ON NEURAL ARCHITECTURES FOR SEGMENTATION IN NATURAL AND MEDICAL IMAGES

by

Qihang Yu

A dissertation submitted to Johns Hopkins University

in conformity with the requirements for the degree of

Doctor of Philosophy

Baltimore, Maryland

February, 2023

Abstract

Segmentation is an important research field in computer vision. It requires recognizing and segmenting the objects at the pixel level. In the past decade, many deep neural networks have been proposed, which have been central to the development in this area. These frameworks have demonstrated human-level or beyond performance on many challenging benchmarks, and have been widely used in many real-life applications, including surveillance, autonomous driving, and medical image analysis. However, it is non-trivial to design neural architectures with both efficiency and effectiveness, especially when they need to be tailored to the target tasks and datasets.

In this dissertation, I will present our research works in this area from the following aspects. (i) To enable automatic neural architecture design on the costly 3D medical image segmentation, we propose an efficient and effective neural architecture search algorithm that tackles the problem in a coarse-to-fine manner. (ii) To further take advantage of the neural architecture search, we propose to search for a channel-level replacement for 3D networks, which leads to strong alternatives to 3D networks. (iii) To perform segmentation with great detail, we design a coarse-to-fine segmentation framework for matting-level segmentation; (iv) To provide stronger features for segmentation, we propose a stronger transformer-based backbone that can work on dense tasks. (v) To better resolve the panoptic segmentation

problem in an end-to-end manner, we propose to combine transformers with the traditional clustering algorithm, which leads to a more intuitive segmentation framework with better performance.

Thesis Readers

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

Dr. Vishal Patel
Associate Professor
Electrical and Computer Engineering department
Johns Hopkins University

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

Acknowledgments

First and foremost, I want to express thanks to my advisor Prof. Alan Yuille for his patience, guidance, and support during my journey as a Ph.D. student. I began as a summer intern working with him. During the internship, I was deeply impressed by his great passion and insightful thoughts. It was this experience that made me decide to pursue computer vision research. I am more than lucky to have such a great mathematician and computer scientist as my advisor who guided me to be more than a researcher. Alan has been working hard to enrich the research collaboration and discussion in the lab, and he also has been helping us with our research and career developments. There is no doubt that I benefit a lot from his careful guidance. Next, I would like to thank Prof. Wei Shen for teaching me the fundamentals of computer vision and providing consistent support and guidance on my research on medical imaging and vision transformer. I thank Dr. Lingxi Xie for mentoring my projects and teaching me great research practices.

Besides, I would like to thank my thesis committee, Prof. Alan Yuille, Prof. Vishal Patal, and Prof. Wei Shen. I am also grateful to Prof. Rama Chellappa, Prof. Yinzhi Cao, Prof. Mick Bonner, and Prof. Rene Vidal for serving on my GBO committee and providing valuable suggestions. I would also like to thank Zachary Burwell, MaDonna Perry, and Kim Franklin for scheduling my GBO and helping on completing my degree requirements.

I am fortunate to have three great internships in NVIDIA, Adobe, and Google respectively. I am grateful for my received mentorship from Dong Yang, Holger Roth, Daguang Xu at NVIDIA, Jianming Zhang, He Zhang, Yilin Wang, Ning Xu, and Zhe Lin at Adobe, Liang-Chieh Chen, Yukun Zhu, Maxwell Collins, and Hartwig Adam at Google. They help me pursue a higher research quality, and also learn the differences between academic research and industry research.

During my Ph.D. journey, I am lucky to work with a group of talented and passionate people at CCVL. I appreciate the numerous discussions, suggestions, and collaborations. I would like to thank Lingxi, Wei, Yan, Adam, Yongyi, Weichao, Zhuotun, Chenxi, Zhishuai, Siyuan, Cihang, Yuyin, Qing, Chenxu, Huiyu, Qi, Yi, Yingda, Hongru, Fengze, Yingwei, Yixiao, Jieru, Zhuowan, Zihao, Chenglin, Yutong, Angtian, Chen, Jieneng, Ju, Shuhao, Yihong. Especially, I would like to thank everyone in FELIX project, including Yingda, Linda, Satomi, Seyoun, Fengze, Jieneng, Zhuotun, Yongyi, Yan, Wei, Lingxi, Yuyin, Kenneth, Bert, and Elliot.

Finally, I would like to acknowledge my parents, Xiaowei Yu and Hui Zeng, for their unconditioned love and support. I can never be here and pursue my research dream without their support and education.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Computer vision targets semantically understanding the input visual signals. The inputs can be in 2D or 3D, such as 2D natural images and 3D medical images. Various information is extracted from the inputs, and one of the most intuitive and informative outputs is the segmentation mask, which densely labels every pixel with its associated semantic class. This helps recognize and localize the objects in the complex scene and provides useful cues for further analysis and diagnosis.

In recent years, deep neural networks have been dominant in many computer vision tasks, including image classification [92], object detection [89], pose estimation [206], action recognition [221], *etc*. In the domain of segmentation tasks, since the pioneering works of FCN [153], many neural architectures have been proposed to further push the quality of segmentation masks. Nonetheless, the neural architectures usually need to be modified accordingly when applied to different tasks or datasets for optimal performance, which is usually non-trivial and requires experts' efforts. On the other hand, despite the

promising results achieved by current deep learning frameworks, there still lacks an end-to-end framework with an intuitive mechanism for complex segmentation problems, especially when it comes to panoptic segmentation [115], where heuristic post-processing and merging are usually used to obtain the final results.

In this dissertation, I aim to present our research for efficient and effective segmentation models. This dissertation consists of three parts. In part I, I will present our efforts in designing neural architecture search algorithms for efficient and effective 3D segmentation models. These algorithms aim to get rid of inconsistency problems between searching and deployment, which is even more serious when it comes to the 3D scenario that has a higher memory and computation demand. Besides, the found architectures not only show better accuracy-cost trade-offs but also show a better generalization ability. In part II, I will discuss our works incorporating the coarse-to-fine mechanism into the segmentation framework, which enables segmentation with refined details and fine-grained features. In part III, I will introduce our efforts in rethinking the mask transformer from a clustering perspective, where we unify the challenging panoptic segmentation problem and transformers as a process of iterative clustering assignment and updates. The proposed models in all three parts will be evaluated on supervised learning tasks including image classification and segmentation.

1.1 Neural Architecture Search for Efficient and Effective 3D Segmentation Models

Part I of this dissertation focuses on how to design neural architecture search algorithms for 3D segmentation, which enables automatic designs of 3D segmentation models and gets rid

of the heavy need for experts' involvement.

In Chapter 2, we will try to resolve the inconsistency problem during searching and deployment in neural architecture search, especially for 3D medical image segmentation. Though Neural Architecture Search (NAS) has shown promising results in finding networks with better accuracy-cost trade-off automatically, it also incurs a huge computation cost during the searching process. This is especially serious when it comes to the 3D data and dense prediction task, which further increases the computation and memory costs. In C2FNAS [258], we propose to disentangle the search space into topology-level and operation-level, and design searching methods tailored for each space respectively. By doing so, the search costs are significantly reduced, which is desired especially for 3D medical image segmentation. We further propose a topology prior and clustering-based method to further reduce the searching costs. Our contributions can be summarized into 3 folds: (1) we search a 3D segmentation network from scratch in a coarse-to-fine manner without sacrificing network size or input size; (2) we design the specific search space and search method for each stage based on medical image segmentation priors; (3) our model achieves state-of-the-art performance on 10 datasets from MSD challenge and shows great robustness and transfer-ability.

In Chapter 3, we will go beyond the topology and operation-level search. 3D Convolution Neural Networks (CNNs) have been widely applied to 3D scene understanding, such as video analysis and volumetric image recognition. However, 3D networks can easily lead to over-parameterization, which incurs expensive computation costs. In CAKES [251], we propose Channel-wise Automatic KErnel Shrinking (CAKES), to enable efficient 3D learning by shrinking standard 3D convolutions into a set of economic operations (*e.g.*,

1D, 2D convolutions). Unlike previous methods, CAKES performs channel-wise kernel shrinkage, which enjoys the following benefits: 1) enabling operations deployed in every layer to be heterogeneous so that they can extract diverse and complementary information to benefit the learning process; and 2) allowing for an efficient and flexible replacement design, which can be generalized to both spatial-temporal and volumetric data. As a result, CAKES shows superior performance to other methods with similar model sizes, and it also achieves comparable performance to state-of-the-art with much fewer parameters and computational costs on tasks including 3D medical imaging segmentation and video action recognition.

1.2 Coarse-to-Fine Framework for Accurate and Reliable Segmentation

Part II of this dissertation focuses on introducing coarse-to-fine mechanisms when designing segmentation frameworks, which improves the segmentation quality with refined details and leads to better features.

In Chapter 4, we study the problem of image matting, which is a fundamental computer vision problem which aims to predict an alpha matte to precisely cut out an image region. Recently, researchers start to study the matting problem in a trimap-free setting. One direction is to get rid of any external guidance, and hope that the matting model can capture both semantics and details by end-to-end training on large-scale datasets. Nevertheless, these methods are faced with the generalization challenge due to the lack of semantic guidance when tested on complex real-world images. In this work, we introduce a Mask Guided (MG) Matting [261] method which takes a general coarse mask as guidance. MG

Matting is very robust to the guidance input and can obtain high-quality matting results using various types of mask guidance such as a trimap, a rough binary segmentation mask or a low-quality soft alpha matte. To achieve such robustness to guidance input, we propose a Progressive Refinement Network (PRN) module, which learns to provide self-guidance to progressively refine the uncertain matting regions through the decoding process. To further enhance the robustness of our method to external guidance, we also develop a series of guidance mask perturbation operations including random binarization, random morphological operations, and also a stronger perturbation CutMask to simulate diverse guidance inputs during training. In addition to alpha matting prediction, we also revisit the foreground color prediction problem for matting. Without accurately recovering the foreground color in the transparent region, the composited image will suffer from the fringing issue. We note that the foreground color labels in the widely-used dataset DIM are suboptimal for model training due to the labeling noise and limited diversity. As a simple yet effective solution, we propose Random Alpha Blending (RAB) to generate synthetic training data from random alpha mattes and images. We show that such simple method can improve the foreground color prediction accuracy without requiring additional manual annotations. As a result, combining with the proposed PRN, MG Matting is able to generate more visual plausible composition results.

In Chapter 5, we propose an efficient and effective vision transformer backbone for extracting strong pixel-level representation given an input image. We propose Glance-and-Gaze Transformer (GG-Transformer [255]), inspired by the Glance-and-Gaze human behavior when recognizing objects in natural scenes [60], which takes advantage of both the long-range dependency modeling ability of Transformers and locality of convolutions

in a complementary manner. A GG-Transformer block consists of two parallel branches: A Glance branch performs self-attention within adaptively-dilated partitions of input images or feature maps, which preserves the global receptive field of the self-attention operation, meanwhile reduces its computation cost to a linear complexity as local window attention [149] does; A Gaze branch compensates locality to the features obtained by the Glance branch, which is implemented by a light-weight depth-wise convolutional layer. A merging operation finally re-arranges the points in each partition to their original locations, ensuring that the output of the GG-Transformer block has the same size as the input. As a proof-of-concept experiment, we compare original transformer encoder [214], Swin's transformer encoder [149], and the proposed Glance-and-Gaze transformer encoder with the same meta architecture on ImageNet. As a result, Glance-and-Gaze not only significantly outperforms Swin by 1.78%, but also even surpasses original transformer encoder by 0.49%, indicating that Glance-and-Gaze transformer encoder does serve as an efficient and effective alternative for building hierarchical vision transformer backbone. Our further experimental evaluations show consistent improvements in image classification on ImageNet[185] (+0.8% top-1 accuracy), object detection on MSCOCO[138] (+0.4% box AP), and semantic segmentation on ADE20K [277] (+1.9% mIoU).

1.3 Segmentation as Clustering through Mask Transformers

Part III of this dissertation focuses on rethinking the mask transformer from a clustering perspective, which provides a unified view of the end-to-end segmentation problem.

In Chapter 6, we propose a clustering perspective to understand and re-design the transformer decoder for grouping pixels into object-level representation. In CMT-DeepLab [252], which reformulates and further improves the previous end-to-end panoptic segmentation system [217] from the traditional clustering perspective. The panoptic segmentation result is naturally obtained by assigning each pixel to its most similar cluster center based on the feature affinity. In the Clustering Mask Transformer (CMT) module, the pixel features, cluster centers, and pixel-cluster assignments are updated in a manner similar to the clustering algorithms [151, 1].

In Chapter 7, we further simplify the transformer decoder and better align it to k-means clustering algorithm. In *k*MaX-DeepLab [254], we make a crucial observation that the cross-attention scheme actually bears a strong similarity to the traditional *k*-means clustering [151] by regarding the object queries as cluster centers with learnable embedding vectors. Our examination of the similarity inspires us to propose the novel **k**-means **Ma**sk **X**former (*k*MaX-DeepLab), which rethinks the relationship between pixel features and object queries, and redesigns the cross-attention from the perspective of *k*-means clustering. Specifically, when updating the cluster centers (*i.e.*, object queries), our *k*MaX-DeepLab performs a different operation. Instead of performing *softmax* on the large spatial dimension (image height times width) as in the original Mask Transformer's cross-attention [217], our *k*MaX-DeepLab performs *argmax* along the cluster center dimension, similar to the *k*-means pixel-cluster assignment step (with a *hard* assignment). We then update cluster centers by aggregating the pixel features based on the pixel-cluster assignment (computed by their feature affinity), similar to the *k*-means center-update step.

1.4 Relevant Publications

The following publications contribute to the main idea of this dissertation.

- Chapter 2 - **Qihang Yu**, Dong Yang, Holger Roth, Yutong Bai, Yixiao Zhang, Alan Yuille, Daguang Xu. C2FNAS: Coarse-to-Fine Neural Architecture Search for 3D Medical Image Segmentation, in CVPR 2020.
- Chapter 3 - **Qihang Yu**, Yingwei Li, Jieru Mei, Yuyin Zhou, Alan Yuille. CAKES: Channel-wise Automatic KErnel Shrinking for Efficient 3D Networks, in AAAI 2021.
- Chapter 4 - **Qihang Yu**, Jianming Zhang, He Zhang, Yilin Wang, Zhe Lin, Ning Xu, Yutong Bai, Alan Yuille. Mask Guided Matting via Progressive Refinement Network, in CVPR 2021.
- Chapter 5 - **Qihang Yu**, Yingda Xia, Yutong Bai, Yongyi Lu, Alan Yuille, Wei Shen. Glance-and-Gaze Vision Transformer, in NeurIPS 2021.
- Chapter 6 - **Qihang Yu**, Huiyu Wang, Dahun Kim, Siyuan Qiao, Maxwell Collins, Yukun Zhu, Hartwig Adam, Alan Yuille, Liang-Chieh Chen. CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation, in CVPR 2022.
- Chapter 7 - **Qihang Yu**, Huiyu Wang, Siyuan Qiao, Maxwell Collins, Yukun Zhu, Hartwig Adam, Alan Yuille, Liang-Chieh Chen. k-means Mask Transformer, in ECCV 2022.

The other related publications provide contexts for this dissertation:

- **Qihang Yu**, Lingxi Xie, Yan Wang, Yuyin Zhou, Elliot Fishman, Alan Yuille. Recurrent Saliency Transformation Network: Incorporating Multi-Stage Visual Cues for Small Organ Segmentation, in CVPR 2018.
- **Qihang Yu**, Yingda Xia, Lingxi Xie, Elliot Fishman, Alan Yuille. Thickened 2D Networks for Efficient 3D Medical Image Segmentation.
- Lingxi Xie, **Qihang Yu**, Yuyin Zhou, Yan Wang, Elliot Fishman, Alan Yuille. Recurrent Saliency Transformation Network for Tiny Target Segmentation in Abdominal CT Scans, in TMI.
- Yixiao Zhang, Xiaosong Wang, Ziyue Xu, **Qihang Yu**, Alan Yuille, Daguang Xu. When Radiology Report Generation Meets Knowledge Graph, in AAAI 2020.
- Yingwei Li, Xiaojie Jin, Jieru Mei, Xiaochen Lian, Linjie Yang, Cihang Xie, **Qihang Yu**, Yuyin Zhou, Song Bai, Alan Yuille. Neural Architecture Search for Lightweight Non-Local Networks, in CVPR 2020.
- Yingda Xia*, **Qihang Yu***, Wei Shen, Yuyin Zhou, Elliot Fishman, Alan Yuille. Detecting pancreatic ductal adenocarcinoma in multi-phase CT scans via alignment ensemble, in MICCAI 2020.
- Yingwei Li, **Qihang Yu**, Mingxing Tan, Jieru Mei, Peng Tang, Wei Shen, Alan Yuille, Cihang Xie. Shape-Texture Debiased Neural Network Training, in ICLR 2021.
- Jieneng Chen, Yongyi Lu, **Qihang Yu**, Xiangde Luo, Ehsan Adeli, Yan Wang, Le Lu, Alan Yuille, Yuyin Zhou. TransUNet: Transformers Make Strong Encoders for Medical Image Segmentation, in IMLH 2021.

- Dahun Kim, Jun Xie, Huiyu Wang, Siyuan Qiao, **Qihang Yu**, Hong-Seok Kim, Hartwig Adam, In So Kweon, Liang-Chieh Chen. TubeFormer-DeepLab: Video Mask Transformer, in CVPR 2022.

- Ju He, Shuo Yang, Shaokang Yang, Adam Kortylewski, Xiaoding Yuan, Jie-Neng Chen, Shuai Liu, Cheng Yang, **Qihang Yu**, Alan Yuille. PartImageNet: A Large, High-Quality Dataset of Parts, in ECCV 2022.

- Yingda Xia, **Qihang Yu**, Linda Chu, Satomi Kawamoto, Seyoun Park, Fengze Liu, Jieneng Chen, Zhuotun Zhu, Bowen Li, Zongwei Zhou, Yongyi Lu, Yan Wang, Wei Shen, Lingxi Xie, Yuyin Zhou, Christopher Wolfgang, Ammar Javed, Daniel Fadaei Fouladi, Shahab Shayesteh, Jefferson Graves, Alejandra Blanco, Eva S Zinreich, Benedict Kinny-Köster, Kenneth Kinzler, Ralph H Hruban, Bert Vogelstein, Alan L Yuille, Elliot K Fishman.

Chapter 2

C2FNAS: Coarse-to-Fine Neural Architecture Search for 3D Medical Image Segmentation

We start with segmentation for 3D medical imaging analysis. These works are part of the FELIX project [233], which was summarized as a system that works amazingly in early pancreatic cancer detection. This chapter introduces one of our efforts to automatic 3D neural architecture design during the FELIX project development stage. As it is usually difficult and time-consuming to design a specific model for 3D medical image segmentation, this work can help significantly reduce the search cost and save the times of trial-and-error process, while leading to a model with better efficiencies and effectiveness. Yet most existing NAS algorithms were designed for 2D natural images, and applying them to 3D scenarios could incur huge computational costs and a sacrifice is made leading to inconsistency during search and deployment. Here we try to disentangle the searching process which yields an algorithm that works better for searching 3D segmentation models.

2.1 Introduction

Medical image segmentation is an important pre-requisite of computer-aided diagnosis (CAD) which has been applied in a wide range of clinical applications. With the emerging of deep learning, great achievements have been made in this area. However, it remains very difficult to get satisfying segmentation for some challenging structures, which could be extremely small with respect to the whole volume, or vary a lot in terms of location, shape, and appearance. Besides, abnormalities, which result in a huge change in texture, and anisotropic property (different voxel spacing) make the segmentation tasks even harder. Some examples are shown in Fig 2.1.

Meanwhile, manually designing a high-performance 3D segmentation network requires adequate expertise. Most researchers are building upon existing 3D networks, such as 3D U-Net [52] and V-Net [162], with moderate modifications. In some cases, an individual network is designed and only works well for a certain task. To leverage this problem, Neural Architecture Search (NAS) technique is proposed in [290], which aims at automatically discovering better neural network architectures than human-designed ones in terms of performance, parameters amount, or computation cost. Starting from NASNet [291], many novel search spaces and search methods have been proposed [16, 75, 139, 141, 181]. However, only a few works apply NAS on medical image segmentation [112, 229, 285], and they only achieve a comparable performance versus those manually designed networks.

Inspired by the successful handcrafted architectures such as ResNet [92] and MobileNet [187], many NAS works focus on searching for network building blocks. However,

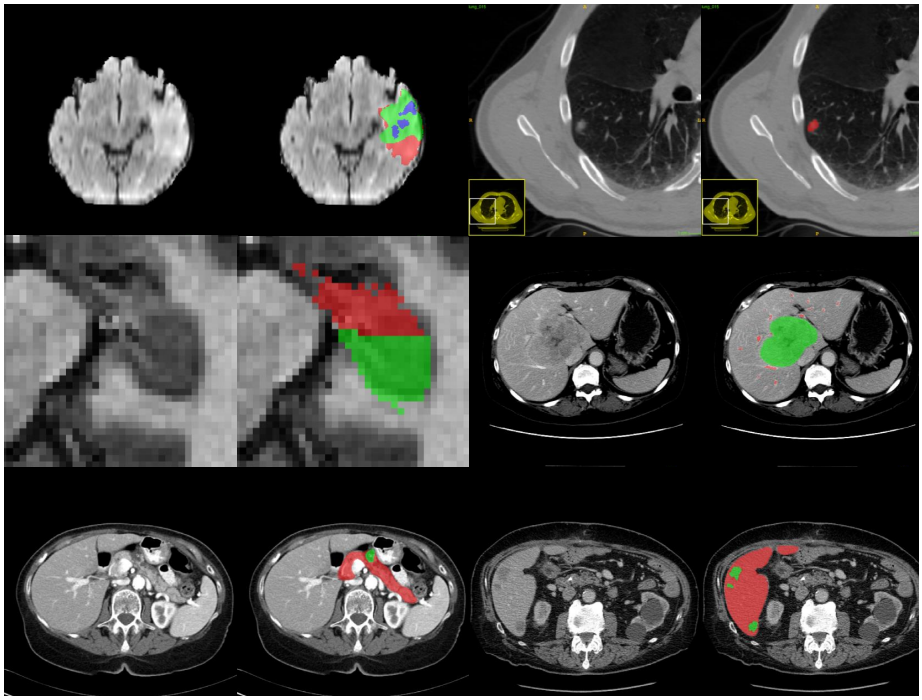

Figure 2.1: Image and mask examples from MSD tasks (from left to right and top to bottom): brain tumours, lung tumours, hippocampus, hepatic vessel and tumours, pancreas tumours, and liver tumours, respectively. The abnormalities, texture variance, and anisotropic properties make it very challenging to achieve satisfying segmentation performance. Red, green, and blue correspond to labels 1, 2, and 3, respectively, of each dataset.

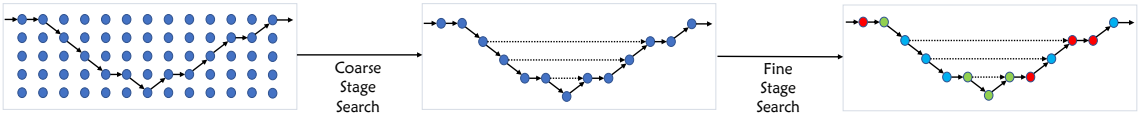

Figure 2.2: An illustration of proposed C2FNAS. Each path from the left-most node to the right-most node is a candidate architecture. Each color represents one category of operations, *e.g.* depthwise conv, dilated conv, or 2D/3D/P3D conv which are more common in medical image area. The dotted line indicates skip connections from the encoder to decoder. The macro-level topology is determined by coarse stage search, while the micro-level operations are further selected in fine stage search.

such works usually search in a shallow network while deploying with a deeper one. An inconsistency exists in network size between the search stage and deployment stage [38]. [18] and [82] avoided this problem through activating only one path at each iteration, and [38]

proposed to progressively reduce search space and enlarge the network in order to reduce the performance gap.

Nevertheless, when the network topology is involved in the search space, things become more complex because no inconsistency is allowed in network size. [139] incorporated the network topology into search space and relieved the memory tensity instead with a sacrifice on batch size and crop size. However, on memory-costly tasks such as 3D medical image segmentation, the memory scarcity cannot be solved by lowering the batch size or cropping size, since they are already very small compared to those of 2D tasks. Reducing them to a smaller number would lead to much worse performance and even failure on convergence.

To avoid the inconsistency on network size or input size between the search stage and deployment stage, we propose a coarse-to-fine neural architecture search scheme for 3D medical image segmentation (see Fig. 2.2). In detail, we divide the search procedure into the coarse stage and the fine stage. In the coarse stage, the search is in a small search space with limited network topologies, therefore searching in a train-from-scratch manner is affordable for each network. Moreover, to reduce the search space and make the search procedure more efficient, we constrain the search space under inspirations from successful medical segmentation network designs: (1) U-shape encoder-decoder structure; (2) Skip-connections between the down-sampling paths and the up-sampling paths. The search space is largely reduced with these two priors. Afterwards, we apply a topology-similarity-based evolutionary algorithm considering the search space properties, which makes the searching procedure focused on the promising architecture topologies. In the fine stage, the aim is to find the best operations inside each cell. Motivated by [285], we let the network itself choose the operation among 2D, 3D and pseudo-3D (P3D), so that it can capture features

from different viewpoints. Since the topology is already determined by coarse stage, we mitigate the memory pressure in single-path one-shot NAS manner [82].

For validation, we apply the proposed method on ten segmentation tasks from MSD challenge [195] and achieve state-of-the-art performance. The network is searched using the pancreas dataset which is one of the largest dataset among the 10 tasks. Our result on this proxy dataset surpasses the previous state-of-the-art by a large margin of 1% on pancreas and 2% on pancreas tumours. Then, we apply the same model and training/testing hyper-parameters across the other tasks, demonstrating the robustness and transferability of the searched network.

Our contributions can be summarized into 3 folds: (1) we search a 3D segmentation network from scratch in a coarse-to-fine manner without sacrificing network size or input size; (2) we design the specific search space and search method for each stage based on medical image segmentation priors; (3) our model achieves state-of-the-art performance on 10 datasets from MSD challenge and shows great robustness and transfer-ability.

