastive learning?” In *NeurIPS*, 2020.
- [244] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool, “Scan: Learning to classify images without labels,” in *ECCV*, 2020.
- [245] Zhaowei Cai and Nuno Vasconcelos, “Cascade R-CNN: High quality object detection and instance segmentation,” *TPAMI*, 2019.
- [246] Wenhui Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao, “Pyramid vision transformer: A versatile backbone for dense prediction without convolutions,” in *ICCV*, 2021.
- [247] Zhenda Xie, Yutong Lin, Zhuliang Yao, Zheng Zhang, Qi Dai, Yue Cao, and Han Hu, “Self-supervised learning with swin transformers,” *arXiv preprint arXiv:2105.04553*, 2021.
- [248] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba, “Scene parsing through ade20k dataset,” in *CVPR*, 2017.
- [249] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun, “Unified perceptual parsing for scene understanding,” in *ECCV*, 2018.
- [250] Muzammal Naseer, Kanchana Ranasinghe, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang, “Intriguing properties of vision transformers,” in *NeurIPS*, 2021.
- [251] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song, “Natural adversarial examples,” in *CVPR*, 2021.
- [252] Dan Hendrycks and Thomas Dietterich, “Benchmarking neural network robustness to common corruptions and perturbations,” in *ICLR*, 2019.

- [253] Elad Amrani and Alex Bronstein, "Self-supervised classification network," *arXiv preprint arXiv:2103.10994*, 2021.
- [254] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze, "Deep clustering for unsupervised learning of visual features," in *ECCV*, 2018.
- [255] Asano YM., Rupprecht C., and Vedaldi A., "Self-labelling via simultaneous clustering and representation learning," in *ICLR*, 2020.
- [256] Chen Wei, Lingxi Xie, Xutong Ren, Yingda Xia, Chi Su, Jiaying Liu, Qi Tian, and Alan L Yuille, "Iterative reorganization with weak spatial constraints: Solving arbitrary jigsaw puzzles for unsupervised representation learning," in *CVPR*, 2019.
- [257] Nikos Komodakis and Spyros Gidaris, "Unsupervised representation learning by predicting image rotations," in *ICLR*, 2018.
- [258] Carl Doersch, Abhinav Gupta, and Alexei A Efros, "Unsupervised visual representation learning by context prediction," in *ICCV*, 2015.
- [259] Zhaowen Li, Zhiyang Chen, Fan Yang, Wei Li, Yousong Zhu, Chaoyang Zhao, Rui Deng, Liwei Wu, Rui Zhao, Ming Tang, *et al.*, "MST: Masked self-supervised transformer for visual representation," in *NeurIPS*, 2021.
- [260] Yucheng Zhao, Guangting Wang, Chong Luo, Wenjun Zeng, and Zheng-Jun Zha, "Self-supervised visual representations learning by contrastive mask prediction," *arXiv preprint arXiv:2108.07954*, 2021.
- [261] Jason Tyler Rolfe, "Discrete variational autoencoders," in *ICLR*, 2017.
- [262] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai, "Vi-bert: Pre-training of generic visual-linguistic representations," in *ICLR*, 2020.
- [263] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee, "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks," in *NeurIPS*, 2019.
- [264] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu, "Uniter: Universal image-text representation learning," in *ECCV*, 2020.

- [265] Simone Bianco, Remi Cadene, Luigi Celona, and Paolo Napoletano, “Benchmark analysis of representative deep neural network architectures,” *IEEE Access*, 2018.
- [266] Jie Hu, Li Shen, and Gang Sun, “Squeeze-and-excitation networks,” in *CVPR*, 2018.
- [267] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, “Deep residual learning for image recognition,” in *CVPR*, 2016.
- [268] Sergey Ioffe and Christian Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *ICML*, 2015.
- [269] Ilya Loshchilov and Frank Hutter, “SGDR: Stochastic gradient descent with warm restarts,” in *ICLR*, 2016.
- [270] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick, *Detectron2*, <https://github.com/facebookresearch/detectron2>, 2019.
- [271] Jonathan Long, Evan Shelhamer, and Trevor Darrell, “Fully convolutional networks for semantic segmentation,” in *CVPR*, 2015.
- [272] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille, “DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs,” *TPAMI*, 2017.
- [273] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik, “Semantic contours from inverse detectors,” in *ICCV*, 2011.
- [274] Kai Yuanqing Xiao, Logan Engstrom, Andrew Ilyas, and Aleksander Madry, “Noise or signal: The role of image backgrounds in object recognition,” in *ICLR*, 2020.
- [275] Zhenda Xie, Yutong Lin, Zheng Zhang, Yue Cao, Stephen Lin, and Han Hu, “Propagate yourself: Exploring pixel-level consistency for unsupervised visual representation learning,” in *CVPR*, 2021.
- [276] Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li, “Dense contrastive learning for self-supervised visual pre-training,” in *CVPR*, 2021.