2.2 Related Work

2.2.1 Medical Image Segmentation

Deep-learning-based methods have achieved great success in natural image recognition [92], detection [182], and segmentation [30], and they also have been dominating medical image segmentation tasks in recent years. Since U-Net was first introduced in biomedical image segmentation [183], several modifications have been proposed. [52] extended the 2D U-Net to a 3D version. Later, V-Net [162] is proposed to incorporate residual blocks and soft dice

loss. [167] introduced attention modules to reinforce the U-Net model. Researchers also tried to investigate other possible architectures despite U-Net. For example, [184, 257, 279] cut 3D volumes into 2D slices and handle them with 2D segmentation network. [146] designed a hybrid network by using ResNet50 as 2D encoder and appending 3D decoders afterwards. In [232], 2D predictions are fused by a 3D network to obtain a better prediction with contextual information.

However, until now, U-Net-based architectures are still the most powerful models in this area. Recently, [104] introduced nnU-Net and won the first place in Medical Segmentation Decalthon (MSD) Challenge [195]. They ensemble 2D U-Net, 3D U-Net, and cascaded 3D U-Net. The network is able to dynamically adapt itself to any given segmentation task by analysing the data attributes and adjusting hyper-parameters accordingly. The optimal results are achieved with different combinations of the aforementioned networks given various tasks.

2.2.2 Neural Architecture Search

Neural Architecture Search (NAS) aims at automatically discovering better neural network architectures than human-designed ones. At the beginning stage, most NAS algorithms are based on either reinforcement learning (RL) [9, 290, 291] or evolutionary algorithm (EA) [181, 236]. In RL-based methods, a controller is responsible for generating new architectures to train and evaluate, and the controller itself is trained with the architecture accuracy on the validation set as rewards. In EA-based methods, architectures are mutated to produce better off-springs, which are also evaluated by accuracy on the validation set. Since the parameter sharing scheme was proposed in [171], more search methods were

proposed, such as differentiable NAS approaches [141] and one-shot NAS approaches [16], which reduced the search cost to several GPU days or even several GPU hours.

Besides the successes NAS has achieved in natural image recognition, researchers also tried to extend it to other areas such as segmentation [139], detection [75], and attention mechanism [129]. Moreover, there are also some works applying NAS to the medical image segmentation area. [285] designed a search space consisting of 2D, 3D, and pseudo-3D (P3D) operations, and let the network itself choose between these operations at each layer. [163, 246] use the policy gradient algorithm for automatically tuning the hyper-parameters and data augmentations. In [112, 229], the cell structure is explored with a pre-defined 3D U-Net topology.

2.3 Coarse-to-Fine Neural Architecture Search

2.3.1 Inconsistency Problem

Early works of NAS [9, 181, 236, 290, 291] typically use a controller based on EA or RL to select network candidates from search space; then the selected architecture is trained and evaluated. Such methods need to train numerous models from scratch and thus lead to an expensive search cost. Recent works [16, 141] propose a differentiable search method that reduces the search cost significantly, where each network is treated as a sub-network of a super-network. However, a critical problem is that the super-network cannot fit into the memory. For these methods, a trade-off is made by sacrificing the network size at the search stage and building a deeper one at deployment, which results in an inconsistency problem. [18] proposed to activate a single path of the super-network at each iteration to

reduce the memory cost, and [38] proposed to progressively increase the network size with a reduced approximate search space. However, these methods also face problems when the network topology is included in the search. For instance, the progressive manner cannot deal with the network topology. As for single-path methods, since there exist illegal paths in network topology, some layers are naturally trained more times compared to others, which results in a serious fairness problem [50].

A straightforward way to solve the issue is to train each candidate from scratch respectively, yet the search cost is too expensive considering the magnitude of the search space, which may contain millions of candidates or more. Auto-DeepLab [139] introduces network topology into search space and sacrifices the input size instead of network size at the training stage, where it uses a much smaller batch size and crop size. However, it introduces a new inconsistency at input size to solve the old one at network size. Besides, for memory-costly tasks such as 3D medical image segmentation, sacrificing input size is infeasible. The already small input size needs to be reduced to unreasonably smaller to fit the model in memory, which usually leads to an unstable training problem in terms of convergence, and the method only yields a random architecture finally.

2.3.2 Coarse-to-fine Neural Architecture Search

In order to resolve the inconsistency in network size and input size, and combine NAS with medical image segmentation, we develop a coarse-to-fine neural architecture search method for automatically designing 3D segmentation networks. Without loss of generality, the architecture search space $\mathcal{A}$ consists of topology search space $\mathcal{S}$, which is represented by a directed acyclic graph (DAG), and cell operation space $\mathcal{C}$, which is represented by the

color of each node in the DAG. Each network candidate is a sub-graph $s \in \mathcal{S}$ with color scheme $c \in \mathcal{C}$ and weights w, denoted as $\mathcal{N}(s, c, w)$.

Therefore, the search space $\mathcal{A}$ is divided into two parts: a small search space of topology $\mathcal{S}$, and a huge search space of operation $\mathcal{C}$:

$$\mathcal{A} = \mathcal{S} \times \mathcal{C}. \tag{2.1}$$

The topology search space is usually small and it is affordable to handle the inconsistency by training each candidate from scratch. For instance, the topology search space $\mathcal{S}$ only has up to 2.9×10^4 candidates for a network with 12 cells [139]. The operation search space $\mathcal{C}$ can have millions of candidates, but since topology s is given, techniques in NAS for recognition, *e.g.* activating only one path at each iteration, are incorporated naturally to solve the memory limitation. Therefore, by regarding neural architecture search from scratch as a process of constructing a colored DAG, we divide the search procedure into two stages: (1) Coarse stage: search at the macro-level for the network topology, and (2) Fine stage: search for the best way to color each node, *i.e.* finding the most suitable operation configuration.

We start with defining the macro-level and micro-level. Each network consists of multiple cells, which are composed of several convolutional layers. On the macro-level, by defining how every cell is connected to each other, the network topology is uniquely determined. Once the topology is determined, we need to define which operation each node represents. On the micro-level, we assign an operation to each node, which represents the operation inside the cell, such as standard convolution or dilated convolution.

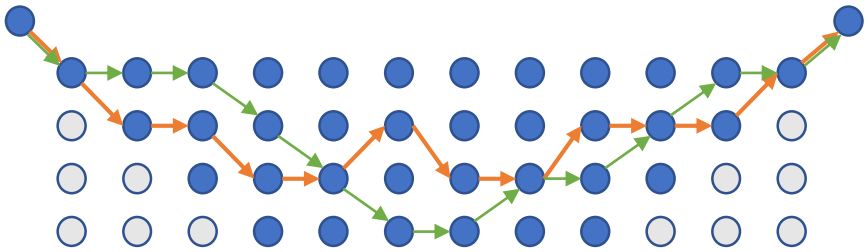

Figure 2.3: An example of how introduced priors help reduce search space. The grey nodes are eliminated entirely from the graph. Besides, many illegal paths have been pruned off as well. An example of an illegal path and a legal path is shown as the orange line path and green line path separately.

With this two-stage procedure, we first construct a DAG representing network topology, then assign operations to each cell by coloring the corresponding node in the graph. Therefore, a network is constructed from scratch in a coarse-to-fine manner. By separating the macro-level and micro-level, we relieve the memory pressure and thus resolve the inconsistency problem between the search stage and deployment stage.

2.3.3 Coarse Stage: Macro-level Search

In this stage, we mainly focus on searching the topology of the network. A default operation is assigned to each cell, specifically standard 3D convolution in this chapter, and the cell is used as the basic unit to construct the network.

Due to memory constraints and fairness problems, training a super-network and evaluating candidates with a weight-sharing method is infeasible, which means each network needs to be trained from scratch. The search on the macro-level is formulated into a bi-level

optimization with weight optimization and topology optimization:

$$w_s = \arg\min_{w} \mathcal{L}_{\text{train}}(\mathcal{N}(s, c_0, w)), \tag{2.2}$$

$$s^* = \arg\max_{s \in \mathcal{S}} \text{Acc}_{\text{val}}(\mathcal{N}(s, c_0, w_s)), \tag{2.3}$$

where s represents the current topology and c_0 denotes a default coloring scheme, *e.g.* standard 3D convolution everywhere, and $\mathcal{L}_{\text{train}}$ is the loss function used at the training stage, and Acc_{val} the accuracy on the validation set.

It is extremely time-consuming, especially considering that 3D networks have heavier computation requirements compared with 2D models. Thus, it is necessary to reduce the search space to make the search procedure more focused and efficient.

We revisit the successful medical image segmentation networks, and we find they all share something in common: (1) a U-shape encoder-decoder topology and (2) skip-connections between the down-sampling paths and the up-sampling paths. We incorporate these priors into our method and prune the search space accordingly. An illustration of how the priors help prune search space is shown in Fig. 2.3. Therefore, the search space $\mathcal{S}$ is pruned to $\mathcal{S}'$ and the topology optimization becomes:

$$\mathcal{S}' = \text{PriorPrune}(\mathcal{S}), \tag{2.4}$$

$$s^* = \arg\max_{s \in \mathcal{S}'} \text{Acc}_{\text{val}}(\mathcal{N}(s, c_0, w_s)). \tag{2.5}$$

To further improve the search efficiency, we propose an evolutionary algorithm based on topology similarity to make use of macro-level properties. The idea is that with an assumption of continuous relaxation of topology search space, two similar networks should

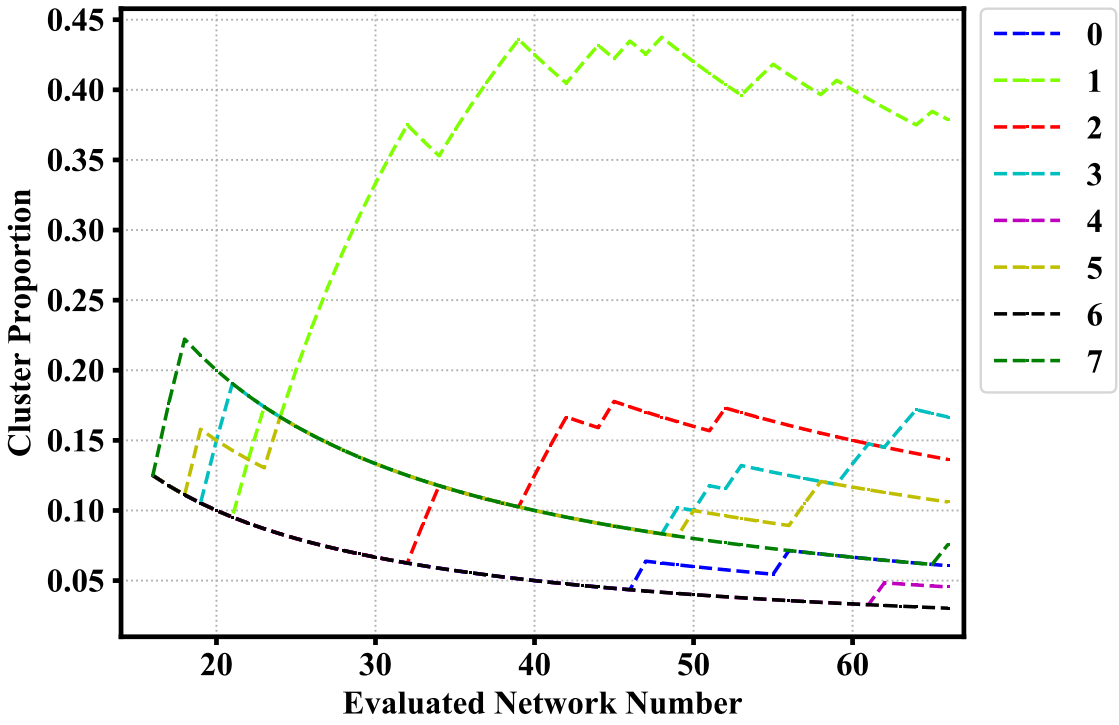

Figure 2.4: Proportion of clusters sampled during searching at the coarse stage. This figure illustrates the effectiveness of the proposed evolutionary searching algorithm. Different clusters are in different colors. The x-axis label "Evaluated Network Number" means the total number of networks trained and evaluated, while the y-axis label "Cluster Proportion" is the proportion of the number of networks belonging to a specific cluster to the total number of evaluated networks. It is shown that the algorithm gradually focuses on the most promising cluster 1, making the search procedure more efficient.

also share a similar performance. Specifically, we represent each network topology with a code, and we define the network similarity as the euclidean distance between two codes. The smaller the distance is, the more similar the two networks are. Based on the distance measurement, we classify all network candidates into several clusters with K-means algorithm [151] based on their encoded codes. The evolution procedure is prompted in the unit of the cluster. In detail, when producing the next generation, we random sample some

Algorithm 1 Topology Similarity based Evolution

```
population ← all topologies
𝒫 = {p_1, p_2, ..., p_k} ← Cluster(population)
history ℋ ← ∅
set of trained models ℳ = {m_1, m_2, ..., m_k} ← {∅}^k
for i = 1 to k do
    model.topology ← RandomSample(p_i)
    model.accuracy ← TrainEval(model.topology)
    add model to ℋ and m_i
while |ℋ| ≤ l do
    while HasIdleGPU() do
        model for compare 𝒟 ← ∅
        for i = 1 to k do
            add RandomSample(m_i) to 𝒟
        rank 𝒫 based on corresponding accuracy in 𝒟
        model.topology ← SampleUntrained(p_rank1)
        model.accuracy ← TrainEval(model.topology)
        add model to ℋ and m_rank1
return highest-accuracy model in ℋ
```

networks from each cluster, and rank the clusters by comparing the performance of these networks. The higher rank of a cluster is, the higher proportion of the next generation will come from this cluster. As shown in Fig. 2.4, the topology proposed by our algorithm gradually falls into the most promising cluster, demonstrating its effectiveness. To better make use of computation resources, we further implement this EA algorithm in an asynchronous manner as shown in Algorithm 1.

2.3.4 Fine Stage: Micro-level Search

After the topology of the network is determined, we further search the model at a fine-grained level by replacing the operations inside each cell. Each cell is a small fully convolutional module, which takes 1 or 2 input tensors and outputs 1 tensor. Since the

topology is pre-determined in the coarse stage, cell i is simply represented by its operations O_i, which is a subset of the possible operation set $\mathcal{O}$. Our cell structure is much simpler compared with [139], this is because there is a trade-off between the cell complexity and cell numbers. Given the tense memory requirement of 3D models, we prefer more cells instead of a more complex cell structure.

The set of possible operations, $\mathcal{O}$, consisting of the following 3 choices: (1) $3 \times 3 \times 3$ 3D convolution; (2) $3 \times 3 \times 1$ followed by $1 \times 1 \times 3$ P3D convolution; (3) $3 \times 3 \times 1$ 2D convolution;

Considering the magnitude of fine stage search space, training each candidate from scratch is infeasible. Therefore, to address the problem of memory limitation while making search efficient, we adopt single-path one-shot NAS with uniformly sampling [82] as our search method. In detail, we construct a super-network where each candidate is a sub-network of it, and then at each iteration of the training procedure, a candidate is uniformly sampled from the super-network and trained and updated. After the training procedure ends, we perform a random search for the final operation configuration. That is to say, at the searching stage, we random sample K candidates, and each candidate is initialized with the weights from the trained super-network. All these candidates are ranked by validation performance, and the one with the highest accuracy is finally picked.

Therefore, optimization of the fine stage is in single-path one-shot NAS manner with uniform sampling, which is formulated as:

$$w = \arg\min_{w} \mathbb{E}_{c \in \mathcal{C}}[\mathcal{L}_{\text{train}}(\mathcal{S}(s^*, c, w))], \tag{2.6}$$

$$c^* = \arg\max_{c} \text{Acc}_{\text{val}}(\mathcal{S}(s^*, c, w)), \tag{2.7}$$

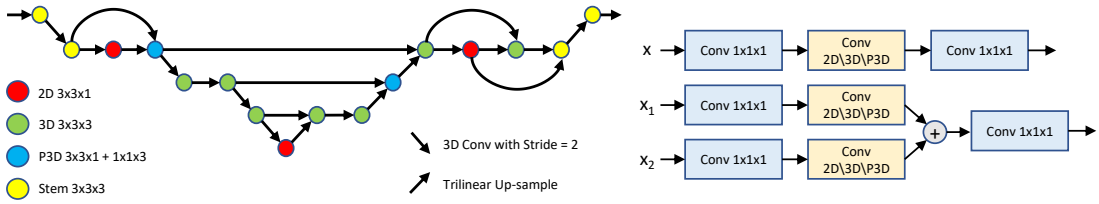

Figure 2.5: Left: The final architecture of C2FNAS-Panc. Red, green, and blue denote cell with 2D, 3D, P3D operations separately. **Right:** The structure of a cell with single input and two inputs.

where $\mathcal{C}$ is the search space of fine stage, *i.e.* all possibles combinations of operations.

After the coarse stage is finished, the topology s^* is obtained. And the operation configuration c^* comes from the fine stage. Therefore, the final network architecture $\mathcal{N}(s^*, c^*, w)$ is constructed.

2.4 Experiments

In this section, we firstly introduce our implementation details of C2FNAS, and then report our found architecture (searched on MSD Pancreas dataset) with semantic segmentation results on all 10 MSD datasets [195], which is a public comprehensive benchmark for general-purpose algorithmic validation and testing covering a large span of challenges, such as small data, unbalanced labels, large-ranging object scales, multi-class labels, and multi-modal imaging, *etc*. It contains 10 segmentation datasets, *i.e.* Brain Tumours, Cardiac, Liver Tumours, Hippocampus, Prostate, Lung Tumours, Pancreas Tumours, Hepatic Vessels, Spleen, and Colon Cancer.

2.4.1 Implementation Details

Coarse Stage Search At coarse stage search, the network has 12 cells in total, where 3 of them are down-sampling cells and 3 up-sampling cells, so the model size is moderate. With the priors introduced in Section 4.3, the search space is largely reduced from 2.9×10^4 to 9.24×10^2.

For network architecture, we define one stem module at the beginning of the network, and another one at the end. The beginning module consists of two 3D $3 \times 3 \times 3$ convolution layers, and strides are 1, 2 respectively. The end module consists of two 3D $3 \times 3 \times 3$ convolution layers, and a trilinear up-sampling layer between the two layers. Each cell takes the output of its previous cell as input, and it will also take another input if it satisfies (1) it has a previous-previous cell at the same feature resolution level, or (2) it is the first cell after an up-sampling. In situation (1), the cell takes its previous-previous cell's output as additional input. And in the situation (2), it takes the output of the last cell before the corresponding down-sampling as another input, which serves as the skip-connection from the encoder part to the decoder part. A convolution with kernel size $1 \times 1 \times 1$ serves as pre-processing for the input. The two inputs go through convolution separately and get summed afterwards, then a $1 \times 1 \times 1$ convolution is applied to the output. The filter number starts with 32, and it is doubled after a down-sampling layer and halved after an up-sampling layer. All down-sampling operations are implemented by a $3 \times 3 \times 3$ 3D convolution with stride 2, and up-sampling by a trilinear interpolation with scale factor 2 followed by a $1 \times 1 \times 1$ convolution. Besides, in the coarse stage, we also set the operations in all cells to standard 3D convolution with a kernel size of $3 \times 3 \times 3$.

For the evolutionary algorithm part, we first represent each network topology with a code, which is a list of numbers and the length is the same as cell numbers. The number starts at 0 and increases one after a down-sampling and decreases one after an up-sampling. We use the K-means algorithm to classify all candidates into 8 clusters based on the Euclidean metric of corresponding codes. In the beginning, two networks are randomly sampled from each cluster. Afterwards, whenever there is an idle GPU, one trained network is sampled from each cluster, and the cluster to which the best network belongs is picked and a new network is sampled from that cluster for training. Meanwhile, the algorithm also random samples a cluster with the probability of 0.2 to add randomness and avoid local minimum. After 50 networks are evaluated, the algorithm terminates and returns the best network topology it has found.

We conduct the coarse stage search on the MSD Pancreas Tumours dataset, which contains 282 3D volumes for training and 139 for testing. The dataset is labelled with both pancreatic tumours and normal pancreas regions. We divide the training data into 5 folds sequentially, where the first 4 folds are for training and the last fold is for validation purposes. To address the anisotropic problem, we re-sample all cases to an isotropic resolution with a voxel distance of 1.0 mm for each axis as data pre-processing.

At the training stage, we use batch size of 8 with 8 GPUs, and patch size of $[96, 96, 96]$, where two patches are randomly cropped from each volume at each iteration. All patches are randomly rotated by $[0^\circ, 90^\circ, 180^\circ, 270^\circ]$ and flipped as data augmentation. We use SGD optimizer with a learning rate of 0.02, momentum of 0.9, and weight decay of 0.00004. Besides, there is a multi-step learning rate schedule which decays the learning rate at iterations $[8000, 16000]$ with a factor of 0.5. We use 1000 iterations for the warm-up

Task	Brain				Liver			Pancreas			Prostate		
Class	1	2	3	Avg	1	2	Avg	1	2	Avg	1	2	Avg
CerebriuDIKU [170]	**69.52**	43.11	66.74	59.79	94.27	57.25	75.76	71.23	24.98	48.11	69.11	86.34	77.73
Lupin	66.15	41.63	64.15	57.31	94.79	61.40	78.10	75.99	21.24	48.62	72.73	87.62	80.18
NVDLMED [231]	67.52	45.00	68.01	60.18	95.06	71.40	83.23	78.42	38.48	58.45	69.36	86.66	78.01
K.A.V.athlon	66.63	46.62	67.46	60.24	94.74	61.65	78.20	74.97	43.20	59.09	73.42	87.80	80.61
nnU-Net [104]	67.71	47.73	68.16	61.20	**95.24**	**73.71**	**84.48**	79.53	52.27	65.90	**75.81**	**89.59**	**82.70**
C2FNAS-Panc	67.62	48.56	69.09	61.76	94.91	71.63	83.27	80.59	52.87	66.73	73.11	87.43	80.27
C2FNAS-Panc*	67.62	**48.60**	**69.72**	**61.98**	94.98	72.89	83.94	**80.76**	**54.41**	**67.59**	74.88	88.75	81.82

Task	Lung	Heart	Hippocampus			HepaticVessel			Spleen	Colon	Avg (Task)	Avg (Class)
Class	1	1	1	2	Avg	1	2	Avg	1	1		
CerebriuDIKU [170]	58.71	89.47	89.68	88.31	89.00	59.00	38.00	48.50	95.00	28.00	67.01	66.40
Lupin	54.61	91.86	89.66	88.26	88.96	60.00	47.00	53.50	94.00	9.00	65.61	65.89
NVDLMED [231]	52.15	92.46	87.97	86.71	87.34	63.00	64.00	63.50	96.00	56.00	72.73	71.66
K.A.V.athlon	60.56	91.72	89.83	88.52	89.18	62.00	63.00	62.50	**97.00**	36.00	71.51	70.89
nnU-Net [104]	69.20	**92.77**	**90.37**	**88.95**	**89.66**	63.00	69.00	66.00	96.00	56.00	76.39	75.00
C2FNAS-Panc	69.47	92.13	86.87	85.44	86.16	63.78	69.41	66.60	96.60	55.68	75.87	74.42
C2FNAS-Panc*	**70.44**	92.49	89.37	87.96	88.67	**64.30**	**71.00**	**67.65**	96.28	**58.90**	**76.97**	**75.49**

Table 2.1: Comparison with state-of-the-art methods on MSD challenge test set (number from MSD leaderboard) measured by **Dice-Sørensen coefficient (DSC)**. * denotes the 5-fold model ensemble. The numbers of tasks hepatic vessel, spleen, and colon from other teams are rounded. We also report the average on tasks and on targets respectively for an overall comparison across all tasks/targets.

stage, where the learning rate increases linearly from 0.0025 to 0.02, and 20000 iterations for training. The loss function is the summation of Dice Loss and Cross-Entropy Loss, and we adopt Instance Normalization [212] and ReLU activation function. We also use Horovod [189] to speed up the multi-GPU training procedure.

At the validation stage, we test the network in a sliding window manner, where the stride = 16 for all axes. Dice-Sørensen coefficient (DSC) metric is used to measure the performance, which is formulated as $\text{DSC}\ (\mathcal{Y}, \mathcal{Z}) = \frac{2 \times |\mathcal{Y} \cap \mathcal{Z}|}{|\mathcal{Y}| + |\mathcal{Z}|}$, where $\mathcal{Y}$ and $\mathcal{Z}$ denote for the prediction and ground-truth voxels set for a foreground class. The DSC has a range of $[0, 1]$ with 1 implying a perfect prediction.

Fine Stage Search In the fine stage search, we mainly choose the operations from $[\text{2D, 3D, P3D}]$ for each cell. This search space can be large as 5.3×10^5. Since the search space is numerous, we adopt a single-path one-shot NAS method based on super-network,

Model	Params (M)	FLOPs (G)
3D U-Net [52]	19.07	825.30
V-Net [162]	45.59	301.88
VoxResNet [23]	6.92	173.02
ResDSN [287]	10.03	188.37
Attention U-Net [167]	103.88	1162.75
C2FNAS-Panc	17.02	150.78

Table 2.2: Comparison of parameters and FLOPs with other 3D networks. The FLOPs are calculated based on input size $96 \times 96 \times 96$.

which is trained by uniform sampling.

The data pre-processing, data split, and training/validation setting are exactly the same as what we use in the coarse stage, except that we double the number of iterations to ensure the super-network convergence. At each iteration, a random path is chosen for training. After the super-network training is finished, we random sample 2000 candidates from the search space, and use the super-network weight to initialize these candidates. Since the validation process takes a very long time due to the sliding window method, we increase the stride to 48 at all axes to speed up the search stage.

The coarse search stage takes 5 days with 64 NVIDIA V100 GPUs with 16GB memory. In the fine stage, the super-network training costs 10 hours with 8 GPUs, and the searching procedure, where 2000 candidates are evaluated on the validation set, takes 1 day with 8 GPUs. The large search cost is mainly because training and evaluating a 3D model itself is very time-consuming.

Deployment Stage The final network architecture based on the topology searched in the coarse stage and operations searched in the fine stage is shown in Fig. 2.5. We keep the training setting same when deploying this network architecture, which means no inconsistency exists in our method.

Task	Lung	Pancreas		
Class	1	1	2	Avg
C2FNAS-C-Lung	**71.74**	80.26	52.51	66.39
C2FNAS-C-Panc	69.05	**80.39**	53.32	66.86
C2FNAS-F-Panc	69.77	80.37	**56.36**	**68.37**

Table 2.3: Comparison with different stages and different proxy datasets on 5-fold cross-validation.

We use the same training setting mentioned in the coarse stage, and the iteration is 40000 and multi-step decay at iterations $[16000, 32000]$. The model is trained based on the same settings from scratch for each dataset, except that Prostate dataset has a very small size on the Z (Axial) axis, and Hippocampus dataset has a very small shape around only 50 for each axis. Therefore we change the patch size to $128 \times 128 \times 32$ and stride = $[16, 16, 4]$ for the Prostate, and up-sample all data to shape $96 \times 96 \times 96$ for Hippocampus.

2.4.2 Segmentation Results

We report our test set results of all 10 tasks from the MSD challenge and compare C2FNAS with other state-of-the-art methods.

Our test set results are summarized in Table 2.1. We notice that other methods apply multi-model ensemble to reinforce the performance, *e.g.* nnU-Net ensembles 5 or 10 models based on 5-fold cross-validation with one or two models, NVDLMED and CerebriuDIKU ensemble models trained from different viewpoints. Therefore, besides single-model results, we also report results with a 5-fold cross-validation model ensemble, which means 5 models are trained in a 5-fold cross-validation setting, and final test results are fused with results from these 5 models with a majority voting.

Our model shows superior performance than state-of-the-art methods on most tasks,

especially the challenging ones, while enjoying a lighter model size compared to most popular 3D models (see Table 2.2). We also have a higher performance in terms of average on task/class. It is noticeable that the previous state-of-the-art nnU-Net uses various kinds of data augmentation and test-time augmentation to boost the performance, while we only adopt simple data augmentation of rotation and flip, and no test-time augmentation is applied. Small datasets such as Heart and Hippocampus rely more on augmentation while a powerful architecture is easy to get over-fitting, which illustrates why our performance on these datasets does not outperform the competitors. Besides, nnU-Net uses different networks and hyper-parameters for each task, while we use the same model and hyper-parameters for all tasks, showing that our model is not only more powerful but also much more robust and generalizable. Some visualization comparisons are available in Fig. 2.6.

2.5 Ablation Study

2.5.1 Coarse Stage versus Fine Stage

To verify the improvement of this two-stage design, we compare the performance of the network from coarse stage and the network from fine stage. The “C2FNAS-C-Panc" indicates the coarse stage network searched on the pancreas dataset, where the topology is searched and all operations are in a standard 3D manner, while “C2FNAS-F-Panc" is the fine stage network, where the operation configuration is searched. We compare their performance on the pancreas and lung datasets with a 5-fold cross-validation. The result is shown in table 2.3. It is noticeable that the fine stage search not only improves the performance on the target dataset (pancreas) but also increases the model generality, thus

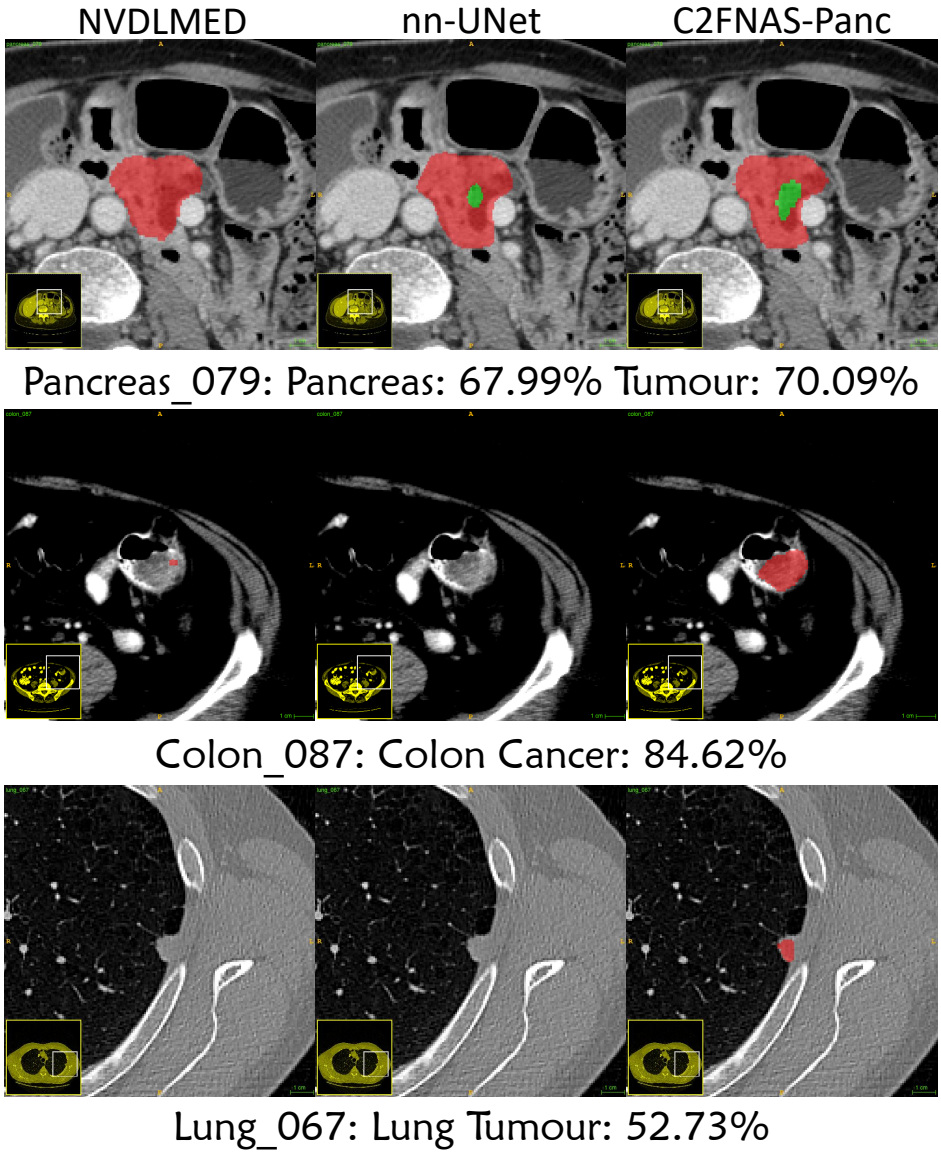

Figure 2.6: The visualization comparison between state-of-the-art methods (1st and 2nd teams) and C2FNAS-Panc on MSD test sets. We visualize one case from each of the three most challenging tasks: pancreas and pancreas tumours, colon cancer, and lung tumours. Red denotes abnormal pancreas, colon cancer, and lung tumours respectively, and green denotes pancreas tumours. The case id and dice score of C2FNAS-Panc are at the bottom.

obtaining a better performance on other datasets (lung).

2.5.2 Search on Different Datasets

Our model is searched on MSD Pancreas dataset, which contains 282 cases, and it is one of the largest datasets in MSD challenge. To verify the data number effect on our method, we also search a model topology on MSD Lung dataset, which contains 64 cases, as an ablation study. The search method and hyper-parameters are the same as what we use on the pancreas dataset. The result is summarized in Table 2.3. The “C2FNAS-C-Lung" is the topology searched on the lung dataset, while “C2FNAS-C-Panc" is the topology searched on the pancreas dataset. Topology on lung dataset performs better on lung task, while topology on pancreas dataset performs better on pancreas task. However, it is noticeable that both topologies show good performance on another dataset, demonstrating that our method works well even on a smaller dataset and the models are of great generality.

2.5.3 Incorporate Model Scaling as Third Stage

Inspired by EfficientNet [202], we add model scaling into the search space as the third search stage. In this ablation study, we only study for scaling of filter numbers for simplicity, but a compound scaling including patch size and cell numbers is feasible. Following [202], we adopt a grid search for a channel number multiplier ranging from 0.25 to 2.0 with a step of 0.25. We report the results based on single fold validation set on the pancreas and lung datasets respectively, which are summarized in Table 2.4. It shows that model scaling can increase the model capacity and lead to better performance. Nevertheless, scaling up the model also results in much higher model parameters and FLOPs. Considering the

Task	Lung	Pancreas			Hippocampus		
Class	1	1	2	Avg	1	2	Avg
0.25	72.32	79.24	40.02	59.63	80.29	79.81	80.05
0.50	73.89	80.51	46.34	63.43	80.74	80.84	80.79
0.75	76.15	81.40	47.50	64.45	80.88	81.72	81.30
1.00	74.26	80.74	49.94	65.34	81.82	82.10	81.96
1.25	76.94	81.45	48.03	64.74	82.13	82.24	82.19
1.50	75.37	81.40	48.87	65.14	81.02	81.39	81.21
1.75	75.98	81.85	49.03	65.44	81.52	81.31	81.42
2.00	**77.75**	**82.18**	**50.61**	**66.40**	**82.57**	**82.34**	**82.46**

Table 2.4: Influence of model scaling, the number in the first column indicates the scale factor applied to model C2FNAS-Panc. The results are based on single fold of validation set and the final searched model on pancreas dataset.

large extra computation cost and to keep the model in a moderate size, we do not include model scaling in our main experiment. Yet we report it in ablation study as a potential and promising way to reinforce C2FNAS and achieve even higher performance.

2.6 Conclusions

In this chapter, we propose to use coarse-to-fine neural architecture search to automatically design a transferable 3D segmentation network for 3D medical image segmentation, where the existing NAS methods cannot work well due to the memory-consuming property in 3D segmentation. Besides, our method, with the consistent model and hyper-parameters for all tasks, outperforms MSD champion nnU-Net, a series of well-modified and/or ensembled 2D and 3D U-Net. We do not incorporate any attention module or pyramid module, which means this is a much more powerful 3D backbone model than current popular network architectures.

Chapter 3

CAKES: Channel-wise Automatic KErnel Shrinking for Efficient 3D Networks

In this chapter, we keep studying the topics of using Neural Architecture Search to automatically obtain better 3D medical segmentation models. Unlike C2FNAS which focus more on the searching process, here we focus more on a novel search space tailored for 3D CNN architectures, which not only brings more possibilities during the search process but also ensure the final model specializes in the 3D learning, especially the segmentation tasks. Specifically, we consider replacing 3D convolution with its efficient alternative in a channel-wise manner, which leads to a more fine-grained architecture design and more promising results.

3.1 Introduction

3D learning has attracted more and more research attention with the recent advance of deep neural networks. However, 3D convolution layers typically result in expensive computation

and suffer from convergence problems due to over-fitting issues and the lack of pre-trained weights [22, 201].

To resolve the redundancy in 3D convolutions, many efforts have been investigated to design efficient alternatives. For instance, [177] and [210] propose to factorize the 3D kernel and replace the 3D convolution with Pseudo-3D (P3D) and (2+1)D convolution, where 2D and 1D convolution layers are applied in a structured manner. [238] suggests that replacing 3D convolutions with low-cost 2D convolutions at the bottom of the network significantly improves recognition efficiency.

Despite their effectiveness for spatial-temporal information extraction, there are several limitations of existing alternatives to 3D convolutions. Firstly, these methods (*e.g.*, P3D) are specifically tailored to video datasets, where data can be explicitly separated into time and space. However, for volumetric data such as CT/MRI where all three dimensions should be treated equally, conventional spatial-temporal operators can lead to biased information extraction. Moreover, existing operations are still insufficient even for spatial-temporal data since they may exhibit certain levels of redundancy either along the temporal or the spatial dimension, as empirically suggested in [238]. Secondly, existing replacements are manually designed. Consequently, this process can be time-consuming and may lead to sub-optimal results.

To address these issues, we introduce *Channel-wise Automatic KErnel Shrinking (CAKES)*, as a general efficient alternative to existing 3D operations. Specifically, the proposed method simplifies conventional 3D operations by adopting a combination of diverse and economic operations (*e.g.*, 1D, 2D convolutions), where these different operators can extract complementary information to be utilized in the same layer. Our approach is not

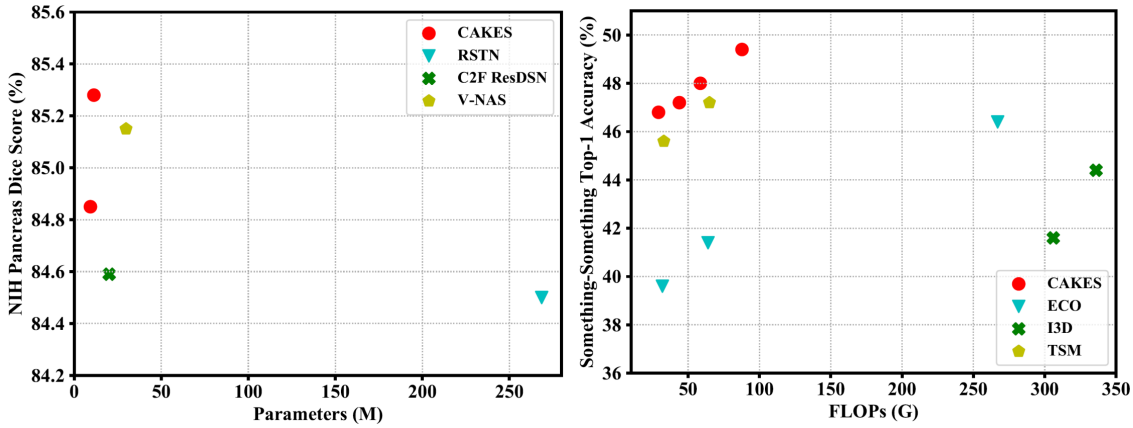

Figure 3.1: CAKES shows better accuracy-cost trade-off on both 3D medical image segmentation (left) and action recognition (right) tasks.

tailored to any specific type of input (*e.g.*, videos), but can be generalized to different types of data and backbone architectures to achieve a fine-grained and efficient replacement.

As a proof test, our CAKES with a naive manual setting already exhibits superior performances compared with existing 3D replacements (Table 3.1 & 3.3). However, the manual selection of the set of replacing operators as well as their positioning requires trial-and-error. To further improve the performance and the model efficiency, we introduce a new search space consisting of computationally-efficient candidate operators, to facilitate the search for the optimal replacement configuration given a backbone architecture. With our search space design, the proposed CAKES is feasible to obtain a good architecture in several GPU days.

The proposed algorithm delivers high-performance and efficient models. As shown in Fig. 3.1, evaluated on both 3D medical image segmentation and video action recognition tasks, our method achieves a better accuracy-cost trade-off. Compared with its 3D baseline, CAKES not only shows superior performance but also effectively reduces the model size

(56.80% less on medical and 19.35% less on video) and computational cost (53.76% less on medical and 19.01% less on video) significantly. The proposed method surpasses their 2D/3D/P3D counterparts significantly.

Our contributions can be summarized into three folds:

(1) We propose a more generic, efficient, and flexible alternative to 3D convolution by shrinking 3D kernels into heterogeneous yet complementary efficient counterparts at a fine-grained level.

(2) We automate the replacement configuration for simplifying 3D networks by customizing a search space based on CAKES and combining it with neural architecture search.

(3) By applying CAKES to different 3D models, we achieve comparable results to state-of-the-art while being much more efficient on both volumetric medical data and temporal-spatial video data.

3.2 Related Work

3.2.1 Efficient 3D Convolutional Neural Networks

Despite the great advances of 3D CNNs [22, 52, 208, 278], existing 3D networks usually require a heavy computational budget. Besides, 3D CNNs also suffer from unstable training due to a lack of pre-trained weights [22, 145, 201]. These facts have motivated researchers to find efficient alternatives to 3D convolutions. For example, it is suggested in [158, 209] apply group convolution [118] and depth-wise convolution [47] to 3D networks to obtain resource-efficient models. Another type of approach suggests replacing each 3D convolution layer with a structured combination of 2D and 1D convolution layers to achieve better

performance while being more efficient. For instance, [177] and [210] propose to use a 2D spatial convolution layer followed by a 1D temporal convolution layer to replace a standard 3D convolution layer. Besides, [238] demonstrates that 3D convolutions are not needed everywhere and some of them can be replaced by 2D counterparts. Similar attempts also occur in the medical imaging area [146]. For example, [78] tries to replace consecutive 3D convolution layers through consecutive 2D convolution layers followed by a 1D convolution layer.

Our method differs from these methods by the following folds: (1) Instead of applying homogeneous operations to all channels, CAKES allows assigning complementary heterogeneous operations at channel-wise, which leads to a more flexible design and a potentially better trade-off between accuracy and efficiency [203]; and (2) We enable the automatic optimization of the replacement configuration instead of manual design through a new search space.

3.2.2 Neural Architecture Search

Neural Architecture Search (NAS) aims at automatically discovering better network architectures than human-designed ones. It has been proved successful not only for 2D natural image recognition [290, 250], but also for other tasks such as segmentation [139] and detection [75]. Besides the success on natural images, there are also some trials on other data formats such as videos [186] and 3D medical images [259, 285]. Earlier NAS algorithms are based on either reinforcement learning [9, 290, 291] or evolutionary algorithm [181, 236]. However, these methods often require training each network candidate from scratch, therefore the intensive computational costs hamper its usage, especially with

a limited computational budget. Since [171] first proposed parameter sharing scheme, more and more search methods such as differentiable NAS approaches [38, 141, 243, 62] and one-shot NAS approaches [16, 82, 197, 130] began to investigate how to effectively reduce the search cost to several GPU days or even several GPU hours.

Moreover, [79, 161] successfully connect network pruning with NAS and design more efficient search methods. Some methods [203, 161, 197] also incorporate kernel size into the search space. Nevertheless, most of them only consider simple cases with choices among 3×3, 5×5, *etc.*, while we consider much more diverse and general kernel deployment across different channels in 3D settings.

3.3 Method

3.3.1 Revisit Variants of 3D Convolution

We first revisit 3D convolutions and existing alternatives. Without loss of generality, let $\mathbf{X}$ of size $C_i \times D_i \times H_i \times W_i$ denotes the input tensor, where C_i stands for the input channel number, and D_i, H_i, W_i represent the spatial depth (or temporal length), the spatial height, and the spatial width, respectively. The weights of the corresponding 3D kernel are denoted as $\mathbf{W}^{C_o \times C_i \times k_d \times k_h \times k_w}$, where C_o is the output channel number and $k_d \times k_h \times k_w$ denote the kernel size. For simplicity, we consider each output channel individually in the formulation. Therefore, the output tensor $\mathbf{Y}$ of shape $C_o \times D_o \times H_o \times W_o$ can be derived as following:

$$\mathbf{Y}_c^{D_o \times H_o \times W_o} = \mathbf{X}^{C_i \times D_i \times H_i \times W_i} \oplus \mathbf{W}_c^{C_i \times k_d \times k_h \times k_w}, \tag{3.1}$$

where $\oplus$ denotes convolution, c is the output channel index, *i.e.*, $1 \leq c \leq C_o$.

The computation overhead of 3D convolutions can be significantly heavier than their 2D counterparts. Consequently, the expensive computation and over-parameterization induced by 3D deep networks impede the scalability of network capacity. Recently, there are many works seeking to alleviate the high demand of 3D convolutions. One common strategy is to decouple the spatial and temporal components [177, 210]. The underlying assumption here is that the spatial and temporal kernels are orthogonal to each other, and therefore can effectively extract complementary information from different dimensions. Another option is to discard 3D convolutions and simply use 2D operations instead [238]. Mathematically speaking, these replacements can be written as:

$$\mathbf{W}_c^{C_i \times k_d \times k_h \times k_w} \leftarrow \{\mathbf{W}_c^{C_i \times 1 \times k_h \times k_w}, \mathbf{W}_c^{C_i \times k_d \times 1 \times 1}\} \tag{3.2}$$

$$\mathbf{W}_c^{C_i \times k_d \times k_h \times k_w} \leftarrow \{\mathbf{W}_c^{C_i \times 1 \times k_h \times k_w}\}, \tag{3.3}$$

where $\leftarrow$ indicates the replacement operation. Similar ideas also occur in 3D medical image analysis, where the images are volumetric data. For instance, it is shown in [145] that using 2D convolutions in the encoder and replacing 3D convolutions with Pseudo-3D (P3D) operations in the decoder not only largely reduce the computation overhead but also improves the performance over the traditional 3D networks.

Though these methods have furthered the model efficiency compared with standard 3D convolutions, there are several limitations yet to be tackled. On the one hand, as shown in Eqn. (3.2), decomposing the kernels into orthogonal 2D and 1D components is designed for a specific data type (*i.e.*, spatial-temporal), which may not well generalize to other types such as volumetric data. On the other hand, directly replacing 3D kernels with 2D operators

(Eqn. (3.3)) cannot effectively capture information along the third dimension.

To address these issues, we propose *Channel-wise Automatic KErnel Shrinking (CAKES)*, as a general alternative to 3D convolutions. The core idea is to shrink standard 3D kernels into a set of cheaper 1D, 2D, and 3D components. To ensure the flexibility of our design and avoid the tricky manual configuration, we further make the shrinkage channel-specific, thus heterogeneous kernels can extract complementary information as a 3D kernel does. We additionally introduce a brand-new search space so that the replacement configuration can be optimized automatically.

3.3.2 Kernel Shrinking as Path-level Selection

Let's consider the case for a single output channel, and abbreviate $\mathbf{W}_c^{C_i \times k_d \times k_h \times k_w}$ to $\mathbf{W}_c^{k_d \times k_h \times k_w}$ for simplicity. We aim to find the optimal sub-kernel $\mathbf{W}_c^{k_d' \times k_h' \times k_w'}$ ($1 \leq k_d' \leq k_d$,$1 \leq k_h' \leq k_h$,$1 \leq k_w' \leq k_w$) as the substitute for 3D kernel $\mathbf{W}_c^{k_d \times k_h \times k_w}$. Therefore, the original 3D kernels can be effectively reduced to smaller sub-kernels, leading to a more efficient model.

As shown in Fig. 3.2(a), even only considering different kernel sizes, there are $k_d \times k_h \times k_w$ sub-kernel options for a 3D kernel, which makes it impractical to find the optimal sub-kernel via manual designs. Therefore, we provide a new perspective—to formulate this problem as path-level selection [141], *i.e.*, to encode sub-kernels into a multi-path super-network and select the optimal path among them (Fig. 3.2(c)). Then this problem can be solved in a differentiable manner.

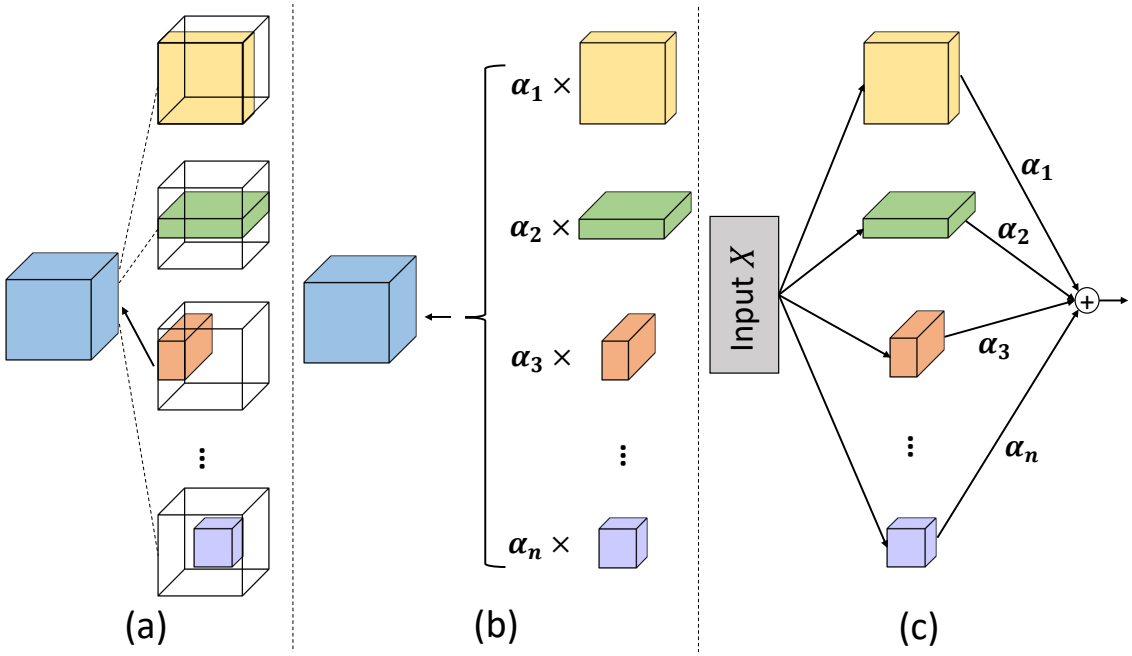

Figure 3.2: (a) Various sub-kernels of the same 3D kernel. (b) Representation of 3D kernel as a weighted summation of sub-kernels. (c) Path-level selection.