# Huiyu Wang

---

(310) 948 - 0803  
huiyu@jhu.edu  
<https://csrhddlam.github.io>

EDUCATION		
	<b>Johns Hopkins University</b> Ph.D. Candidate in Computer Science Advisor: Alan Yuille	Aug 2017 - present
	<b>University of California, Los Angeles</b> M.S. in Electrical Engineering GPA: 3.97 / 4.00	Sep 2015 - Dec 2016
	<b>Shanghai Jiao Tong University</b> B.S. in Information Engineering GPA: 90.0 / 100.0	Sep 2011 - Jun 2015
EXPERIENCE		
	<b>Google LLC</b> , Remote Student Researcher Mentors: Liang-Chieh Chen, Yukun Zhu	Aug 2019 - Aug 2021
	<b>Google LLC</b> , Seattle, WA Research Intern Mentors: Yukun Zhu, Liang-Chieh Chen	May 2019 - Aug 2019
	<b>Allen Institute for Artificial Intelligence</b> , Seattle, WA Research Intern <i>AI2 Outstanding Intern of 2018 Award</i> Mentors: Mohammad Rastegari, Aniruddha Kembhavi, Ali Farhadi	May 2018 - Aug 2018
	<b>Johns Hopkins University</b> , Baltimore, MD Research Assistant Advisor: Alan Yuille	Feb 2017 - Jun 2017
	<b>TuSimple LLC</b> , San Diego, CA Research Engineering Intern Mentor: Xiaodi Hou	Jun 2016 - Nov 2016
	<b>University of California, Los Angeles</b> , Los Angeles, CA Research Assistant Advisors: Ying Nian Wu, Song-Chun Zhu	Apr 2016 - Jun 2016
	<b>Shanghai Jiao Tong University</b> , Shanghai, China Undergraduate Researcher Advisor: Li Song	Dec 2014 - Jun 2015
PUBLICATIONS	Jinghao Zhou, Chen Wei, <b>Huiyu Wang</b> , Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. iBOT: Image BERT pre-training with online tokenizer. <i>Computing Research Repository, arXiv:2111.07832</i> , 2021.	
	Huaijin Pi, <b>Huiyu Wang</b> , Yingwei Li, Zizhang Li, and Alan Yuille. Searching for	

TrioNet: Combining Convolution with Local and Global Self-Attention. In *Proceedings of the 32nd British Machine Vision Conference (BMVC)*, 2021.

Mark Weber\*, **Huiyu Wang**\*, Siyuan Qiao\*, Jun Xie, Maxwell D. Collins, Yukun Zhu, Liangzhe Yuan, Dahun Kim, Qihang Yu, Daniel Cremers, Laura Leal-Taixe, Alan Yuille, Florian Schroff, Hartwig Adam, and Liang-Chieh Chen. DeepLab2: A TensorFlow Library for Deep Labeling. *Computing Research Repository, arXiv:2106.09748*, 2021.

**Huiyu Wang**, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. MaX-DeepLab: End-to-End Panoptic Segmentation with Mask Transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5463-5474, 2021.

Chen Wei, **Huiyu Wang**, Wei Shen, and Alan Yuille. CO2: Consistent Contrast for Unsupervised Visual Representation Learning. In *9th International Conference on Learning Representations (ICLR)*, 2021.

Liang-Chieh Chen, **Huiyu Wang**, and Siyuan Qiao. Scaling Wide Residual Networks for Panoptic Segmentation. *Computing Research Repository, arXiv:2011.11675*, 2020.

**Huiyu Wang**, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-DeepLab: Stand-Alone Axial-Attention for Panoptic Segmentation. In *Proceedings of the 16th European Conference on Computer Vision (ECCV)*, pages 108-126, Springer, Cham, 2020. (**Spotlight**)

Siyuan Qiao, **Huiyu Wang**, Chenxi Liu, Wei Shen, and Alan Yuille. Rethinking Normalization and Elimination Singularity in Neural Networks. *Computing Research Repository, arXiv:1911.09738*, 2019.

Adam Kortylewski, Qing Liu, **Huiyu Wang**, Zhishuai Zhang, and Alan Yuille. Combining Compositional Models and Deep Networks For Robust Object Classification under Occlusion. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1333-1341, 2020. (**Spotlight**)

Qing Liu, Lingxi Xie, **Huiyu Wang**, and Alan Yuille. Semantic-Aware Knowledge Preservation for Zero-Shot Sketch-Based Image Retrieval. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3662-3671, 2019.

Siyuan Qiao, **Huiyu Wang**, Chenxi Liu, Wei Shen, and Alan Yuille. Weight Standardization. *Computing Research Repository, arXiv:1903.10520*, 2019.

**Huiyu Wang**, Aniruddha Kembhavi, Ali Farhadi, Alan Yuille, and Mohammad Rastegari. ELASTIC: Improving CNNs with Dynamic Scaling Policies. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2258-2267, 2019. (**Oral**)

Zhixuan Wei, Weidong Chen, Jingchuan Wang, **Huiyu Wang**, and Kang Li. Semantic Mapping for Safe and Comfortable Navigation of a Brain-Controlled Wheelchair. In *Proceedings of the 6th International Conference on Intelligent Robotics and Applications (ICIRA)*, pages 307-317, Springer, Berlin, Heidelberg, 2013.

FELLOWSHIPS  
AND AWARDS

Outstanding Reviewer, CVPR  
Outstanding Reviewer, ECCV

2021  
2020

AI2 Outstanding Intern	2018
The SCSK® Scholarship	2014
2nd prize, National Undergraduate Electronic Design Contest	2013
1st prize, TI® Cup Electronic Design Contest of SJTU	2013
Academic Excellence Scholarship, 2nd class, SJTU	2012, 2013, 2014

#### SKILLS

Programming Languages: Python, C++, Java, Matlab, C#, L<sup>A</sup>T<sub>E</sub>X  
Deep Learning Tools: PyTorch, TensorFlow, MXNet, Caffe