We first represent a general replacement to 3D kernel as follows (Fig. 3.2(b)):

$$\mathbf{W}_c^{k_d \times k_h \times k_w} \leftarrow \{\alpha_i \mathbf{W}_c^{k_d^i \times k_h^i \times k_w^i}\}_i, \tag{3.4}$$

where α_i is the weight of i-th sub-kernel $\mathbf{W}_c^{k_d^i \times k_h^i \times k_w^i}$, $1 \leq k_d^i \leq k_d$, $1 \leq k_h^i \leq k_h$, $1 \leq k_w^i \leq k_w$. With this formulation, the problem of finding the optimal sub-kernel of $\mathbf{W}_c^{k_d \times k_h \times k_w}$ can be approximated as finding the optimal setting of $\{\alpha_i\}$ and then keeping the sub-kernel with maximum α_i. Due to the linearity of convolution, Eqn. (3.1) can then be derived as below:

$$\mathbf{X} \oplus \mathbf{W}_c^{k_d \times k_h \times k_w} \leftarrow \sum_i \alpha_i (\mathbf{X} \oplus \mathbf{W}_c^{k_d^i \times k_h^i \times k_w^i}). \tag{3.5}$$

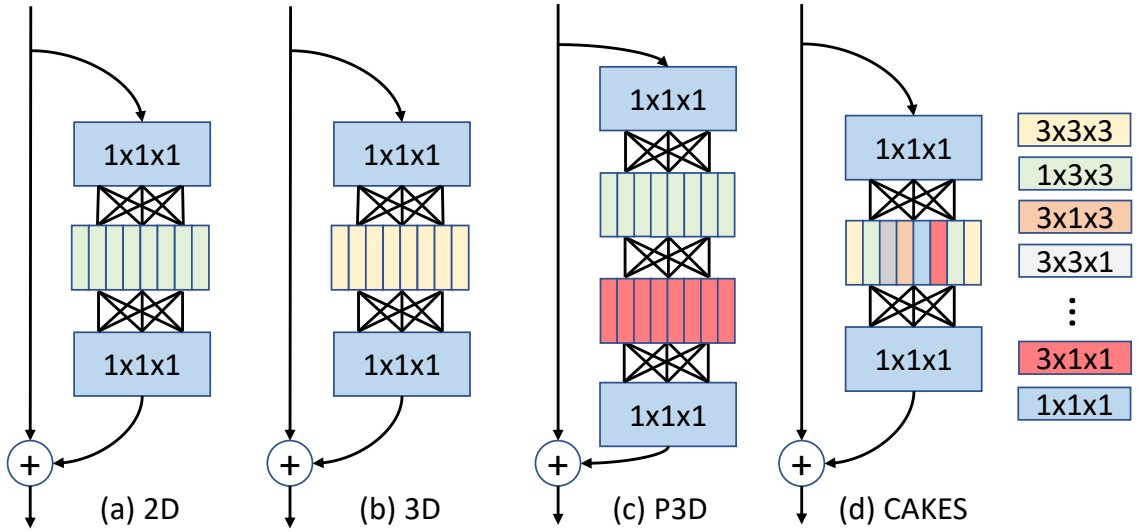

Figure 3.3: An illustrative example of comparison between different types of convolution in a residual block [92]. (a) 2D Convolution. (b) 3D Convolution. (c) P3D Convolution. (d) the proposed CAKES. In our case, starting from a 3D convolution, the 3D operation at each channel is replaced with an efficient sub-kernel.

To solve for the path weights $\{\alpha_i\}$, we reformulate Eqn. (3.5) as an over-parameterized multi-path super-network, where each candidate path consists of a sub-kernel (Fig. 3.2(c)). By relaxing the selection space, *i.e.*, relaxing the conditions on α to be continuous, Eqn. (3.5) can be then formulated as a differential NAS problem and optimized via gradient descent [141].

3.3.3 Channel-wise Shrinkage

While previous replacements [145, 177, 210] consist of homogeneous operations in the same layer, we argue that a more efficient replacement requires customized operations at each channel. As shown in Fig. 3.3, kernel shrinking in a channel-wise fashion can generate heterogeneous operations which extract diverse and complementary information within the

same layer, and thereby yields a fine-grained and more efficient replacement (Fig. 3.3(d)) than prior methods which use layer-wise replacements (Fig. 3.3(a) & (b) & (c)).

Contrary to previous layer-wise replacement, our core idea is to replace 3D kernel at each channel individually, thus the target is to find the optimal sub-kernel $\mathbf{W}_c^{k_d^c \times k_h^c \times k_w^c}$ as the substitute for the c-th output channel 3D kernel $\mathbf{W}_c^{k_d \times k_h \times k_w}$:

$$\mathbf{W}_c^{k_d \times k_h \times k_w} \leftarrow \{\mathbf{W}_c^{k_d^c \times k_h^c \times k_w^c}\}, \tag{3.6}$$

where the optimal size of the sub-kernel ($k_d^c \times k_h^c \times k_w^c$) is subjected to $1 \leq k_d^c \leq k_d$, $1 \leq k_h^c \leq k_h$, $1 \leq k_w^c \leq k_w$. Hence the computation incurred by Eqn. (3.1) can be largely reduced by our replacement as above.

With our channel-wise replacement design, the original 3D kernels are substituted by a series of diverse and cheaper operations at different channels as follows (recall that C_o is the output channel number):

$$\mathbf{W} \leftarrow \{\mathbf{W}_1^{k_d^1 \times k_h^1 \times k_w^1}, \mathbf{W}_2^{k_d^2 \times k_h^2 \times k_w^2}, \ldots, \mathbf{W}_{C_o}^{k_d^{C_o} \times k_h^{C_o} \times k_w^{C_o}}\}. \tag{3.7}$$

Benefited from channel-wise shrinkage, our method provides a more general and flexible design for replacing 3D convolution than previous approaches (Eqn. (3.2) and Eqn. (3.3)), where it can also be easily reduced to arbitrary alternatives (*e.g.*, 2D, P3D) by integrating these operations into the set of candidate sub-kernels. An illustration example can be found in Fig 3.3.

3.3.4 Search for an Efficient Replacement

As aforementioned, given the tremendous feasible choices, it is impractical to manually find the optimal replacement for a 3D kernel through a trial-and-error process. Especially, it

becomes even more intractable as the replacement procedure is conducted in a channel-wise manner. Therefore, we propose a new search space for efficient 3D networks and automate the process of learning an efficient replacement to fully exploit the redundancies in 3D convolution operations. By formulating kernel shrinkage as a path-level selection problem, we first construct a super-network where every candidate sub-kernel is encapsulated into a separate trainable branch (Fig. 3.2(c)) at each channel. Once the path weights are learned in a differentiable manner, the optimal path (sub-kernel) can be determined.

Search Space A well-designed search space is crucial for NAS algorithms [244]. Here we aim to answer the following questions: *Should the 3D convolution kernel be kept or replaced per channel? If replaced, which operation should be deployed instead?*

To address these questions, for each channel, we define a set $\mathcal{S}$, which contains all candidates of sub-kernels (replacement) given a 3D kernel $\mathbf{W}^{k_d \times k_h \times k_w}$:

$$\begin{aligned} \mathcal{S} &= \{\mathbf{W}^{k_{d_1} \times k_{h_1} \times k_{w_1}}, \mathbf{W}^{k_{d_2} \times k_{h_2} \times k_{w_2}}, \ldots, \mathbf{W}^{k_{d_n} \times k_{h_n} \times k_{w_n}}\} \\ \mathbf{W}_c^{k_d^c \times k_h^c \times k_w^c} &= \text{Choose}(\mathcal{S}). \end{aligned} \tag{3.8}$$

As the original 3D convolution kernel can be considered a sub-kernel of itself, *i.e.*, $\mathbf{W}^{k_d \times k_h \times k_w} \in \mathcal{S}$, it can be kept in the final configuration. The final optimal operation $\mathbf{W}_c$ is chosen among $\mathcal{S}$.

Another critical problem for NAS is how to reduce search cost. To make the search cost affordable, we adopt a differentiable NAS paradigm where the model structure is discovered in a single-pass super-network training. Drawing inspirations from previous NAS methods, we directly use the scaling parameters in the normalization layers as the path weights α of the multi-path super-network (Eqn. (3.5)) [79, 161]. And our goal is then equivalent to finding the optimal sub-network architecture based on the learned path weights. To achieve

this goal, we introduce two different search manners which aim at either maximizing the performance or optimizing the computation cost of the sub-network as a search priority, named as performance-priority and cost-priority search, respectively.

Performance-Priority Search The search aims to maximize the performance by finding the optimal sub-kernels given the backbone architecture. During the search procedure, following [14, 16], we randomly pick an operation for each channel at each iteration. This not only allows for memory saving by activating and updating one path per iteration but also propels the weights of the paths in the super-network training to be decoupled. After the super-network is trained, the operation with the largest path weight will be picked as the final choice for the given output channel:

$$\begin{aligned} \mathbf{W}_c^{k_d \times k_h \times k_w} &\leftarrow \{\mathbf{W}^{k_{d_{i^*}} \times k_{h_{i^*}} \times k_{w_{i^*}}}\}, \\ \text{where } \ i^* &= \text{argmax}_{i \in \{1 \cdots n\}}(\alpha_i). \end{aligned} \tag{3.9}$$

Cost-Priority Search Performance-priority may neglect the possible negative effects on the computation cost. In order to obtain more compact models, we also introduce a “cost-priority" search method. Inspired by [161], we search the model in a pruning manner with a penalty on expensive operations. The outputs of each sub-kernels are concatenated and aggregated by the following $1 \times 1 \times 1$ convolution. To make the searched architecture more compact, we introduce a “cost-aware" penalty term—A lasso term on α which is used as the penalty loss to push many path weights to near-zero values. Therefore, the total training loss $\mathcal{L}$ can be written as:

$$\mathcal{L} = \mathcal{E} + \lambda \sum_i \beta_i |\alpha_i|, \tag{3.10}$$

where β_i is a “cost-aware" term to balance the penalty term, which is proportional to the parameters or FLOPs cost of the sub-kernel. In Table 3.1, we also empirically show that this term can lead to a more efficient architecture. The introduction of β_i aims at giving more penalty to “expensive" operations and leading to a more efficient replacement. λ is the coefficient of the penalty term, and $\mathcal{E}$ is the conventional training loss (*e.g.*, cross-entropy loss combined with the regularization term such as weight decay).

3.4 Experiments

3.4.1 3D Medical Image Segmentation

Dataset We evaluate the proposed method on two public datasets: 1) Pancreas Tumours dataset from the Medical Segmentation Decathlon Challenge (MSD) [195], which contains 282 cases with both pancreatic tumours and normal pancreas annotations; and 2) NIH Pancreas Segmentation dataset [184], consisting of 82 abdominal CT volumes. **For the MSD dataset**, we use 226 cases for training and evaluate the segmentation performance on the rest 56 cases. The resolution along the axial axis of this dataset is extremely low and the number of slices can be as small as 37. For data preprocessing, all images are resampled to an isotropic 1.0 mm^3 resolution. **For the NIH dataset**, the resolution of each scan is $512 \times 512 \times L$, where $L \in [181, 466]$ is the number of slices along the axial axis and the voxel spacing ranges from 0.5 mm to 1.0 mm. We test the model in a 4-fold cross-validation manner following previous methods [279, 280].

Implementation Details For all experiments, C2FNAS [259] is used as the backbone architecture. When replacing the operations, we keep the stem (the first two and the last

Methods	Params (M)	FLOPs (G)	Pancreas DSC (%)	Tumor DSC (%)	Average DSC (%)
2D	11.29	97.77	79.16	43.02	61.09
3D	22.50	188.48	80.34	47.57	63.96
P3D	13.16	112.88	**80.36**	45.27	62.82
$\text{CAKES}^{\text{M}}_{\text{1D}}$	7.56	67.53	79.77	42.73	61.25
$\text{CAKES}^{\text{M}}_{\text{2D}}$	11.29	97.77	80.09	46.17	63.13
$\text{CAKES}^{\text{M}}_{\text{1,2,3D}}$	11.41	99.17	79.82	45.27	62.55
$\text{CAKES}^{\text{P}}_{\text{1D}}$	**7.56**	**67.53**	80.32	45.57	62.95
$\text{CAKES}^{\text{P}}_{\text{2D}}$	11.29	97.77	80.05	48.51	64.28
$\text{CAKES}^{\text{P}}_{\text{1,2,3D}}$	11.26	99.68	80.12	**48.72**	**64.42**
$\text{CAKES}^{\text{C}}_{\text{1,2,3D}}$	9.72	87.16	80.34	47.95	64.15

Table 3.1: Comparison among different operations and configurations. The subscripts of 1D, 2D, and 3D indicate the dimensions of the operations being used. The superscripts of “M", “P", “C" represent “Manual", “Performance-Priority", and “Cost-Priority" respectively.

two convolution layers) the same. For 3D medical images, for simplicity, we choose a set of most representative sub-kernels as $\mathcal{S}$. The operations set contains conv1D ($1 \times 1 \times 3$, $1 \times 3 \times 1$, $3 \times 1 \times 1$), conv2D ($1 \times 3 \times 3$, $3 \times 1 \times 3$, $3 \times 3 \times 1$) from different directions, and conv3D ($3 \times 3 \times 3$). For every 3D kernel at each output channel, a sub-kernel from $\mathcal{S}$ will be chosen as the replacement. **For manual settings**, we assign all candidate operations uniformly across the output channels. **For NAS settings**, we include both “performance-priority” and “cost-priority” search for performance comparison.

Training stage For the MSD dataset, we use random crop with patch size of $96 \times 96 \times 96$, random rotation (0°, 90°, 180°, and 270°) and flip in all three axes as data augmentation. The batch size is 8 with 4 GPUs. We use SGD optimizer with a learning rate starting from 0.01 with polynomial decay of power of 0.9, momentum of 0.9, and weight decay of 0.00004. The training lasts for 40k iters. The loss function is the summation of dice loss [162] and cross-entropy loss. For NIH dataset, the patch size is set as $96 \times 96 \times 64$, following the settings in [285]. The found architecture will be trained **from scratch** to

Method	Params	Average DSC	Max DSC	Min DSC
Coarse-to-fine [279]	268.56M	82.37%	90.85%	62.43%
RSTN [257]	268.56M	84.50%	91.02%	62.81%
C2F ResDSN [286]	20.06M	84.59%	91.45%	69.62%
V-NAS [285]	29.74M	85.15%	91.18%	70.37%
$\text{CAKES}^{\text{C}}_{1,2,3\text{D}}$	**9.27M**	84.85%	91.61%	59.32%
$\text{CAKES}^{\text{P}}_{1,2,3\text{D}}$	11.26M	**85.28**%	**91.98**%	**72.78**%

Table 3.2: Comparison with prior arts on the NIH dataset.

ensure a fair comparison. Both the super-network and the found architecture are trained under the same settings as aforementioned. For the search stage with "cost-priority" setting, a lasso term with coefficient $\lambda = 1.0 \times 10^{-4}$ is applied to the path weights. And it is further re-weighted by $\beta = \{\frac{9}{13}, \frac{3}{13}, \frac{1}{13}\}$ for 3D, 2D, 1D operations respectively, which is their ratio of the parameters. After the training process, the operation with the largest α is chosen as the final replacement for 3D operation for each channel.

Testing stage We test the network in a sliding-window manner, where the patch size is $96 \times 96 \times 96$ and stride is $32 \times 32 \times 32$ for the MSD dataset and patch size is $96 \times 96 \times 64$ and stride is $20 \times 20 \times 20$ for NIH dataset. The result is measured with Dice-Sørensen coefficient (DSC) metric, which is formulated as $\text{DSC}\ (\mathcal{Y}, \mathcal{Z}) = \frac{2 \times |\mathcal{Y} \cap \mathcal{Z}|}{|\mathcal{Y}| + |\mathcal{Z}|}$, where $\mathcal{Y}$ and $\mathcal{Z}$ denote the prediction and ground-truth voxels set for a foreground class. The DSC has a range of $[0, 1]$ with 1 implying a perfect prediction.

Manual Settings vs. Auto Settings As observed from Table 3.1, even under manual settings, CAKES is already much more efficient with slightly inferior performance (*e.g.*, from 3D to manual $\text{CAKES}^{\text{M}}_{\text{2D}}$, parameters drop from 22.50M to 11.29M, and FLOPs drop from 188.48G to 97.77G, with performance gap of $< 1.0\%$). Besides, $\text{CAKES}^{\text{M}}_{\text{2D}}$ outperforms its counterpart with standard convolution 2D layers by more than 2.0% with

the same model size, which indicates the benefits of our design. In addition, with the proposed search space and method, CAKES can further reduce the performance gap and even surpasses the original 3D model with much fewer parameters and computations, *e.g.*, model size is reduced from 22.50M (3D) to 11.26M ($\text{CAKES}^{\text{P}}_{1,2,3\text{D}}$), and FLOPs drop from 188.48G (3D) to 99.68G ($\text{CAKES}^{\text{P}}_{1,2,3\text{D}}$), with a performance improvement of 0.46%. Compared with P3D, $\text{CAKES}^{\text{P}}_{1,2,3\text{D}}$ also yields superior performance (+1.60%) with a more compact model (11.26M vs. 13.16M), which further indicates the effectiveness of the proposed method.

Influence of the Search Space From Table 3.1, we can see that using different search spaces, CAKES consistently outperforms its counterparts with standard 1D/2D/3D convolutions. Out of different search spaces, we find that $\text{CAKES}^{\text{P}}_{1\text{D}}$ (7.56M params and 67.53G FLOPs) offers the most efficient model with comparable performance, while $\text{CAKES}^{\text{P}}_{2\text{D}}$ (11.29M params and 97.77G FLOPs) can already surpass the 3D baseline (22.50M params and 188.48G FLOPs) with half parameters and computation cost. After we enlarge the search space, $\text{CAKES}^{\text{P/C}}_{1,2,3\text{D}}$ obtains a configuration with even higher performance/efficiency (last 2 rows of Table 3.1).

Generalization to different backbone architectures We also test our method on different backbone architectures. Applying $\text{CAKES}^{\text{C}}_{1,2,3\text{D}}$ to another strong model 3D ResDSN [286, 134], our method consistently leads to a more efficient model with much fewer parameters (10.03M to 4.63M) and FLOPs (192.07G to 98.12G) with comparable performance (61.96% to 61.65%).

NIH Results We compare CAKES with state-of-the-art methods in Table 3.2, where it can be observed that the proposed method leads to a much more compact model size

Model	Params (M)	FLOPs (G)	top1	top5
C2D	23.9	33.0	17.2	43.1
P3D	27.6	37.9	44.8	74.6
C3D	46.5	62.6	46.8	75.3
$\text{CAKES}^{\text{M}}_{1,2\text{D}}$	**20.1**	**28.0**	46.2	75.2
$\text{CAKES}^{\text{M}}_{2,3\text{D}}$	35.2	47.7	46.8	76.0
$\text{CAKES}^{\text{P}}_{1,2\text{D}}$	20.9	29.1	47.1	75.9
$\text{CAKES}^{\text{P}}_{2,3\text{D}}$	37.5	50.7	**47.4**	**76.1**
$\text{CAKES}^{\text{P}}_{1,2,3\text{D}}$	33.5	43.9	47.2	75.7
$\text{CAKES}^{\text{C}}_{1,2\text{D}}$	20.5	29.3	46.8	76.0
$\text{CAKES}^{\text{C}}_{2,3\text{D}}$	35.7	41.4	46.9	75.6
$\text{CAKES}^{\text{C}}_{1,2,3\text{D}}$	35.0	38.7	46.9	75.5

Table 3.3: Comparison among operations and configurations for ResNet50 backbone in terms of parameter number, computation amount (FLOPs), and performance on Something-Something V1 dataset.

compared to other models. For instance, our model size is more than 25× smaller than that of [279] and [257]. It is well worth noting that our model performed in a single-stage fashion already outperforms many state-of-the-art methods conducted in a two-stage coarse-to-fine manner [279, 257, 286] on the NIH pancreas dataset with much fewer model parameters and FLOPS. It is also noteworthy to mention that the applied architecture is searched from another dataset (MSD), where images are collected under different protocols and have different resolutions. This result indicates the generalization of our searched model. By directly applying the architecture searched on the MSD dataset, our method also outperforms [285] which was directly searched on the NIH dataset with less than half the model size.

3.4.2 Action Recognition in Videos

Dataset Something-Something V1 [81] is a large-scale action recognition dataset that requires comprehensive temporal modeling. There are about 110k videos for 174 classes

Method	Backbone Architecture	#Frame	FLOPs	#Param.	top1	top5
TSN [221]	ResNet-50	8	33G	24.3M	19.7	46.6
TRN-2stream [276]	BNInception	8+8	-	36.6M	42.0	-
ECO [289]	BNIncep+3D Res18	8	32G	47.5M	39.6	-
ECO [289]	BNIncep+3D Res18	16	64G	47.5M	41.4	-
ECO_{En} *Lite* [289]	BNIncep+3D Res18	92	267G	150M	46.4	-
I3D [22]	3D ResNet-50	32×2clip	153G×2	28.0M	41.6	72.2
NL I3D [225]	3D ResNet-50	32×2clip	168G×2	35.3M	44.4	76.0
NL I3D+GCN [226]	3D ResNet-50+GCN	32×2clip	303G×2	62.2M	46.1	76.8
TSM [136]	ResNet-50	8	33G	24.3M	45.6	74.2
TSM [136]	ResNet-50	16	65G	24.3M	47.2	77.1
S3D [238]	BNInception	64	66.38G	-	47.3	78.1
S3D-G [238]	BNInception	64	71.38G	-	48.2	**78.7**
$\text{CAKES}^{\text{C}}_{1,2\text{D}}$	ResNet-50	8	**29.3G**	**20.5M**	46.8	76.0
$\text{CAKES}^{\text{P}}_{2,3\text{D}}$	ResNet-50	8	50.7G	37.5M	**47.4**	76.1
$\text{CAKES}^{\text{P}}_{1,2,3\text{D}}$	ResNet-50	8	43.9G	33.5M	47.2	75.7
$\text{CAKES}^{\text{C}}_{1,2\text{D}}$	ResNet-50	16	**58.6G**	**20.5M**	48.0	78.0
$\text{CAKES}^{\text{P}}_{2,3\text{D}}$	ResNet-50	16	101.4G	37.5M	48.6	78.6
$\text{CAKES}^{\text{P}}_{1,2,3\text{D}}$	ResNet-50	16	87.8G	33.5M	**49.4**	78.4

Table 3.4: Comparing CAKES against other methods on Something-Something V1 dataset. We mainly consider the methods that adopt convolutions in a fully-connected manner and only take RGB as input for a fair comparison.

with diverse objects, backgrounds, and viewpoints.

Implementation Details We adopt ResNet50 [92] with pre-trained weight on ImageNet [118] as our backbone. The 3D convolution weights are initialized by repeating the 2D kernel by 3 times along the temporal dimension following [22], while 1D convolution weights are initialized by averaging the 2D kernel on spatial dimensions and then repeat by 3 times along the temporal axis. For the temporal dimension, we use the sparse sampling method as in [221]. For spatial dimension, the short side of the input frames is resized to 256 and then cropped to 224×224.

Training Stage We use random cropping and flipping as data augmentation. We train

the network with a batch size of 96 on 8 GPUs with SGD optimizer. The learning rate starts from 0.04 for the first 50 epochs and decays by a factor of 10 for every 10 epochs afterwards. The total training epochs are 70. We also set the dropout ratio to 0.3 following [226]. The training settings remain the same for both final network and search stage, except that when searching with “performance-priority" we double the training epochs to ensure convergence, and with “cost-priority", we use a lasso term with $\lambda = 1.0 \times 10^{-4}$ and $\beta = \{\frac{9}{13}, \frac{3}{13}, \frac{1}{13}\}$ for 3D, 2D, 1D operations respectively.

Testing Stage we sample the middle frame in each segment and perform a center crop for each frame. We report the results of **single crop**, unless otherwise specified.

Ablation Study We study the impacts of both different operation sets and manual/auto configurations. The results are summarized in Table 3.3. Considering the spatial-temporal property of video data, we study the following different operations set: (1) Spatial 2D convolution and temporal 1D convolution; (2) Spatial 2D convolution and 3D convolution; (3) Spatial 2D, temporal 1D, and 3D convolutions.

Operation Set with 1D & 2D Sub-kernels As shown in Table 3.3, $\text{CAKES}^{\text{C}}_{\text{1,2D}}$ surpass the 2D baseline by a large margin (+29.6%) , while the model size reduces by 14.23%. This suggests that TSN [221] may lack the ability to capture temporal information, therefore replacing some of the 2D operations to temporal 1D operations can significantly increase the performance and reduce the model size. Besides, it also surpasses P3D, where each 2D convolution is followed by a temporal 1D convolution, with a significant advantage on both performance (+2.0%) and model cost (25.72% fewer params and 53.19% fewer FLOPs), indicating CAKES makes better use of redundancies in the networks than P3D. Therefore, CAKES using an operation set containing 1D and 2D sub-kernels can be an

ideal design when looking for efficient video understanding networks.

Operation Set with 2D & 3D Sub-kernels We aim to see how CAKES balances the trade-off between performance and model cost. From Table 3.3, $\text{CAKES}^{\text{C}}_{\text{2,3D}}$ yields a much more compact model (-23.23%/33.87% params/FLOPs) with comparable performance to C3D. Under "performance-priority" setting, $\text{CAKES}^{\text{P}}_{\text{2,3D}}$ searches a slightly larger model, yet its performance boosts significantly to 47.4%.

Operation Set with 1D & 2D & 3D Sub-kernels When compared to $\text{CAKES}^{\text{C}}_{\text{2,3D}}$, $\text{CAKES}^{\text{C}}_{\text{1,2,3D}}$ shows a similar performance with much fewer FLOPs (38.7G vs. 41.4G). Besides, under the "performance-priority" setting, $\text{CAKES}^{\text{P}}_{\text{1,2,3D}}$ produces a comparable performance to $\text{CAKES}^{\text{P}}_{\text{2,3D}}$ with less computation cost (43.9G vs. 50.7G). This result indicates that with a more general search space (*e.g.*, 1D, 2D, and 3D), the proposed CAKES can find more flexible designs, which leads to better performance/efficiency.

Results A comparison with other state-of-the-art methods is shown in Table 3.4. We report the model performance under both 8-frame and 16-frame settings. Compared with other state-of-the-art methods, $\text{CAKES}^{\text{P}}_{\text{2D,3D}}$ sampling only 8 frames can already outperform most current methods. With smaller parameters and FLOPs, $\text{CAKES}^{\text{P}}_{\text{2D,3D}}$ surpasses those complex models such as non-local networks [225] with graph convolution [226]. Comparing $\text{CAKES}^{\text{C}}_{\text{1,2D}}$ to other efficient video understanding frameworks such as ECO [289] and TSM [136], our model is not only more light-weight (58.6G vs. 64G/65G), but also delivers better performance (48.0% vs. 41.4%/47.2%). And our best model $\text{CAKES}^{\text{P}}_{\text{1,2,3D}}$ achieves a new state-of-the-art performance of 49.4% top-1 accuracy with moderate model size. An interesting finding is that although $\text{CAKES}^{\text{P}}_{\text{1,2,3D}}$ shows

similar performances to $\text{CAKES}^{\text{P}}_{2,3\text{D}}$ with 8-frame inputs, it achieves a much higher accuracy when it comes to the 16-frame scenario, which demonstrates that with a more general search space, $\text{CAKES}^{\text{P}}_{1,2,3\text{D}}$ shows stronger transferability than other counterparts.

Cost-Priority Architecture We plot the found architecture on both medical data ($\text{CAKES}^{\text{C}}_{1,2,3\text{D}}$) and video data ($\text{CAKES}^{\text{C}}_{2,3\text{D}}$) respectively in Fig 3.4. **For the architecture searched on Something-Something dataset**, we note that the algorithm prefers efficient 2D operations at the bottom of the network, and favors 3D operation at the top of the network. This implies that the search algorithm successfully finds that **temporal information extracted from high-level features is more useful**, which coincides with the observation in [238]. **For the architecture found on the MSD dataset**, we calculated the number of operations computed on all three data dimensions, and the numbers are (1285, 1260, 1314). This suggests that the searched model, unlike the searched network for videos, tends to **treat each dimension equally**, which aligns with the property of volumetric medical data. In addition, the number of 1D, 2D, 3D operations are 1378, 1065, and 117 respectively, indicating that the efficient 1D/2D operations are more preferred.

Performance-Priority Architecture As shown in Fig. 3.5, for $\text{CAKES}^{\text{P}}_{1,2,3\text{D}}$, the pure temporal sub-kernel ($3 \times 1 \times 1$) is rarely chosen at the top of the network, while it plays a more important role as the network goes deeper. This observation agrees with our previous finding: temporal information extracted from high-level features is more useful. For $\text{CAKES}^{\text{P}}_{1,2,3\text{D}}$ on the medical image as shown in Fig. 3.5, we calculate the numbers of each type of sub-kernels and computation on three axes. The numbers of 1D, 2D, and 3D sub-kernel are 1135, 1073, 352 and the number of operations computed on all three data dimensions are 1437, 1433, 1467, which again coincides with our previous finding

that each dimension plays an equally important role for the symmetric volumetric data. Compared to cost-priority CAKES^{C}, we notice that performance-priority CAKES^{P} favors the operation set with more 3D sub-kernels, which can provide a larger model capacity.

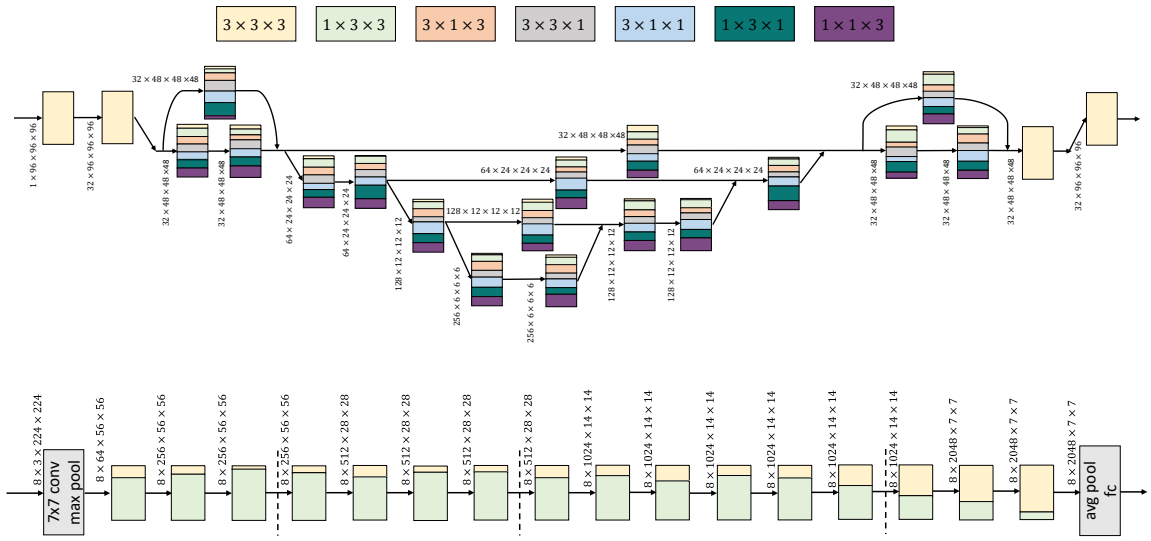

Figure 3.4: The searched architecture of CAKES^{C} on medical data and video data. Each color represents a type of sub-kernel. The heights of these blocks are proportional to their ratios in the corresponding convolution layer. The beginning and ending $1 \times 1 \times 1$ convolutions at each residual block are not visualized.

3.5 Conclusions

As an important solution to various 3D vision applications, 3D networks still suffer from over-parameterization and heavy computations. How to design efficient alternatives to 3D operations remains an open problem. In this chapter, we propose Channel-wise Automatic KErnel Shrinking (CAKES), where standard 3D convolution kernels are shrunk into efficient sub-kernels at channel-level, to obtain efficient 3D models. Besides, by formulating kernel

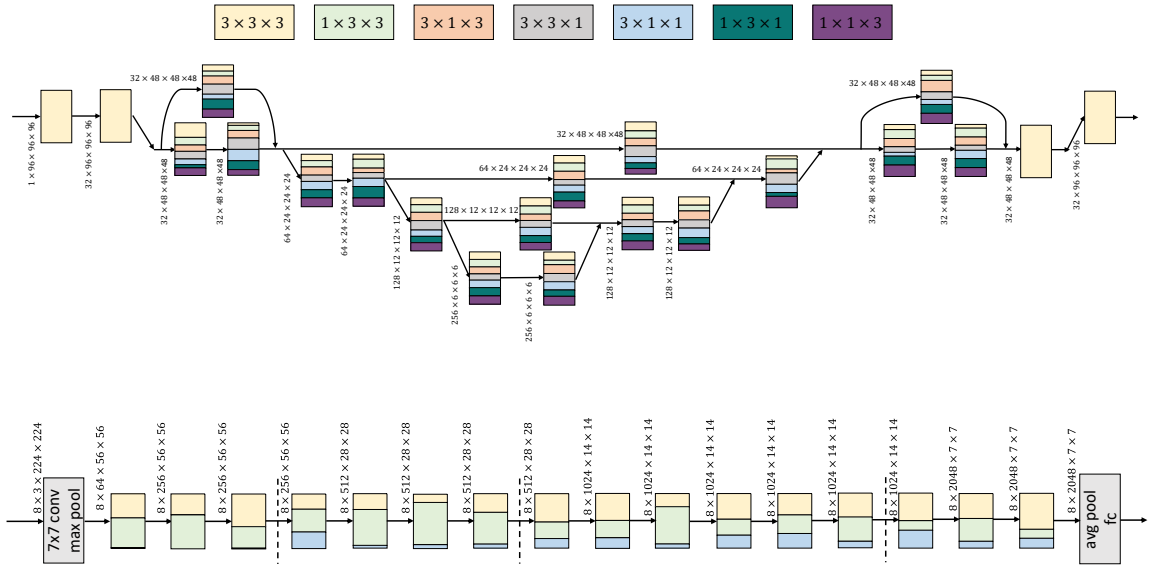

Figure 3.5: The searched architecture of CAKES^{P} on medical data and video data. Each color represents a type of sub-kernel. The heights of these blocks are proportional to their ratios in the corresponding convolution layer. The beginning and ending $1 \times 1 \times 1$ convolutions at each residual block are not visualized.

shrinkage as a path-level selection problem, our method can automatically explore the redundancies in 3D convolutions and optimize the replacement configuration. By applying on different backbone models, the proposed CAKES significantly outperforms previous 2D/3D/P3D and other state-of-the-art methods on both 3D medical image segmentation and action recognition from videos.

Chapter 4

Mask Guided Matting via Progressive Refinement Network

Starting from this chapter, we switch to natural image segmentation topics, which is more challenging and usually brings useful insights for 3D segmentation models. We start with natural image matting, which is a special segmentation task that predicts a transparent mask with hair-level details. In this chapter, we propose a general mask-guided matting framework with progressive refinement modules, which does not need users' input and thus can fully automatic deals with large-scale images.

4.1 Introduction

Image matting is a fundamental computer vision problem that aims to predict an alpha matte to precisely cut out an image region. It has many applications in image and video editing [220, 242, 123]. Most previous matting methods require a well-annotated trimap as an auxiliary guidance input [220], which explicitly defines the regions of foreground and background as well as the unknown part for the matting methods to solve. Although such

annotation makes the problem more tractable, it can be quite burdensome for users and limits the usefulness of these methods in many non-interactive applications.

Recently, researchers start to study the matting problem in a trimap-free setting. One direction is to get rid of any external guidance, and hope that the matting model can capture both semantics and details by end-to-end training on large-scale datasets [268, 176]. Nevertheless, these methods are faced with the generalization challenge due to the lack of semantic guidance when tested on complex real-world images. Another line of works investigates alternatives to the trimap guidance, easing the requirement for human input [142, 188, 98, 84]. For example, [98, 84] proposed techniques for automatic trimap generation, while [188] takes background images instead as extra inputs. However, these methods often require a very specific type of guidance they are trained with and thus become less appealing when the guidance inputs may have varied characteristics or forms.

In this work, we introduce a Mask Guided (MG) Matting method which takes a general coarse mask as guidance. MG Matting is very robust to the guidance input and can obtain high-quality matting results using various types of mask guidance such as a trimap, a rough binary segmentation mask or a low-quality soft alpha matte. To achieve such robustness to guidance input, we propose a Progressive Refinement Network (PRN) module, which learns to provide self-guidance to progressively refine the uncertain matting regions through the decoding process. To further enhance the robustness of our method to external guidance, we also develop a series of guidance mask perturbation operations including random binarization, random morphological operations, and also a stronger perturbation CutMask to simulate diverse guidance inputs during training.

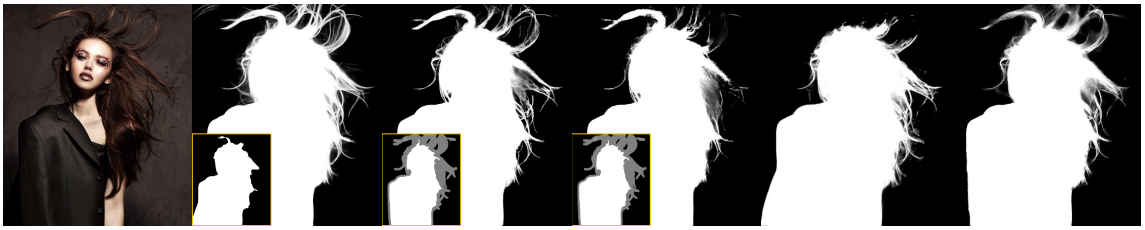

Figure 4.1: A visual comparison of MG and other matting methods including the commercial matting method in PhotoShop. The guidance input (see Sec. 4.5 for details.) is located at the bottom-left of each image. Note that BSHM [142] has an internal segmentation prediction network and thus does not take the external mask. Best viewed zoomed in.

In addition to alpha matting prediction, we also revisit the foreground color prediction problem for matting. Without accurately recovering the foreground color in the transparent region, the composited image will suffer from the fringing issue. We note that the foreground color labels in the widely-used dataset [242] are suboptimal for model training due to the labeling noise and limited diversity. As a simple yet effective solution, we propose Random Alpha Blending (RAB) to generate synthetic training data from random alpha mattes and images. We show that such a simple method can improve foreground color prediction accuracy without requiring additional manual annotations. As a result, combined with the proposed PRN, MG Matting is able to generate more visually plausible composition results.

Our contributions can be summarized as follows:

- We propose Mask Guided Matting, a general matting framework working with guidance masks in various qualities and even forms, and achieve a new state-of-the-art performance evaluated on both synthetic and real-world datasets.

- We introduce Progressive Refinement Network (PRN) along with a guidance perturbation training pipeline as a solution to learning a robust matting model.
- We study the problem of foreground color prediction for matting and propose a simple improvement using random alpha blending.

In addition, we collect and release a high-quality matting benchmark dataset of real images to evaluate the real-world performance of matting models.

4.2 Related Work

Trimap-based Image Matting A majority of matting methods require a trimap as additional input, which divides an image into foreground, background, and unknown regions. Traditional methods are often sampling-based or propagation-based. Sampling-based ones [73, 51, 90, 190, 219] estimate foreground/background color statistics through sampling pixels in the definite foreground/background regions to solve the alpha matte in the unknown region. The propagation-based methods [36, 121, 123, 124, 199, 91], also known as affinity-based methods, estimate alpha mattes by propagating the alpha value from the foreground and background pixels to the unknown area.

Recently, deep learning approaches have been proven successful in many areas, including classification [92, 202, 132, 130], detection [89, 7, 8], and segmentation [28, 260]. It also has achieved great success in image matting. [242] created a matting dataset with annotated mattes composited to various background images, and trained a deep network on it. Later, [160] introduced a generative adversarial framework to improve the results. [204] proposed to combine the sampling-based method and deep learning. [156] introduced a

new index-guided upsampling and unpooling operations to better keep details in the predictions. [97] proposed a two-encoder two-decoder architecture to simultaneously estimate foreground and alpha. [131] further boosts the performance with a contextual attention module.

Trimap-free Image Matting It is noticeable that there are also some trials [3, 192] to get rid of the trimap to predict alpha matte. [268] proposed a framework consisting of a segmentation network and a fusion network, where the input is only a single RGB image. Later, [142] introduced a trimap-free framework consisting of a mask prediction network, quality unification network, and matting refinement network for human portrait matting. The trimap-free matting performance is further boosted with attention module [176]. However, these trimap-free methods still have some gaps to trimap-based ones in terms of performance. Another direction is to use alternative guidance to trimap. [188] introduced a framework taking background images along with other potential priors (*e.g.*, segmentation mask, motion cue) as additional inputs. It shows great potential and can obtain comparable performance to state-of-the-art trimap-based methods.

Foreground Color Decontamination Many conventional matting methods [73, 123] proposed to predict both alpha matte and foreground color for extracting foreground objects. However, it is only very recently [97] incorporated foreground prediction into the deep learning framework. Later, [188] also predicts foreground color to reduce artifacts for a better composition result. Nevertheless, these methods mainly add a foreground decoder and directly learn from color label in [242], which only provides limited training samples and, more seriously, the color labels can be inaccurate and noisy(see Fig. 4.3). [69] proposes to use [123] to obtain a smoother color label.

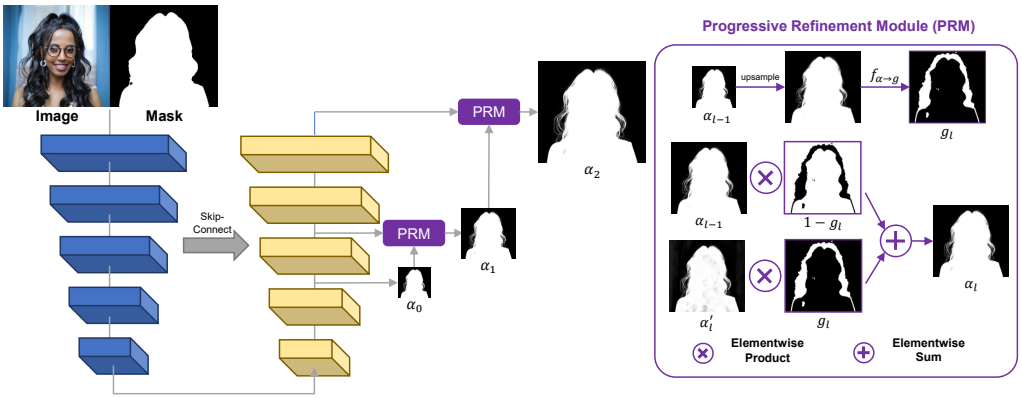

Figure 4.2: The proposed PRN. The network predicts alpha matte at multiple resolutions, while the one at lower resolution provides guidance about the uncertain regions to be refined in the next prediction.

Our method differs from the algorithms mentioned above in the following folds: 1) Our model works in a more general setting where only an easy-to-obtain coarse mask, no matter user-defined or model-predicted, is needed as guidance. It could handle different qualities and even various types of guidance as input. Thus it could be used as either trimap-based or trimap-free model depending on what guidance is available. Our model could also leverage stronger guidance to achieve even finer details. 2) Our methods could also predict the foreground color. Unlike [97], where the foreground prediction is directly learned from the color label, we note that the limited training data and inaccurate human label result in undesired results, especially in the boundary regions. Instead, we propose to use Random Alpha Blending to avoid the bias in the label, which not only introduces more diverse training samples but also avoid the inaccurate color label locating in boundary regions.

4.3 MG Matting

The problem of image matting can be formulated as:

$$\mathbf{I} = \alpha\mathbf{F} + (1-\alpha)\mathbf{B}, \alpha \in [0,1], \tag{4.1}$$

where $\mathbf{I}$, $\mathbf{F}$, $\mathbf{B}$, and α refer to the image color, foreground color, background color, and alpha matte respectively. As only $\mathbf{I}$ is observed, this is a very ill-posed problem. To solve the matting problem, most methods require a trimap input, which labels the foreground region (*i.e.* $\alpha = 1$), the background region (*i.e.* $\alpha = 0$), and the unknown part. In practice, the trimap input can contain various levels of noise and errors, making the matting results inconsistent.

We relax the strong assumption of the trimap by proposing a Mask Guided Matting method. The mask guidance, such as a predicted segmentation mask or a rough manual selection, only provides a coarse spatial prior of the foreground region. Therefore, our MG Matting method needs a more high-level semantic understanding of the input mask, so that it can detect the foreground/background region and the soft transparent part robustly. Meanwhile, our model has to capture image low-level patterns such as edge and texture to produce fine details of the target matte. Coordinating the high-level and the low-level feature learning is the key to the design of our MG Matting method.

To this end, we introduce Progressive Refinement Network (PRN), which provides a coarse-to-fine self-guidance to progressively refine the uncertain regions during the decoding process. In the following, we present the details of PRN, the training formulation, and some data augmentation techniques to enhance the robustness of our model.

4.3.1 Progressive Refinement Network

An overview of the PRN is shown in Fig. 4.2. The structure of our PRN follows the popular encoder-decoder network with skip connections. Our network takes an image and a coarse mask as input and outputs a matte. During the decoding process, PRN has a side matting output at each feature level. The side outputs with deep supervision have been shown to improve feature learning at different scales [239]. However, unlike [239], we find that linearly fusing the side outputs is not ideal for the matting problem (see Table 4.4 for details). This is because the image region closer to the object boundary requires lower-level features to delineate the foreground, while identifying internal object regions needs higher-level guidance.

To address this problem, we introduce a Progressive Refinement Module (PRM) at each feature level to selectively fuse the matting outputs from the previous level and the current level. Specifically, for the current level l we generate a self-guidance mask g_l from the matting output α_{l-1} of the previous level using the following function:

$$f_{\alpha_{l-1} \to g_l}(x, y) = \begin{cases} 1 & \text{if } 0 < \alpha_{l-1}(x, y) < 1, \\ 0 & \text{otherwise.} \end{cases} \tag{4.2}$$

The α_{l-1} is firstly upsampled to match the size of the raw matting output α'_l of the current level and then produces resultant self-guidance mask g_l. The self-guidance mask defines the transparent region (*i.e.* $0 < \alpha < 1$) as unknown and replaces the unknown region of α_{l-1} with the current raw output α'_l to obtain an updated α_l of current level:

$$\alpha_l = \alpha'_l g_l + \alpha_{l-1}(1 - g_l). \tag{4.3}$$

In this way, confident regions predicted from the previous higher-level features are preserved and the current level only needs to focus on refining the uncertain region.

In practice, we obtain alpha matte side outputs at three feature levels of stride 8, 4, and 1 respectively (see Fig. 4.2) and slightly dilate the self-guidance masks for a more robust self-guidance. The initial base matte of $1/8$ image size will be progressively upsampled and refined, and the uncertain regions will also shrink gradually through the decoding process using the proposed PRM. The full network is trained end-to-end to auto-balance the refinement focus at multiple feature levels. Such self-guided refinement also makes the model less reliant on external mask guidance, leading to more robust matting performance.

Training Scheme For loss functions, we adopt the l_1 regression loss, composition loss [242], Laplacian loss [97] and denote them as $\mathcal{L}_{l1}$, $\mathcal{L}_{comp}$, $\mathcal{L}_{lap}$ respectively. We represent the ground truth alpha with $\hat{\alpha}$ and prediction alpha with α. The overall loss function is the summation of them:

$$\mathcal{L}(\hat{\alpha}, \alpha) = \mathcal{L}_{l1}(\hat{\alpha}, \alpha) + \mathcal{L}_{comp}(\hat{\alpha}, \alpha) + \mathcal{L}_{lap}(\hat{\alpha}, \alpha). \tag{4.4}$$

The loss is applied to each output head of the network. To make the training more focused on the unknown region, We further modulate the loss with g_l. The final loss function can be formulated as:

$$\mathcal{L}_{final} = \sum_{l} w_l \mathcal{L}(\hat{\alpha_l} \cdot g_l, \alpha_l \cdot g_l), \tag{4.5}$$

where w_l is the loss weight assigned to the outputs of different levels. We use $w_0 : w_1 : w_2 = 1 : 2 : 3$ in our experiments. g_l is generated from α_{l-1} by Eqn. 4.2, and g_0 is a mask filled with one so that the base level output can be supervised over the whole image to provide more holistic semantic guidance for the next level output.

For data augmentation, we follow the training protocol proposed in [131], including random composite two foreground object images, random resize images with random interpolation methods, random affine transformation, and color jitters. We random crop 512×512 patches centered on an unknown region for training. Each patch is composited to a random background image from MS COCO dataset [138].

Guidance Perturbation To ensure that our model can adapt to guidance masks from different sources and with different qualities, we propose a series of guidance perturbations to generate guidance masks from ground-truth alpha matte during training. Given a ground-truth alpha matte, we first binarize it with a random threshold uniformly sampled from 0 to 1. Then, the mask is dilated and/or eroded in random order with random kernel sizes from 1 to 30.

Moreover, we provide a stronger guidance perturbation named CutMask to further improve the model's robustness. Inspired by the successful natural image augmentation CutMix [263], we randomly select a patch size ranging from $1/4$ to $1/2$ image size. Then, two random patches of the guidance are selected and the content of one patch will overwrite another. This stronger perturbation provides additional localized guidance mask corruption, making the model more robust to semantic noises in external guidance masks.

Besides perturbing external guidance masks, we note that perturbing internal self-guidance masks is also very important to improve the robustness. Therefore, we randomly dilate the self-guidance masks to incorporate more variance. Particularly, during training, the self-guidance mask from output stride 8 is dilated by K_1 random sampled from $[1, 30]$ and the one from output stride 4 is dilated by K_2 from $[1, 15]$. For testing, we fix $K_1 = 15$ and $K_2 = 7$.

Figure 4.3: The color labels in the commonly used training data from [242] are noisy and inaccurate, especially near the boundary part. Note that the hair near the ear falsely gets pinker. Best viewed in color and zoomed in.

4.3.2 Foreground Color Estimation

As indicated in Eqn. 4.1, both alpha matte and foreground color need to be solved for foreground object extraction. Nevertheless, only a few matting methods learn to predict the foreground color [97, 188] and all of them used the popular Composition-1k dataset [242] for training.

However, there are a couple of issues in the Composition-1k dataset. First of all, this dataset only contains 431 foreground images with matting and foreground color ground truth, which is quite limited for training a foreground color model. Moreover, the foreground color labels, which were estimated using the color decontamination feature in PhotoShop [242], are sometimes noisy and inaccurate near the boundary regions (see Fig. 4.3). This can introduce color spills and other artifacts into the images during the data augmentation process, making the learning less stable. Besides, labels are only provided where the alpha value is greater than zero, so existing methods can only apply supervision

to the foreground region [97], leading to unstable behaviors in the undefined part.

To address these issues, we propose a simple yet effective method, named Random Alpha Blending (RAB), to generate synthetic training data by blending a foreground image and a background image using a randomly selected alpha matte. Although the composited images may not be semantically meaningful, they can provide accurate and unbiased foreground color labels in the transparent region. The random alpha blending can also significantly make training data more diverse and improve the generalization of foreground color prediction. Besides, we also note that RAB makes it possible to apply loss supervision over all images, leading to a much smoother prediction which is desired for robust compositing. (See Fig. 4.4)

For foreground estimation, we train a separate model using a basic encoder-decoder network, which takes an image and an alpha matte as input. The loss function is the summation of l_1 regression loss, compositing loss, and Laplacian loss. We note that although training a single model for both matte and foreground color prediction is possible, empirically this will degrade the matting performance [97], and the random alpha blending will destroy the semantic cue for the matting model. In addition, decoupling foreground color prediction from matting makes the color model transferable to the use cases where the matte is already given.

4.4 Experiments on Synthetic Datasets

In this section, we report the evaluation results of our method under the traditional synthetic data setting, where the test images are generated using foreground images with ground truth

Methods	SAD	MSE (10^{-3})	Grad	Conn
Learning Based Matting [274]	113.9	48	91.6	122.2
Closed-Form Matting [123]	168.1	91	126.9	167.9
KNN Matting [36]	175.4	103	124.1	176.4
Deep Image Matting [242]	50.4	14	31.0	50.8
IndexNet Matting [156]	45.8	13	25.9	43.7
AdaMatting [19]	41.7	10.2	16.9	-
Context-Aware Matting [97]	35.8	8.2	17.3	33.2
GCA Matting [131]	35.3	9.1	16.9	32.5
$\text{Ours}_{\text{TrimapFG}}$	**31.5**	**6.8**	**13.5**	**27.3**
$\text{Ours}_{\text{Trimap}}$	32.1	7.0	14.0	27.9

Table 4.1: Results on Composition-1k test set. The subscripts denote the corresponding guidance inputs, *i.e.*, TrimapFG, Trimap. The other evaluated methods all require a trimap as input.

mattes and random background images.

Evaluation Metrics We follow previous methods to evaluate the results by Sum of Absolute Differences (SAD), Mean Squared Error (MSE), Gradient (Grad), and Connectivity (Conn) errors using the official evaluation code [242].

Network Architectures We adopt ResNet34-UNet proposed in [131] with an Atrous Spatial Pyramid Pooling (ASPP) [28] as the backbone for both PRN and color prediction. The first convolution layer is adjusted to take a 4-channel input consisting of an RGB image along with an external guidance input. Moreover, an alpha prediction head (Conv-BN-ReLU-Conv) is attached to the features at output stride 4 and 8 respectively to obtain side outputs.

Training stage To fairly compare with previous deep image matting methods, we train our MG Matting model using the Composition-1k dataset [242] which contains 431 foreground objects and the corresponding ground-truth alpha mattes for training. The network is initialized with ImageNet [59] pre-trained weight. We use crop size 512, batch

Methods	SAD	MSE (10^{-3})	Grad	Conn
Learning Based Matting* [274]	105.04	21	94.16	110.41
Closed-Form Matting* [123]	105.73	23	91.76	114.55
KNN Matting* [36]	116.68	25	103.15	121.45
Deep Image Matting* [242]	47.56	9	43.29	55.90
HAttMatting* [176]	48.98	9	41.57	49.93
Deep Image Matting [242]	48.73	11.2	42.60	49.55
+ Ours	**36.58**	**7.2**	**27.37**	**35.08**
IndexNet Matting [156]	46.95	9.4	40.56	46.80
+ Ours	**35.82**	**5.8**	**25.75**	**34.23**
Context-Aware Matting [97]	36.32	7.1	29.49	35.43
+ Ours	**35.04**	**5.4**	**24.55**	**33.35**
GCA Matting [131]	39.64	8.2	32.16	38.77
+ Ours	**35.93**	**5.7**	**25.94**	**34.35**

Table 4.2: Matting refinement results on Distinction-646 test set. Results with $*$ are from methods trained on Distinction-646 train set as reported in [176] for reference. Other results are only trained on composition-1k.

size of 40 in total on 4 GPUs, Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The learning rate is initialized to 1×10^{-3}. The training lasts for $100,000$ iterations with warm-up at the first $5,000$ iterations and cosine learning rate decay [154, 80]. We also apply a curriculum learning manner to help the PRN training. Particularly, for the first $5,000$ iterations, the predictions of output stride 4 and 1 will be guided by guidance mask generated from ground-truth alpha, and for the next $10,000$ iterations, the guidance will be evenly and randomly generated from self-prediction and ground-truth alpha. Afterwards, each alpha prediction should fully rely on its self-guidance. The foreground color prediction is trained under the exactly same settings except that the generated training samples are composited by random foreground and alpha matte. It is noticeable that with RAB, we can add foreground color supervision on the whole image instead of only foreground regions, which produces more smooth and more stable results (see Fig. 4.4).

Methods	SAD	MSE (10^{-3})
Global Matting [90]	220.39	36.29
Closed-Form Matting [123]	254.15	40.89
KNN Matting [36]	281.92	36.29
Context-Aware Matting [97]	61.72	3.24
Ours	**49.80**	**2.48**

Table 4.3: The foreground result ($\alpha \cdot F$) on the Composition-1k dataset.

Methods	Whole Image		Unknown Area	
	SAD	MSE (10^{-3})	SAD	MSE (10^{-3})
Baseline	43.7	4.5	39.8	11.2
Baseline + Deep Supervision	37.8	3.7	36.3	9.5
Baseline + Fusion Conv	38.1	3.2	36.9	8.8
PRN w/o CutMask	33.9	2.9	32.8	7.5
PRN	32.3	2.5	32.1	7.0

Table 4.4: Ablation studies on Composition-1k dataset. Baselines: a ResNet34-UNet with ASPP; Deep supervision: adding side outputs and deep supervisions; Fusion Conv: using convolutions to combine different outputs.

Testing on Composition-1k The test set consists of 50 unique objects which are composited with 20 background images chosen from Pascal VOC [66], thus providing 1000 test samples in total. We note that since these synthetic datasets use PASCAL VOC images as the background which may contain other salient objects, saliency/segmentation models may not be applicable to obtain a reasonable coarse mask. To best fairly compare MG Matting with other trimap-based methods, we test our model under two settings: 1) TrimapFG: We adopt the confident foreground regions in a trimap as a coarse guidance mask for our network; 2) Trimap: We normalize trimap to $[0, 1]$ with the unknown pixels being 0.5 and use this soft mask as guidance. We follow the evaluation setting in Composition-1k which only computes the evaluation on the unknown region.

We summarize the alpha results and foreground color results in Table 4.1 and Table 4.3 respectively. We note that although our model is not trained with trimap, it still shows great robustness and transferability on these unseen types of guidance. Our model surpasses previous state-of-the-art models by a large margin. It also performs consistently considering the gap between trimap and trimapFG. We also note that our foreground color prediction not only reduces the errors significantly, but also produces much smoother results (see Fig. 4.4), which is desired in complex real-world scenarios where alpha matte can be noisy.

Testing on Distinction-646 Distinction-646 [176] is a recent synthetic matting benchmark dataset, which improves the diversity of Composition-1k. It contains 1000 test samples obtained in a similar manner as Composition-1k. However, this dataset is released without official trimaps or other types of guidance, making it difficult to compare with previously reported results. Therefore, we use this benchmark mainly as a testbed to show how our method can refine a matte produced by another method.

We test a few state-of-the-art trimap-based baselines trained on Composition-1k. We first generate trimaps from ground-truth alpha mattes by thresholding and the unknown region is dilated by kernel size 20. Then, we use these trimap-based methods to generate the matting results. Finally, we use these predicted alpha mattes as the guidance to our MG Matting method, and produce refined mattes.

As shown in Table 4.2, using MG Matting as a refinement method consistently improves the results of other state-of-the-art methods. We also show the results reported by [176] in Table 4.2 for reference.

Ablation Studies To validate the design of PRN and the introduced guidance perturbation, we conduct ablations studies as summarized in Table 4.4. Trimap is used as the

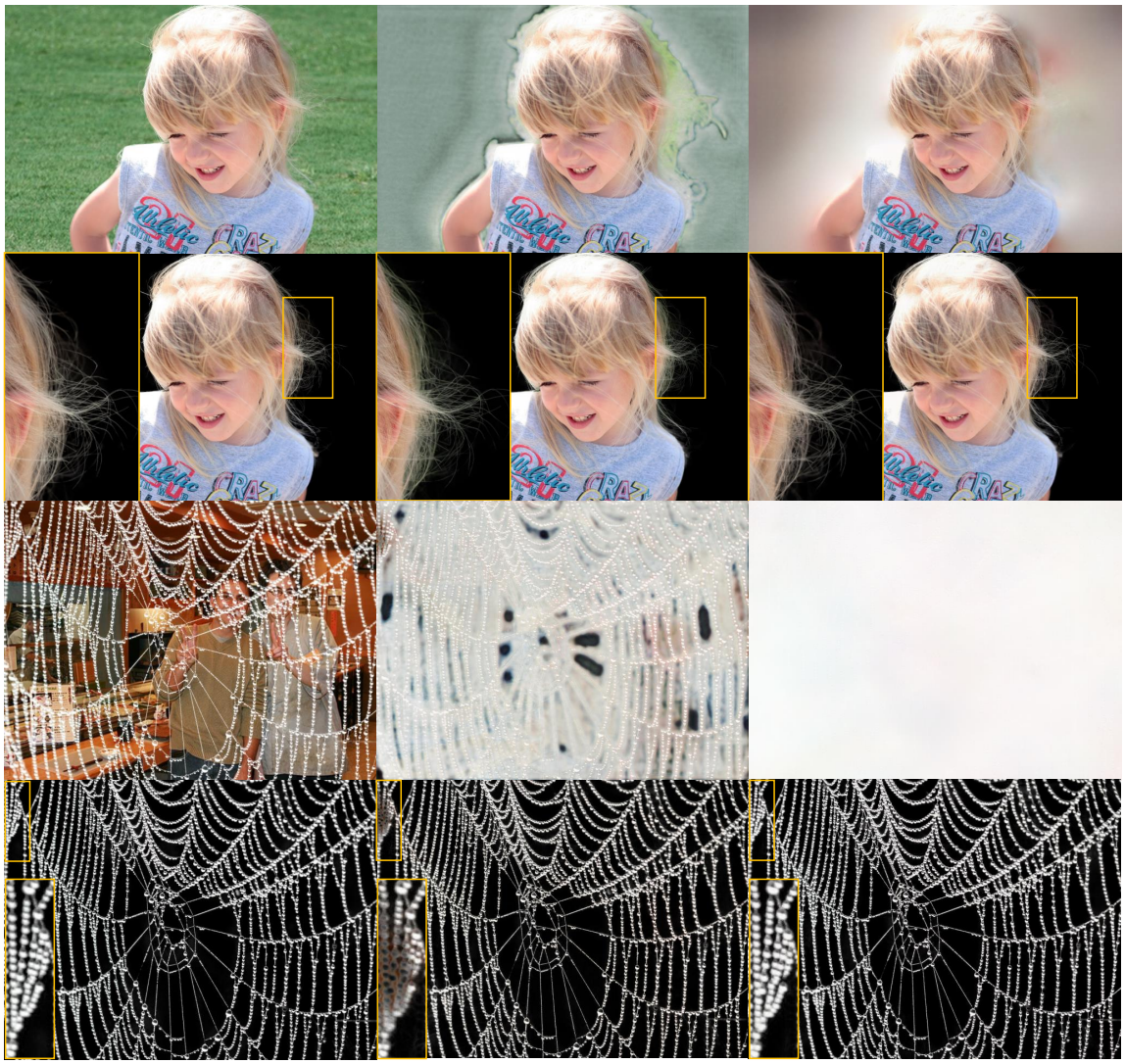

Figure 4.4: A visual comparison of foreground color decontamination. Each column from left to right: Input image and ground truth $\alpha \cdot F$, Foreground color prediction and $\alpha \cdot F$ of [97], predictions of our model with random alpha blending. Note that the background color is mixed into the prediction of [97], while our model can estimate a more smooth foreground color map and be more robust.

guidance mask in these experiments. However, we do not assume that the guidance type is known, so we purposefully do not use it to post-process the prediction by replacing the

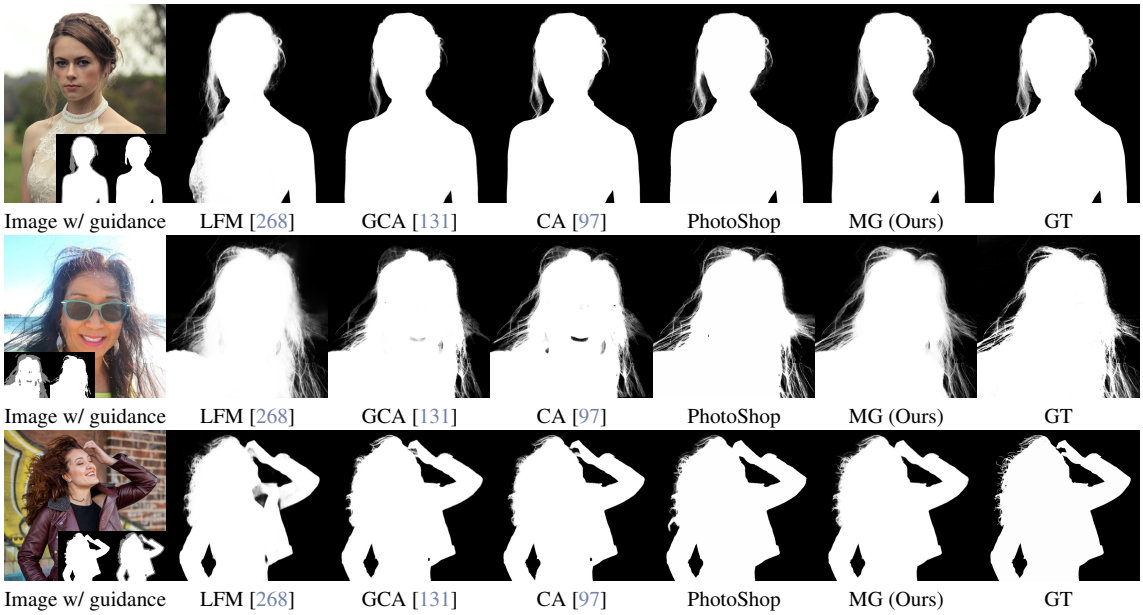

Figure 4.5: The visual comparison results among different methods on our portrait test set. We visualize representative examples with both high-quality studio-level portraits and selfies with strong noises. MG Mating performs well on different-quality images and can maintain details. We note that our results, though only trained on composition-1k, are not only superior to previous state-of-the-art but also produce comparable or better results than commercial methods in PhotoShop.

known foreground and background region. Instead, we report the two scores calculated over the whole image and the unknown region respectively for a more comprehensive evaluation of the robustness of our method.

We report ablations of different variants in Table 4.4. Baseline refers to a pure backbone without any add-ons. Adding side outputs and deep supervision to the baseline improves the performance on both whole image or unknown area. We also try to use two convolution layers to fuse different outputs. However, linearly fusing the side outputs may not lead to better results. In contrast, the proposed PRN can better coordinate the semantic refinement and low-level detail refinement at different levels, thus obtaining a consistent improvement.

Methods	Whole Image		Details	
	SAD	MSE (10^{-3})	SAD	MSE (10^{-3})
Deep Image Matting [242]	28.5	11.7	19.1	74.6
GCA Matting [131]	29.2	12.7	19.7	82.3
IndexNet Matting [156]	28.5	11.5	18.8	72.7
Context-Aware Matting [97]	27.4	10.7	18.2	66.2
Late Fusion Matting [268]	78.6	39.8	24.2	88.3
Ours	**26.8**	**9.3**	**17.4**	**55.1**

Table 4.5: Results on Real-world Portrait test set.

We also show that the CutMask perturbation can further improve both the performance and robustness.

We also validate the effectiveness of RAB. We calculate the MSE and SAD of foreground color ($\mathbf{F}$) over foreground regions (*i.e.* $\alpha > 0$). The baseline achieves MSE $=$ 0.00623 and SAD $=$ 82.30, while with RAB, the performance is boosted to MSE $=$ 0.00321 and SAD $=$ 62.01.

4.5 Experiments on Real-world Portrait Dataset

We note that although the synthetic datasets are well-established benchmarks and provide sufficient data to train a good model, it remains an open question whether models trained on them are robust enough and can produce comparable results in real images. For example, [97] found that some easy data augmentations such as re-JPEGing and gaussian blur can avoid some shortcomings of the synthetic dataset and significantly improve the model's performance on real-world images, though at a cost of higher errors on the synthetic benchmark. This begs the question: *can the results on the synthetic matting dataset reflect the performance on real images?*

Evaluation on real-world images is thus very crucial. However, due to the lack of high-quality matting benchmark datasets of real images, most previous models mainly compare their matting results visually or through a user study. To better evaluate the matting methods in a real-world scenario, we collect a real-world image matting dataset consisting of 637 diverse and high-resolution images with matting annotation made by experts. The images in our dataset have various image qualities and subjects of diverse poses. Moreover, since the dataset mainly contains solid objects where the main body can be easy to predict, we also labeled detail masks covering the hair region and other soft tissues, which tells where the most important details of the image are located. By calculating errors in these regions, we can further compare the ability to capture object details for different models. We will release this dataset for better benchmarking matting methods on real images.

Implementation Details We use the Composition-1k training set to train the model. Considering the semantic gap between the two datasets, we remove the transparent objects from the training data using the data list of [188]. Following [97], we also apply re-JEPGing, gaussian blur, and gaussian noises to the input image to make the model better adapt to real-world noises which are rarely seen in the synthetic dataset. Since these augmentations can change the color of the composited training image, thus the original color label may not be applicable. Therefore, we remove the composition loss from the supervision. Other training settings remain the same as in Sec. 4.4.

For trimap-based baselines, we follow [188] to generate trimaps from segmentation [266] automatically by labeling each pixel with foreground class probability > 0.95 as foreground, < 0.05 as background, and the rest as unknown, the unknown region is further

Figure 4.6: Our model is robust given different quality guidance masks and produces consistent alpha estimation.

dilated by $k = 20$ to ensure it will not miss the long hairs. For our model, we threshold the segmentation at prob $= 0.5$ to a binary mask.

Results We compare the results with state-of-the-art trimap-based methods DIM [242], GCA [131], IndexNet [156], Context-Aware Matting [97], and trimap-free method Late Fusion Matting [268] which is trained on Composition-1k training set and an additional portrait dataset. The results of baselines are obtained through either the open-source inference demos or the provided pre-trained weights.

We summarize the results in Table 4.5 under two settings: Whole Image, where the errors are calculated across the whole image, which can measure the overall quality; Details, where the errors are calculated only in manual-labeled regions containing hair details or other soft areas.

Compared to other methods, our model achieves superior performance, especially regarding the detail part, which illustrates its ability to capture the boundary details. We also note that the trimap-free method LFM performs badly, which could be caused by the fact that their portrait training data is not diverse enough and thus limits the generalizability of their model (see Fig. 4.5 for examples).

We compare our results with another trimap-free method BSHM [142]. We contacted the authors and obtained the test results on a 100 images subset of our portrait dataset. Since [142] can only deal with low-resolution images, we downsample images to longer-side 720, and the metrics are also computed on this scale. [142] achieves MSE 0.0155 and SAD 10.66 for whole image and MSE 0.0910 and SAD 7.60 for detail regions, while our MG Matting obtains a superior performance with MSE 0.0095 and SAD 8.01 for whole image and MSE 0.0637 and SAD 5.94 for details.

Robustness to Guidance. To verify how robust our model is to the external guidance mask, we conduct experiments to feed the network with perturbed external guidance masks. Particularly, we erode/dilate the mask with kernel size 10, 20, 30 respectively. We note that the model predicts consistently given differently perturbed external guidance. The SAD error increases from 26.8 to 27.1, 27.2, 27.4 with mask eroded by 10, 20, and 30 respectively. For dilation, the SAD error goes to 27.0, 27.4, 28.1 with kernel 10, 20, 30 respectively. A visual example is provided in Fig. 4.6.

4.6 Conclusion

In this chapter, we present Mask Guided (MG) Matting, a general framework to resolve the natural image matting problem. Unlike previous methods, our method is not tailored to some specific guidance mask. Instead, it can handle versatile guidance masks such as a trimap, a rough segmentation mask, or a low-quality alpha matte. The key to the robustness of our model lies in the Progressive Refinement Network, which provides self-guidance and progressively refines the uncertain regions during the decoding process. Further, we also propose a simple yet effective method called Random Rendering to resolve the limitation of the existing dataset and learn a better foreground color estimation model, which is important yet rarely studied before. Moreover, we release a new real-world matting dataset with high-quality labels to better quantitatively evaluate matting models in a real-world scenario, which we hope could shed some light on the direction towards real-life matting.

Chapter 5

Glance-and-Gaze Vision Transformer

In this chapter, we discuss our efforts in vision transformer, which shows great potential and serves as strong alternative to CNNs. However, due to its high complexities to the length of input sequence, it is limited for dense tasks such as segmentation, where high-resolution inputs/features are expected. Here we propose an efficient and effective alternative, named Glance-and-Gaze attention module, to the original self-attention. It sparsely glances over the image so not all pixels need to be involved, and compensates the missing local details efficiently through a depthwise convolution in gaze branch. It shows promising results on several public benchmarks.

5.1 Introduction

Convolution Neural Networks (CNNs) have been dominating the field of computer vision, which have been a de-facto standard and achieved tremendous success in various tasks, *e.g.*, image classification [92], object detection [89], semantic segmentation [31], *etc*. CNNs model images from a local-to-global perspective, starting with extracting local features

such as edges and textures, and forming high-level semantic concepts gradually. Although CNNs prove to be successful for various vision tasks, they lack the ability to globally represent long-range dependencies. To compensate a global view to CNN, researchers explored different methods such as non-local operation [225], self-attention [214], Atrous Spatial Pyramid Pooling (ASPP) [31].

Recently, another type of networks with stacked Transformer blocks emerged. Unlike CNNs, Transformers naturally learn global features in a parameter-free manner, which makes them stronger alternatives and raises questions about the necessity of CNNs in vision systems. Since the advent of Vision Transformer (ViT) [63], which applied Transformers to vision tasks by projecting and tokenizing natural images into sequences, various improvements have been introduced rapidly, *e.g.*, better training and distillation strategies [207], tokenization [262], position encoding [49], local feature learning [85]. Moreover, besides Transformers' success on image classification, many efforts have been made to explore Transformers for various down-stream vision tasks [224, 149, 67, 24, 272].

Nevertheless, the advantages of Transformers come at a price. Since self-attention operates on the whole sequences, it incurs much more memory and computation costs than convolution, especially when it comes to natural images, whose lengths are usually much longer than word sequences, if treating each pixel as a token . Therefore, most existing works have to adopt a compromised strategy to embed a large image patch for each token, although treating smaller patches for tokens leads to a better performance (*e.g.*, ViT-32 compared to ViT-16 [63]). To address this dilemma, various strategies have been proposed. For instance, Pyramid Vision Transformer (PVT) [224] introduced a progressive shrinking pyramid to reduce the sequence length of the Transformer with the increase of

network depth, and adopted spatial-reduction attention, where *key* and *value* in the attention module are down-sampled to a lower resolution. Swin-Transformer [149] also adopted the pyramid structure, and further proposed to divide input feature maps into different fix-sized local windows, so that self-attention is computed within each window, which reduces the computation cost and makes it scalable to large image scales with linear complexity.

Nonetheless, we notice that these strategies have some limitations: Spatial-reduction attention can reduce memory and computation costs to learn high-resolution feature maps, yet with a price of losing details which are expected from the high-resolution feature maps. Adopting self-attention within local windows is efficient with linear complexity, but it sacrifices the most significant advantage of Transformers in modeling long-range dependencies.

To address these limitations, we propose Glance-and-Gaze Vision Transformer (**GG-Transformer**), inspired by the Glance-and-Gaze human behavior when recognizing objects in natural scenes [60], which takes advantage of both the long-range dependency modeling ability of Transformers and locality of convolutions in a complementary manner. A GG-Transformer block consists of two parallel branches: A Glance branch performs self-attention within adaptively-dilated partitions of input images or feature maps, which preserves the global receptive field of the self-attention operation, meanwhile reduces its computation cost to a linear complexity as local window attention [149] does; A Gaze branch compensates locality to the features obtained by the Glance branch, which is implemented by a light-weight depth-wise convolutional layer. A merging operation finally re-arranges the points in each partition to their original locations, ensuring that the output of the GG-Transformer block has the same size as the input. We evaluate GG-Transformer

on several vision tasks and benchmarks including image classification on ImageNet [59], object detection on COCO [138], and semantic segmentation on ADE20K [277], and show its efficiency and superior performance, compared to previous state-of-the-art Transformers.

5.2 Related Work

CNN and self-attention. Convolution has been the basic unit in deep neural networks for computer vision problems. Since standard CNN blocks were proposed in [120], researchers have been working on designing stronger and more efficient network architectures, *e.g.*, VGG [194], ResNet [92], MobileNet [187], and EfficientNet [202]. In addition to studying how to organize convolutional blocks into a network, several variants of the convolution layer have also been proposed, *e.g.*, group convolution [118], depth-wise convolution [47], and dilated convolution [249]. With the development of CNN architectures, researchers also seeked to improve contextual representation of CNNs. Representative works, such as ASPP [31] and PPM [270] enhance CNNs with multi-scale context, and NLNet [225] and CCNet [102] provided a non-local mechanism to CNNs. Moreover, instead of just using them as an add-on to CNNs, some works explored to use attention modules to replace convolutional blocks [100, 179, 218, 269].

Vision Transformer. Recently, ViT [63] was proposed to adapt the Transformer [214] for image recognition by tokenizing and flattening 2D images into sequence of tokens. Since then, many works have been done to improve Transformers, making them more suitable for vision tasks. These works can be roughly categorized into three types: (1) **Type I** made efforts to improve the ViT design itself. For example, DeiT [207] introduced a training scheme to get rid of large-scale pre-training and distillation method to further improve the

performance. T2T-ViT [262] presented a token-to-token operation as alternatives to patch embedding, which keeps better local details. (2) **Type II** tried to introduce convolution back into the ViT design. *E.g.*, Chu *et al.* [49] proposed to use convolution for position encoding. Wu *et al.* [230] used convolution to replace the linear projection layers in Transformers. (3) **Type III** tried to replace CNNs by building hierarchical Transformers as a plug-in backbone in many downstream tasks. Wang *et al.* [224] proposed a pyramid vision Transformer, which gradually downsamples the feature map and extract multi-scale features as common CNN backbones do. However, applying self-attention on high-resolution features is not affordable in terms of both memory and computation cost, thus they used spatial-reduction attention, which downsamples *key* and *value* in self-attention as a trade-off between efficiency and accuracy. Later, Liu *et al.* [149] proposed a new hierarchical Transformer architecture, named Swin-Transformer. To handle the expensive computation burden incurred with self-attention, they divided feature maps into several non-overlapped windows, and limited the self-attention operation to be performed within each window. By doing so, Swin-Transformer is more efficient and also scalable to large resolution input. Besides, to compensate the missing global information, a shifted window strategy is proposed to exchange information between different windows.

Our method differs from aforementioned works in the following aspects: Type I, II methods usually utilize a large patch size and thus incompatible to work with high-resolution feature map. Type III methods proposed new attention mechanism to handle the extreme memory and computation burden with long sequences, but they sacrifices accuracy as a trade-off with efficiency. In contrast, GG-Transformer proposes a more efficient Transformer block with a novel Glance-and-Gaze mechanism, which not only

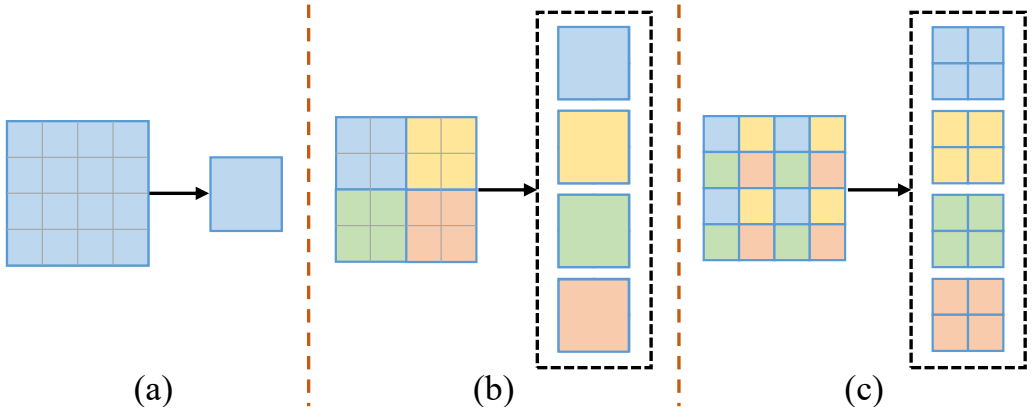

Figure 5.1: Toy examples illustrating different methods to reduce computation and memory cost of self-attention. (a) Spatial reduction [224, 67] spatially downsamples the feature map; (b) Local window [149] restricts self-attention inside local windows; (c) Glance attention (ours) applies self-attention to adaptively-dilated partitions.

enables it to handle long sequences and scalable to high-resolution feature maps, but also leads to a better performance than other efficient alternatives.

5.3 Method

The design of GG-Transformer draws inspiration from how human beings observe the world, which follows the *Glance and Gaze* mechanism. Specifically, humans will glance at the global view, and meanwhile gaze into local details to obtain a comprehensive understanding to the environment. We note that these behaviors surprisingly match the property of self-attention and convolution, which models long-range dependencies and local context, respectively. Inspired from this, we propose GG-Transformer, whose Transformer block consists of two parallel branches: A Glance branch, where self-attention is performed to adaptively-dilated partitions of the input, and a Gaze branch, where a depth-wise convolutional layer is adopted to capture the local patterns.

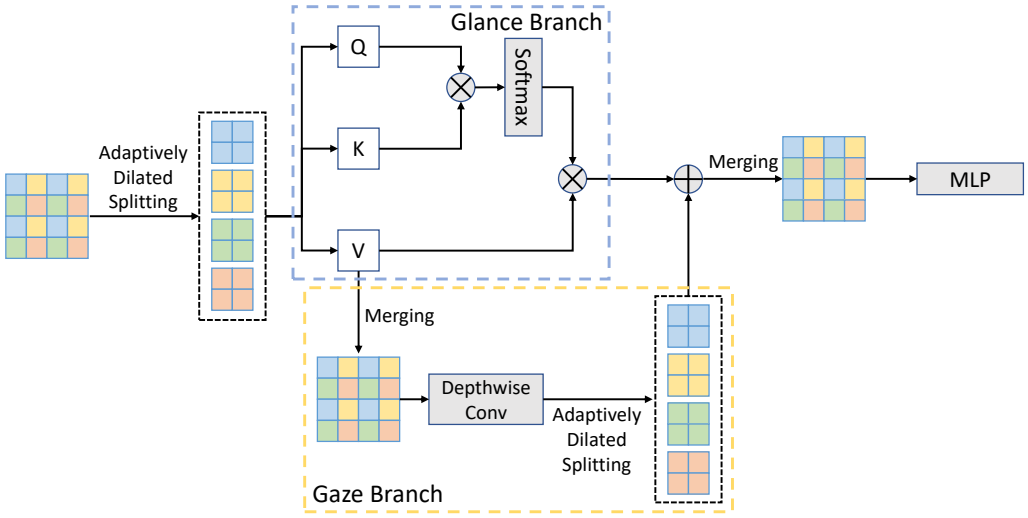

Figure 5.2: A visual illustration of GG Transformer block, where the Glance and Gaze branches parallely extract complementary information.

5.3.1 Revisit Vision Transformer

We start with revisiting the formulation of vision Transformer block, which consists of multi-head self-attention (MSA), layer normalization (LN), and multi-layer perceptron (MLP). A Transformer block processes input features as follows:

$$\mathbf{z}'_\ell = \mathrm{MSA}(\mathrm{LN}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}, \tag{5.1}$$

$$\mathbf{z}_\ell = \mathrm{MLP}(\mathrm{LN}(\mathbf{z}'_\ell)) + \mathbf{z}'_\ell, \tag{5.2}$$

where $\mathbf{z}_\ell$ is the encoded image representation at the ℓ-th block. MSA gives Transformers the advantages of modeling a global relationship in a parameter-free manner, which is formulated as:

$$\mathrm{MSA}(X) = \mathrm{Softmax}(\frac{QK^T}{\sqrt{C}})V, \tag{5.3}$$

where Q, K, $V \in \mathbf{R}^{N \times C}$ are the *query*, *key*, and *value* matrices which are linear mappings of input $X \in \mathbf{R}^{N \times C}$, C is the channel dimension of the input, and N is the length of input sequence. Note that for simplified derivation, we assume the number of heads is 1 in the multi-head self attention, which will not affect the following complexity analysis and can be easily generalize to more complex cases.

For vision tasks, N is often related to the input height H and width W. In practice, a 2D image is often first tokenized based on non-overlapping image patch grids, which maps a 2D input with size $H \times W$ into a sequence of token embeddings with length $N = \frac{H \times W}{P^2}$, where (P, P) is the grid size. In MSA, the relationships between a token and all tokens are computed. Such designs, though effectively capturing long-range features, incur a computation complexity quadratic to N:

$$\Omega(\text{MSA}) = 4NC^2 + 2N^2C. \tag{5.4}$$

For ViTs that only work on $16\times$ down-sampled feature maps (*i.e.*, $P = 16$), the computation cost is affordable, since in this scenario $N = 14 \times 14 = 196$ (for a typical ImageNet setting with input size 224×224). However, when it comes to a more general vision scenario with the need of dense prediction based on high-resolution feature maps (such as semantic segmentation), where the cost dramatically increases by thousands of times or even more. Naively applying MSA to such high-resolution feature maps can easily lead to the problem of out-of-memory (OOM), and extremely high computational cost. Although some efficient alternatives [224, 149] were brought up recently, accuracy is often sacrificed as a trade-off of efficiency. To address this issue, we propose a new vision Transformer that can be applied to longer sequence while keeping high accuracy, inspired by the Glance-and-Gaze

human behavior when recognizing objects in natural scenes [60].

5.3.2 Glance: Efficient Global Modeling with Adaptively-dilated Splitting

To address the efficiency problem of Transformers, existing solutions often adapt Transformers to high-resolution feature maps by down-sampling the key and value during the attention process [224], or limit self-attention to be computed in a local region then exchange information through shifting these local regions to mimic a global view [149]. But limitations exist in these methods. For down-sampling methods, although the output feature maps keep to be high-resolution, they lose some details during the down-sampling processes. Besides, they still has a quadratic complexity and thus may not scale up to a larger input size. For local-region based methods, though they successfully reduce the complexity to a linear level, they cannot directly model a long-range dependency but instead are stuck within local context, which counters the design intuition of Transformer and self-attention for long-range dependency modeling. Besides, two consecutive blocks need to work together to mimic a global receptive field, which may not achieve as good performance as MSA (see Table. 5.5).

Thus, we propose Glance attention, which performs self-attention efficiently with a global receptive field. It shares same time complexity as [149], but directly models long-range dependencies, as shown in Fig. 5.1. Specifically, we first splits an input feature map to several dilated partitions, *i.e.*, the points in a partition are not from a local region but from the whole input feature map with a dilation rate adaptive to the feature map size and the token size. We name this operation *Adaptively-dilated Splitting*. For example, a partition

contains $M \times M$ tokens and it is obtained with an adaptive dilation rate = $(\frac{h}{M}, \frac{w}{M})$, where h, w is the height and width of current feature map respectively, and $hw = N$. Here we assume all divisions have no remainder for simplicity. These partitions are easily to be split from the input feature map or merged back. Specifically, we formulate this process as follows:

$$\mathbf{z}_{\ell-1} = [\mathbf{z}_{\ell-1}^{1,1}, \mathbf{z}_{\ell-1}^{1,2}, \ldots, \mathbf{z}_{\ell-1}^{h,w}], \tag{5.5}$$

where $\mathbf{z}_{\ell-1}^{i,j}$ is the feature token at position (i, j) if reshaping the sequence of token embedding $\mathbf{z}_{\ell-1}$ back into a 2D feature map and use the 2D coordinates accordingly. To reduce the memory and computation burden, while keeping a global receptive field, $\mathbf{z}_{\ell-1}$ is split into several partitions:

$$\mathbf{z}_{\ell-1}^{\dagger} = \text{AdaptivelyDilatedSplitting}(\mathbf{z}_{\ell-1}) \tag{5.6}$$

$$= [\mathbf{z}_{\ell-1}^{\dagger,1,1}, \mathbf{z}_{\ell-1}^{\dagger,1,2}, \ldots, \mathbf{z}_{\ell-1}^{\dagger,\frac{h}{M},\frac{w}{M}}], \tag{5.7}$$

$$\text{where } \mathbf{z}_{\ell-1}^{\dagger,i,j} = [\mathbf{z}_{\ell-1}^{i,j}, \mathbf{z}_{\ell-1}^{i,j+\frac{h}{M}}, \ldots, \mathbf{z}_{\ell-1}^{i+\frac{(M-1)h}{M},j+\frac{(M-1)w}{M}}], \tag{5.8}$$

where $\mathbf{z}_{\ell-1}^{\dagger} \in \mathbf{R}^{N\times C}$, $\mathbf{z}_{\ell-1}^{\dagger,i,j} \in \mathbf{R}^{M^2\times C}$. Afterwards, MSA is applied to each partition, which substantially reduces the computation and memory cost yet does not lose the global feature representation. And then all partitions are merged back into one feature map and go

through the remaining modules:

$$\mathbf{z}_{\ell}^{\prime,i,j} = \mathrm{MSA}(\mathrm{LN}(\mathbf{z}_{\ell-1}^{\dagger,i,j})) + \mathbf{z}_{\ell-1}^{\dagger,i,j}, \tag{5.9}$$

$$\mathbf{z}_{\ell}^{\prime} = \mathrm{Merging}(\mathbf{z}_{\ell}^{\prime,1,1}, \ldots, \mathbf{z}_{\ell}^{\prime,\frac{h}{M},\frac{w}{M}}), \tag{5.10}$$

$$\mathbf{z}_{\ell} = \mathrm{MLP}(\mathrm{LN}(\mathbf{z}_{\ell}^{\prime})) + \mathbf{z}_{\ell}^{\prime}, \tag{5.11}$$

where Merging is an inverse operation of AdaptivelyDilatedSplitting which re-arranges points in each partition back in their original orders.

This new self-attention module (formulated in Eq. 5.6 to 5.10), namely Glance multi-head self attention module (G-MSA), enables a global feature learning with linear complexity:

$$\Omega(\text{G-MSA}) = 4hwC^2 + 2M^2hwC = 4NC^2 + 2M^2NC. \tag{5.12}$$

5.3.3 Gaze: Compensating Local Relationship with Depthwise Convolution

Although the Glance branch can effectively capture long-range representations, it misses the local connections across partitions, which can be crucial for vision tasks relying on local cues. To this end, we propose a Gaze branch to compensate the missing relationship and enhance the modeling power at a negligible cost.

Specifically, to compensate the local patterns missed in the Glance branch, we propose to apply an additional depthwise convolution on the *value* in G-MSA:

$$\mathrm{Gaze}(X) = \mathrm{DepthwiseConv2d}(\mathrm{Merging}(V)), \tag{5.13}$$

which has a neglectable computational cost and thus the overall cost is still significantly reduced:

$$\Omega(\text{GG-MSA}) = 4NC^2 + 2M^2NC + k^2NC, \tag{5.14}$$

where k is the Gaze branch kernel size, and M is the partition size set in Glance branch, both k and M are constants that are much smaller than N. We note that in this way, long-range and short-range features are naturally and effectively learned. Besides, unlike [149], GG-MSA does not require two consecutive blocks (*e.g.*, W-MSA and SW-MSA) to be always used together, instead, it is a standalone module as the original MSA [63].

We propose two ways to determine the kernel size k for better compensating the local features:

Fixed Gazing. A straightforward way is to adopt the same kernel size (*e.g.*, 3×3) for all Gaze branches, which can ensure same local feature learning regardless of the dilation rate.

Adaptive Gazing. Another way is implementing Gazing branch with adaptive kernels, where the kernel size should be the same as dilation rate $(h/M, w/M)$. In this way, GG-MSA still enjoys a complete view of the input.

By combining Glance and Gaze branches together, GG-MSA can achieve superior performance to other counterparts while remaining a low cost.

5.3.4 Network Instantiation

We build a hierarchical GG-Transformer with the proposed Glance-and-Gaze branches as shown in Fig. 5.2. For fair comparison, we follow the settings in Swin-Transformer [149] in terms of network depth and width, with only difference in the attention methods used

in Transformer blocks. Furthermore, we set M to be same as the window size in [149], so that the model size and computation cost are also directly comparable. Note that GG-Transformer has not been specifically tuned by scaling depth and width for a better accuracy-cost trade-off.

We build GG-T and GG-S, which share the same model size and computation costs as Swin-T and Swin-S, respectively. For all GG-Transformers, we set the fixed patch size $M = 7$, expansion ratio of MLP $\alpha = 4$. All GG-Transformer consists of 4 hierarchical stages, which corresponds to feature maps with down-sampling ratio 4, 8, 16, 32, respectively. The first patch embedding layer projects input to a feature map with channel $C = 96$. When transitioning from one stage to the next one, we follow CNN design principles [92] to expand the channel by $2\times$ when the spatial size is down-sampled by $4\times$.

5.4 Experiments

In the following parts, we report results on ImageNet [59] classification, COCO [138] object detection, and ADE20K [277] semantic segmentation to compare GG-Transformer with those state-of-the-art CNNs and ViTs. Afterwards, we conduct ablation studies to verify the design of Glance and Gaze branches and also compare effectiveness of different alternative self-attention designs.

5.4.1 ImageNet Classification

We validate the performance of GG-Transformer on ImageNet-1K [59] classification task, which contains 1.28M training images and 50K validation images for 1000 classes. We report top-1 accuracy with a single 224×224 crop.

method	image size	#param.	FLOPs	ImageNet top-1 acc.
RegNetY-4G [178]	224^2	21M	4.0G	80.0
RegNetY-8G [178]	224^2	39M	8.0G	81.7
RegNetY-16G [178]	224^2	84M	16.0G	82.9
EffNet-B3 [202]	300^2	12M	1.8G	81.6
EffNet-B4 [202]	380^2	19M	4.2G	82.9
EffNet-B5 [202]	456^2	30M	9.9G	83.6
DeiT-T [207]	224^2	5M	1.3G	72.2
DeiT-S [207]	224^2	22M	4.6G	79.8
DeiT-B [207]	224^2	86M	17.5G	81.8
TNT-S [85]	224^2	24M	5.2G	81.3
TNS-B [85]	224^2	66M	14.1G	82.8
T2T-ViT-7 [262]	224^2	4M	1.2G	71.7
T2T-ViT-14 [262]	224^2	22M	5.2G	81.5
T2T-ViT-24 [262]	224^2	64M	14.1G	82.3
PVT-Tiny [224]	224^2	13M	1.9G	75.1
PVT-Small [224]	224^2	25M	3.8G	79.8
PVT-Medium [224]	224^2	44M	6.7G	81.2
PVT-Large [224]	224^2	61M	9.8G	81.7
Swin-T [149]	224^2	28M	4.5G	81.2
Swin-S [149]	224^2	50M	8.7G	83.2
GG-T	224^2	28M	4.5G	82.0
GG-S	224^2	50M	8.7G	83.4

Table 5.1: Comparison of different models on ImageNet-1K classification.

Implementation Details. To ensure a fair comparison, we follow the same training settings of [149]. Specifically, we use AdamW [155] optimizer for 300 epochs with cosine learning rate decay including 20 epochs for linear warm-up. The training batch size is 1024 with 8 GPUs. Initial learning rate starts at 0.001, and weight decay is 0.05. Augmentations and regularizations setting follows [207] including rand-augment [56], mixup [265], cutmix [263], random erasing [275], stochastic depth [101], but excluding repeated repeated augmentation [96] and EMA [172].

Results. A summary of results in Table 5.1, where we compare GG-Transformer with various CNNs and ViTs. It is shown that GG-Transformer achieve better accuracy-cost trade-off compared to other models. Moreover, GG-T, a light-weight model (28M/4.5G/82.0%), can achieve comparable performance to those even much large models such as DeiT-B (86M/17.5G/81.8%), T2T-ViT-24 (64M/14.1G/82.3%), and PVT-Large (61M/9.8G/81.7%). Furthermore, compared to Swin-Transformer, which we follows the architecture and ensures the same model size and computation costs to ensure a fair comparison, our model consistently brings an improvement to baseline, with a consistent improvement of 0.8% and 0.2% for T and S models respectively.

5.4.2 ADE20K Semantic Segmentation

ADE20K [277] is a challenging semantic segmentation dataset, containing 20K images for training and 2K images for validation. We follow common practices to use the training set for training and report mIoU results on the validation sets. We use UperNet [234] as the segmentation framework and replace the backbone with GG-Transformer.

Implementation Details. We follow [149] and use MMSegmentation [53] to implement all related experiments. We use AdamW [155] with a learning rate starting at 6×10^{-5}, weight decay of 0.01, batch size of 16, crop size of 512×512. The learning rate schedule contains a warmup of 1500 iterations and linear learning rate decay. The training is conducted with 8 GPUs and the training procedure lasts for 160K iterations in total. The augmentations follows the default setting of MMSegmentation, including random horizontal flipping, random re-scaling within ratio range [0.5, 2.0] and random photometric distortion. For testing, we follow [272] to utilize a sliding window manner with crop size 512 and stride

Backbone	UperNet			
	Prams (M)	FLOPs (G)	mIoU (%)	mIoU(ms+flip) (%)
ResNet50 [92]	67	952	42.1	42.8
PVT-Small [224]	55	919	43.9	44.8
Swin-T [149]	60	941	44.5	45.8
GG-T (ours)	60	942	46.4	47.2
ResNet101 [92]	86	1029	43.8	44.9
PVT-Medium [224]	74	977	44.9	45.3
Swin-S [149]	81	1034	47.6	49.5
GG-S (ours)	81	1035	48.4	49.6

Table 5.2: Performance comparisons with different backbones on ADE20K validation dataset. FLOPs is tested on 1024×1024 resolution. All backbones are pretrained on ImageNet-1k.

341. ImageNet-1K pretrained weights are used for initialization.

Results. We show results in Table ??, where results both w/ and w/o test-time augmentation are reported. Noticeably, GG-Transformer not only achieves better results to baselines, but also obtain a comparable single-scale testing performance to those with multi-scale testing results. Specifically, GG-T achieves 46.4% mIoU with single-scale testing, which surpasses ResNet50, PVT-Small, Swin-T's multi-scale testing results by 3.6%, 1.6%, 0.6%, respectively. Moreover, our tiny model even can be comparable to those much larger models (*e.g.*, 47.2% of GG-T compared to 47.6% of Swin-S).

5.4.3 COCO Object Detection

We further verify the performance of GG-Transformer when used as a plug-in backbone to object detection task on COCO dataset [138], which contains 118K, 5K, 20K images for training, validation and test respectively. We use Mask-RCNN [89] and Cascaded Mask R-CNN [20] as the detection frameworks, and compare GG-Transformer to various CNN and ViT backbones.

Backbone	P (M)	F (G)	Mask R-CNN						Cascaded Mask R-CNN					
			AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
ResNet50 [92]	44	260	38.2	58.8	41.4	34.7	55.7	37.2	41.2	59.4	45.0	35.9	56.6	38.4
PVT-Small [224]	44	245	40.4	62.9	43.8	37.8	60.1	40.3	-	-	-	-	-	-
Swin-T [149]	48	264	43.7	66.6	47.7	39.8	63.3	42.7	48.1	67.1	52.2	41.7	64.4	45.0
GG-T (ours)	48	265	44.1	66.7	48.3	39.9	63.3	42.4	48.4	67.4	52.3	41.9	64.5	45.0
ResNet101 [92]	63	336	40.0	60.6	44.0	36.1	57.5	38.6	42.9	61.0	46.6	37.3	58.2	40.1
ResNeXt101-32×4d [237]	63	340	41.9	62.5	45.9	37.5	59.4	40.2	44.3	62.8	48.4	38.3	59.7	41.2
PVT-Medium [224]	64	302	42.0	64.4	45.6	39.0	61.6	42.1	-	-	-	-	-	-
Swin-S [149]	69	354	45.4	67.9	49.6	41.4	65.1	44.6	49.7	68.8	53.8	42.8	66.0	46.4
GG-S (ours)	69	355	45.7	68.3	49.9	41.3	65.3	44.0	49.9	69.0	54.0	43.1	66.2	46.4

Table 5.3: Object detection and instance segmentation performance on the COCO `val2017` dataset using the Mask R-CNN framework. P(Params)/F(FLOPs) is evaluated with Mask R-CNN architecture on a 1280×800 image.

Implementation Details. We follow the setting of [149] and use MMDetection [25] to conduct all the experiments. We adopt multi-scale training [21], AdamW optimizer [155] with initial learning rate of 0.0001, weight decay of 0.05, batch size of 16. The training is conducted with 8 GPUs and a 1× schedule. All models are initialized with ImageNet-1K pretrained weights.

Results. As shown in Table 5.3, GG-Transformer achieves superior performance to other backbones in the two widely-used detection frameworks. Specifically, GG-T achieves 44.1 box AP and 39.9 mask AP, which surpasses both CNNs and other ViTs with a similar model size and computation costs. Compared with the state-of-the-art Swin-Transformer, GG-Transformer achieves better performance while keeping the same model size and computation costs for both T and S models.

Gaze Kernel	Top-1
Fixed-(3,3,3,3)	80.28%
Fixed-(5,5,5,5)	80.31%
Adaptive-(9,5,3,3)	80.38%

Table 5.4: Choices of Gaze Kernels.

	Top-1
W& SW-MSA [149]	78.50%
MSA	79.79%
Glance Only	77.21%
Gaze Only	76.76%
Glance+Gaze (Attn)	79.07%
Glance+Gaze (Conv)	80.28%

Table 5.5: Comparison among different self-attentions. Gaze (Conv) uses kernels of Fixed-(3,3,3,3).

	Top-1
DeiT-T	72.2%
GG-DeiT-T	73.8%
DeiT-S	79.9%
GG-DeiT-S	80.5%

Table 5.6: Applying GG-MSA to DeiT backbone.

5.4.4 Ablation Studies

In this part, we conduct ablation studies regarding to the designs of GG-Transformer. Meanwhile, we also compare among different efficient alternatives to MSA. Besides, we verify GG-MSA on another ViT architecture [207] to compare its capacity to MSA directly. We conduct all these experiments based on Swin-T [149] with 100 epochs training and DeiT [207] architectures with 300 epochs training.

Kernel Choice of Gaze Branch. We study the choice of Gaze branch in terms of fixed or adaptive mechanism. The kernel sizes for each stage and results are summarized in Table 5.4, where we observe that both mechanisms work well. Using a larger kernel leads

to a non-significant improvement. In contrast, adaptive manner leads to a slightly better performance. Considering the adaptive manner provides a complete view as the original MSA has, we choose it in our final design.

Glance/Gaze Branch. We study the necessity of both Glance and Gaze branches. Meanwhile, a comparison between different ways to conduct self-attention is also studied. Results are in Table 5.5.

Swin-T [149] serves as the baseline for all variants, which achieves 78.50% top-1 accuracy on ImageNet validation set. Firstly, we note that the local window attention and shifted window attention (W&SW-MSA) in [149] although can significantly reduce the computation complexity and makes Transformer easier to scale-up, it sacrifices the accuracy and the combination of W&SW-MSA to mimic a global view is not as good as the original MSA. We replace the W&SW-MSA with MSA for all blocks in stage 3 and 4 (*i.e.*, stages with down-sampling rate 16 and 32), which leads to a 1.29% performance improvement, indicating there exists a significant performance gap between MSA and its efficient alternative. Notably, when adopting the proposed Glance and Gaze mechanism instead, which shares a same complexity of W& SW-MSA, can achieves much better performance, where the Glance+Gaze (Attn) improves the performance by 0.57%, and Glance+Gaze (Conv) (*i.e.*, GG-T) by 1.78%, which is even higher than MSA by 0.49%.

Besides using depthwise convolution, another natural choice is to also adopt self-attention for implementing the Gaze branch. Therefore, we conduct experiments by using local window attention [149] as the Gaze branch. Note that, unlike depthwise convolution, a self-attention variant of the Gaze branch cannot be integrated with the Glance branch into the same Transformer block while keeping the overall model size and computation cost at

the same level. To ensure a fair comparison, we use two consecutive Transformer blocks where one is Glance attention and another is Gaze attention. Using either convolution or self-attention to implement the Gaze branch can both improve the performance compared to [149], illustrating the effectiveness of the Glance and Gaze designs. However, using self-attention is inferior to depth-wise convolution with a degrade of 1.21%, which may indicate that convolution is still a better choice when it comes to learning local relationships. Besides, using depth-wise convolution as Gaze branch can also naturally be integrated into the Transformer block with Glance attention, thus makes it more flexible in terms of network designs.

We also note that Glance or Gaze branch alone is far from enough, while only a combination of both can lead to a performance gain, which matches the behavior that we human beings can not rely on Glance or Gaze alone. For instance, using Glance alone can only lead to an inferior performance with accuracy of 77.21%, and Gaze alone 76.76%, which is significantly lower than baseline with a degrade of 1.29% and 1.74%, respectively. Nevertheless, we note that this is because Glance and Gaze branches miss important local or global cues which can be compensated by each other. As a result, a combination of both Glance and Gaze gives a high accuracy of 80.28%, which improves the Glance alone and Gaze alone by 3.07% and 3.52% respectively.

Apply to other backbone. We verify the effectiveness of GG-Transformer on another popular ViT architecture DeiT [207], as shown in Table 5.5. We replace MSA with GG-MSA for two DeiT variants [207], DeiT-T and DeiT-S. We show that, although GG-MSA is an efficient alternative to MSA, it can also lead to a performance gain. Compared to DeiT-T and DeiT-S, GG-DeiT-T and GG-DeiT-S bring the performance up by 1.6% and

0.6% respectively, illustrating that it is not only efficient but also effectively even compared to a fully self-attention.

Network runtime. We follow [149] to report and compare work runtime measured by FPS: GG-T achieves 782.34 FPS compared to 737.86 of Swin-T, and GG-S achieves 441.31 FPS compared to 423.51 of Swin-S. The evaluation is done with a single Nvidia tesla v100-sxm2-16gb GPU.

5.5 Limitation

Although GG-Transformer provides a powerful and efficient solution to make Transformers scalable to large inputs, some limitations still exist and worth further exploring.

Firstly, over-fitting is a common problem [63] in Vision Transformers and can be alleviated by large-scale pretraining [63] or strong augmentations and regularization [207]. This problem is more serious for stronger models (GG-Transformer) and in the tasks with relatively small dataset (e.g. semantic segmentation). Secondly, Transformers suffer from performance degradation in modeling longer-range dependencies, when there exists large discrepancy in the training-testing image size. The limitations can come from position encoding, which has fixed size and need to be interpolated to different input sizes, or self-attention itself, which may not adapt well when significant changes happen in input size. Lastly, there is a long-lasting debate on the impacts of AI on human world. As a method improving the fundamental ability of deep learning, our work also advances the development of AI, which means there could be both beneficial and harmful influences depending on the users.

5.6 Conclusion

In this chapter, we present GG-Transformer, which offers an efficient and effective solution to adapting Transformers for vision tasks. GG-Transformer, inspired by how human beings learn from the world, is equipped with parallel and complementary Glance branch and Gaze branch, which offer long-range relationship and short-range modeling, respectively. The two branches can specialize in their tasks and collaborate with each other, which leads to a much more efficient ViT design for vision tasks. Experiments on various architectures and benchmarks validate the advantages of GG-Transformer.

Chapter 6

CMT-DeepLab: Clustering Mask Transformers for Panoptic Segmentation

In this chapter, we introduces our improvements over mask transformer, which is a new transformer-based framework that can handle complex panoptic segmentation problem in an end-to-end manner. Specifically, we propose a clustering view to better understand and further design the mask transformer framework, where the cross-attention and segmentation are unified as a clustering process between pixels and objects. The resulting model CMT-DeepLab shows great performance on the challenging COCO and Cityscapes datasts.

6.1 Introduction

Panoptic segmentation [114], a recently proposed challenging segmentation task, aims to unify semantic segmentation [93] and instance segmentation [86]. Due to its complicated nature, most panoptic segmentation frameworks [114, 240, 41] decompose the problem into several manageable proxy tasks, such as box detection [182], box-based segmentation [89], and semantic segmentation [153].

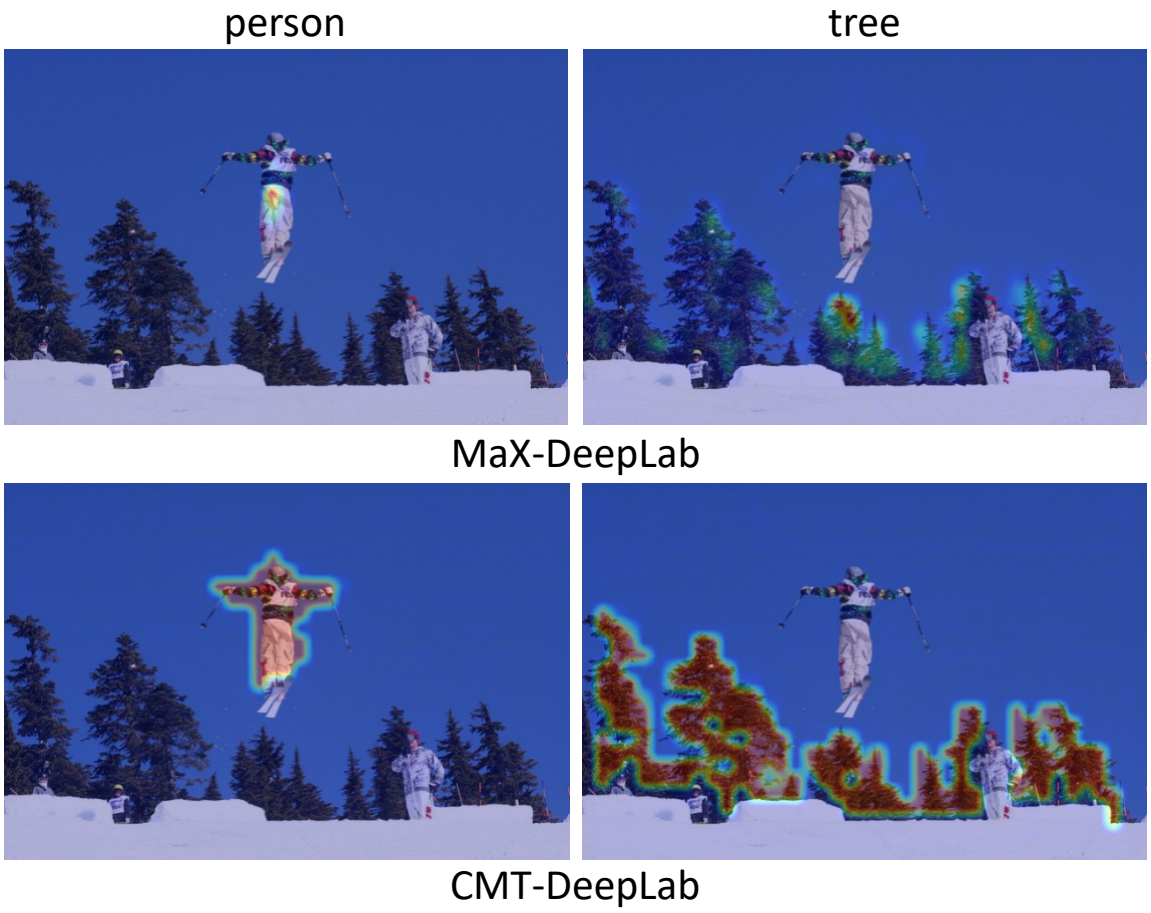

Figure 6.1: Our CMT-DeepLab generates denser cross-attention maps than MaX-DeepLab [217]. The visualization is based on the last transformer layer with averaged multi-head attentions.

Recently, the paradigm has shifted from the proxy-based approaches to end-to-end systems, since the pioneering work DETR [21], which introduces the first end-to-end object detection method with transformers [214]. In their framework, the image features, extracted by a convolutional network [120], are enhanced by transformer encoders. Afterwards, a set of fixed size of positional embeddings, named object queries, interact with the extracted image features through several transformer decoders, consisting of cross-attention and

self-attention modules [5]. The object queries, transformed into output embeddings by the decoders, are then *directly* used for bounding box predictions.

Along the same direction, end-to-end panoptic segmentation framework [217] has been proposed to simplify the panoptic segmentation procedure, avoiding manually designed modules. The core idea is to exploit a set of object queries conditioned on the inputs to predict a set of pairs, each containing a class prediction and a mask embedding vector. The mask embedding vector, multiplied by the image features, yields a binary mask prediction. Notably, unlike the box detection task, where the prediction is based on object queries themselves, segmentation mask prediction requires both object queries and pixel features to interact with each other to obtain the results, which consequently incurs different needs when updating the object queries. To have a deeper understanding towards the role that object queries play, we particularly look into the cross-attention module in the mask transformer decoder, where object queries interact with image features.

Our investigation finds that the update and usage of object queries are performed differently in the transformer-based method for segmentation tasks [217]. Specifically, when updating the object queries, a softmax operation is applied to the image dimension, allowing each query to identify its most similar pixels. On the other hand, when computing the segmentation output, a softmax is performed among the object queries so that each pixel finds its most similar object queries. The formulation may potentially cause two issues: sparse query updates and infrequent pixel-query communication. First, the object queries are only *sparsely* updated due to the softmax being applied to a large image resolution, so it tends to focus on only a few locations (top row in Fig. 6.1). Second, the pixels only have *one* chance to communicate with the object queries in the final output. The first issue is

particularly undesired, since segmentation tasks require dense predictions, and ideally a query should *densely* activate all the pixels that belong to the same target. This is different from the box detection task, where object extremities are sufficient (see Fig. 6 of DETR paper [21]).

To alleviate the issues, we draw inspiration from the traditional clustering algorithms [151, 1]. In the current end-to-end panoptic segmentation system [217], the final segmentation output is obtained by assigning each pixel to the object queries based on the feature affinity, similar to pixel-cluster assignment step in [151, 1]. The observation motivates us to rethink the transformer-based methods from the clustering perspective by considering the object queries as cluster centers. We therefore propose to additionally perform the cluster-update step, where the centers are updated by pooling pixel features based on the clustering assignment, when updating the cluster centers (*i.e.*, object queries) in the cross-attention module. As a result, our model generates denser attention maps (bottom row in Fig. 6.1). We also utilize the pixel-cluster assignment to update the pixel features within each transformer decoder, enabling frequent communication between pixel features and cluster centers.

Additionally, we notice that in the cross-attention module, pixel features are treated as in “bag of words" [119], while the location information is not well utilized. To resolve the issue, we propose to adopt a dynamic position encoding conditioned on the inputs for *location-sensitive* clustering. We explicitly predict a reference mask consisting of a few points for each cluster center. The *location-sensitive* clustering is then achieved by adding location information to pixel features and cluster centers via the coordinate convolution [143] at the beginning of each transformer decoder.

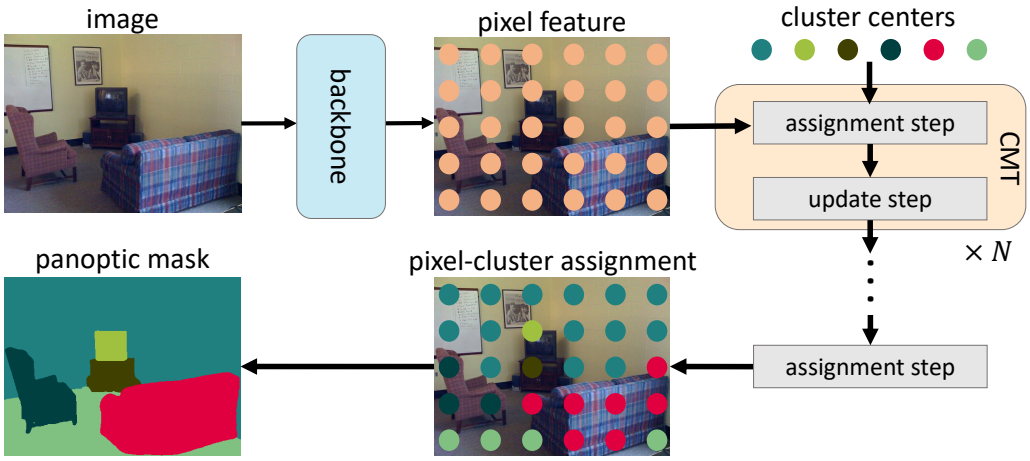

Figure 6.2: Panoptic segmentation from a clustering perspective. In the proposed Clustering Mask Transformer (CMT) layer, pixels are assigned to cluster centers based on the feature affinity, and the clustering results are used to update both pixel features and cluster centers. After several CMT layers, a refined pixel-cluster assignment is obtained, resulting in the final panoptic mask.

Combining all the proposed components results in our CMT-DeepLab, which reformulates and further improves the previous end-to-end panoptic segmentation system [217] from the traditional clustering perspective. The panoptic segmentation result is naturally obtained by assigning each pixel to its most similar cluster center based on the feature affinity (Fig. 6.2). In the Clustering Mask Transformer (CMT) module, the pixel features, cluster centers, and pixel-cluster assignments are updated in a manner similar to the clustering algorithms [151, 1]. As a result, without bells and whistles, our proposed CMT-DeepLab surpasses its baseline MaX-DeepLab [217] by 4.4% PQ and achieves 55.7% PQ on COCO panoptic *test-dev* set [138].

6.2 Related Works

Transformers. Transformer [214] variants [117, 222, 159, 46, 13, 264, 83, 2] have advanced the state-of-the-art in many natural language processing tasks [61, 191, 58] by capturing relations across modalities [5] or in a single context (self-attention) [45, 214]. In computer vision, transformers are either combined with CNNs [225, 17] or used as standalone models [180, 100, 218, 63, 149]. Both classes of methods have boosted various vision tasks, such as image classification [39, 12, 180, 100, 130, 218, 63, 149], object detection [225, 193, 180, 99, 21, 284], semantic segmentation [34, 271, 102, 70, 288, 283], video recognition [225, 39, 111], image generation [169, 95], and panoptic segmentation [218].

Proxy-based Panoptic Segmentation. Most panoptic segmentation methods rely on proxy tasks, such as object bounding box detection. For example, Panoptic FPN [114] follows a box-based approach that detects object bounding boxes and predicts a mask for each box, usually with a Mask R-CNN [89] and FPN [137]. Then, the instance segments ('thing') and semantic segments ('stuff') [29] are fused by merging modules [125, 128, 173, 140, 248, 240, 126] to generate panoptic segmentation. Other proxy-based methods typically start with semantic segments [27, 32, 35] and group 'thing' pixels into instance segments with various proxy tasks, such as instance center regression [108, 211, 165, 247, 42, 218, 133], Watershed transform [215, 6, 15], Hough-voting [10, 122, 15], or pixel affinity [109, 148, 196, 71, 15]. DetectoRS [174] achieved the state-of-the-art in this category with recursive feature pyramid and switchable atrous convolution. Recently, DETR [21] extended the proxy-based methods with its transformer-based end-to-end detector.

End-to-end Panoptic Segmentation. Along the same direction, MaX-DeepLab [217]

proposed an end-to-end strategy, in which class-labeled object masks are directly predicted and are trained by Hungarian matching the predicted masks with ground truth masks. In this work, we improve over MaX-DeepLab by approaching the pixel assignment task from a clustering perspective. Concurrent with our work, Segmenter [198] and MaskFormer [44] formulated an end-to-end strategy from a mask classification perspective, same as MaX-DeepLab [217], but extends from panoptic segmentation to semantic segmentation.

6.3 Method

Herein, we firstly introduce recent transformer-based methods [217] for end-to-end panoptic segmentation. Our observation reveals a difference between the cross-attention and final segmentation output regarding the way that they utilize object queries. We then propose to resolve it with a clustering approach, resulting in our proposed Clustering Mask Transformer (CMT-DeepLab), as shown in Fig. 6.3 and Fig. 6.4. In the following parts, object queries and cluster centers refer to the same learnable embedding vectors and we use them interchangeably for clearer representation.

6.3.1 Transformers for Panoptic Segmentation

Problem Statement. Panoptic segmentation aims to segment the input image $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$ into a set of non-overlapping masks as well as the semantic labels for the corresponding masks:

$$\{y_i\}_{i=1}^{K} = \{(m_i, c_i)\}_{i=1}^{K}. \tag{6.1}$$

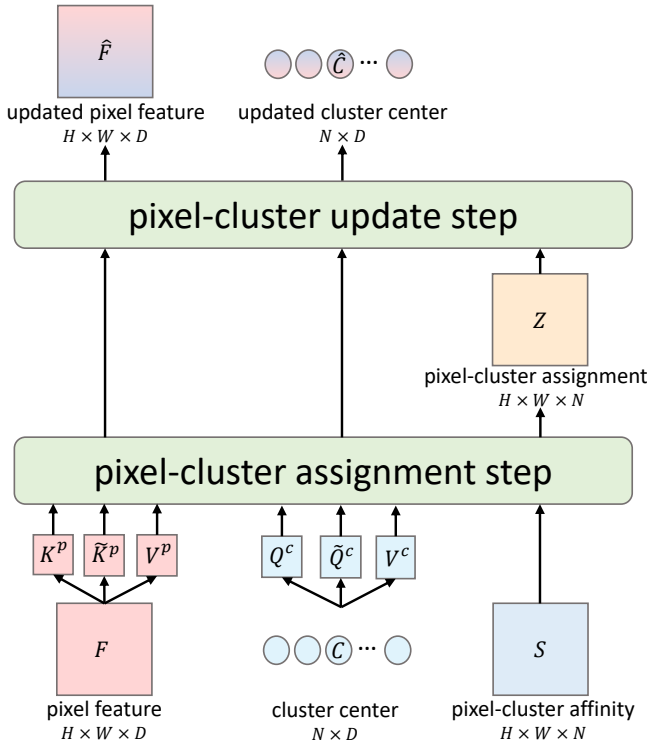

Figure 6.3: A visual illustration of Clustering Mask Transformer layer, where three variables are updated in a dynamic manner based on the clustering results: pixel features, cluster centers, and pixel-cluster affinity. Details of assignment and update steps are illustrated in Fig. 6.4.

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.

Inspired by DETR [21], several transformer-based end-to-end panoptic segmentation methods [217] have been proposed recently, which directly predict N masks and their semantic classes. N is a fixed number and $N \geq K$.

$$\{\hat{y}_i\}_{i=1}^{N} = \{(\hat{m}_i, \hat{p}_i(c))\}_{i=1}^{N}, \tag{6.2}$$

where $\hat{p}_i(c)$ denotes the predicted semantic class confidence for the corresponding mask,

including 'thing' classes, 'stuff' classes, and the void class $\varnothing$.

To predict these N masks, N object queries are utilized to aggregate information from the image features through a transformer decoder, which consists of self-attention and cross-attention modules. The object queries and image features interact with each other in the cross-attention module:

$$\hat{\mathbf{C}} = \mathbf{C} + \underset{HW}{\text{softmax}}(\mathbf{Q}^c \times (\mathbf{K}^p)^{\mathrm{T}}) \times \mathbf{V}^p, \tag{6.3}$$

where $\mathbf{C} \in \mathbb{R}^{N \times D}$ refers to object queries with D channels, and $\hat{\mathbf{C}}$ denotes the updated object queries. We use the underscript to represent the axis for softmax, and superscripts p and c to indicate the feature projected from the image features and object queries, respectively. $\mathbf{Q}^c \in \mathbb{R}^{N \times D}, \mathbf{K}^p \in \mathbb{R}^{HW \times D}, \mathbf{V}^p \in \mathbb{R}^{HW \times D}$ stand for the linearly projected features for query, key, and value. For simplicity, we ignore multi-head attention and feed-forward network (FFN) in the equation.

The object queries, updated by multiple transformer decoders, are employed as dynamic convolution weights (with kernel size 1×1) [107, 205, 227] to obtain the prediction $\mathbf{Z} \in \mathbb{R}^{HW \times N}$ that consists of N binary masks. That is,

$$\mathbf{Z} = \underset{N}{\text{softmax}}(\mathbf{F} \times \mathbf{C}^{\mathrm{T}}), \tag{6.4}$$

where $\mathbf{F} \in \mathbb{R}^{HW \times D}$ refers to the extracted image features.

6.3.2 Current Issues and New Clustering Perspective

Even though effective, the transformer-based architectures were originally designed for object detection [21] and thus they do not naturally deal with segmentation masks. Specifically, they use different formulations for the object query updates and the segmentation specific output head. To be precise, both the update of object queries (Eq. (7.4)) and final output (Eq. (7.3)) are based on their corresponding feature affinity (*i.e.*, $\mathbf{Q}^c \times (\mathbf{K}^p)^{\mathrm{T}}$ and $\mathbf{F} \times \mathbf{C}^{\mathrm{T}}$). However, the following softmax operations are applied along different dimensions. To update the object queries, the softmax is applied to the image spatial dimension (HW) with the goal to identify the most similar pixels for each query. On the other hand, to obtain the final output, the softmax is performed among the object queries (N) so that each pixel finds its most similar object queries. The inconsistency potentially causes two issues. First, the object queries are only *sparsely* updated due to the softmax operated along a large spatial dimension, tending to focus on only a few locations (Fig. 6.1). Second, the output update is only performed *once* in the end, and therefore the pixels only have one chance to receive the information passed from the object queries.

To alleviate the issues, we take a closer look at Eq. (7.3), which assigns each pixel to the object queries based on the feature affinity. This is, in fact, very similar to typical clustering methods [151, 1] (particularly, the pixel-cluster assignment step). This observation motivates us to rethink the transformer-based methods from the typical clustering perspective [282, 1] by considering the object queries $\mathbf{C}$ as cluster centers. With the clustering perspective in mind, we re-interpret Eq. (7.3) as the pixel-cluster assignment. This interpretation naturally inspires us to perform a cluster-update step where the cluster centers are updated by pooling pixel features based on the clustering assignment, *i.e.*,

$\mathbf{Z}^{\mathrm{T}} \times \mathbf{F} = (\mathrm{softmax}_N(\mathbf{F} \times \mathbf{C}^{\mathrm{T}}))^{\mathrm{T}} \times \mathbf{F}$.

We propose to extend the formulation to a transformer decoder module, whose query, key, and value are obtained by linearly projecting the image features and cluster centers:

$$\hat{\mathbf{C}} = \mathbf{C} + (\underset{N}{\mathrm{softmax}}(\tilde{\mathbf{K}}^p \times (\tilde{\mathbf{Q}}^c)^{\mathrm{T}}))^{\mathrm{T}} \times \mathbf{V}^p. \tag{6.5}$$

Comparing Eq. (7.4) and Eq. (6.5), we have the query $\tilde{\mathbf{Q}}^c$ and key $\tilde{\mathbf{K}}^p$ coming from another linear projection, and the softmax is performed along the cluster center dimension.

In the following subsection, we detail how the clustering perspective alleviates the issues of current transformer-based methods. In the discussion, we use object queries and cluster centers interchangeably.

6.3.3 Clustering Mask Transformers

In this subsection, we redesign the cross-attention in the transformer decoder from the clustering perspective, aiming to resolve the issues raised in Sec. 6.3.2.

Residual Path between Cluster Assignments. Similar to other designs [21], we stack the transformer decoder multiple times. To facilitate the learning of pixel-cluster assignment, we add a residual connection [92] between clustering results including the final segmentation result. That is,

$$\mathbf{Z} = \underset{N}{\mathrm{softmax}}(\mathbf{S} + \tilde{\mathbf{K}}^p \times (\tilde{\mathbf{Q}}^c)^{\mathrm{T}}), \tag{6.6}$$

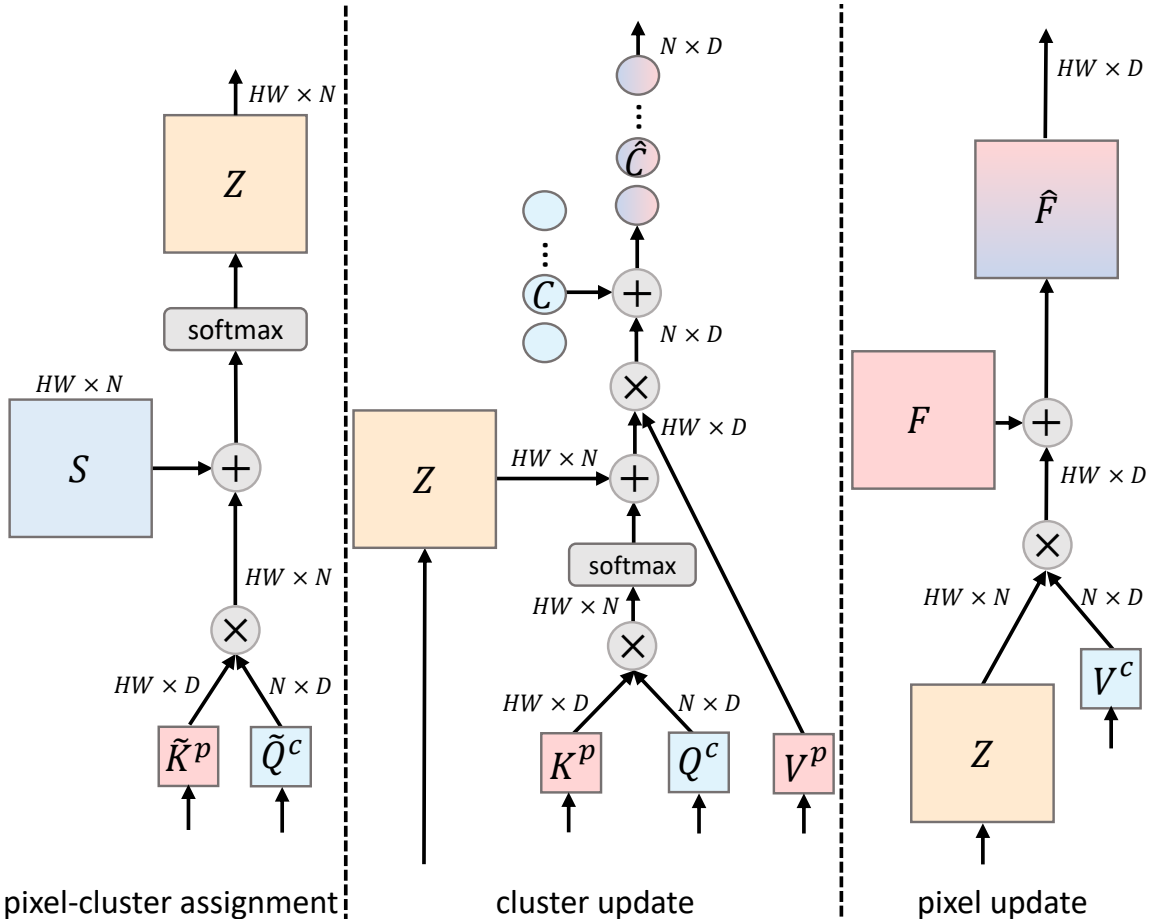

Figure 6.4: Detailed visual illustration of pixel-cluster assignment (left), cluster centers update (middle), and pixel features update (right). The tensor shapes are specified for illustration.

where $\mathbf{S} \in \mathbb{R}^{HW \times N}$ is the affinity logits between linearly projected pixel features and cluster centers in the previous decoder (left panel of Fig. 6.4). We emphasize that since our clustering results have the same format as the segmentation output, we are able to add residual connections between them, which is further supervised by the ground-truths.

Solution to Sparse Query Update. We propose a simple and effective solution to avoid the sparse query update by combining the proposed clustering center update (*i.e.*, Eq. (6.5))

with the original cross-attention (*i.e.*, Eq. (7.4)), resulting in

$$\begin{aligned}\hat{\mathbf{C}} =& \mathbf{C} + \underset{HW}{\text{softmax}}(\mathbf{Q}^c \times (\mathbf{K}^p)^{\mathrm{T}}) \times \mathbf{V}^p + \mathbf{Z}^{\mathrm{T}} \times \mathbf{V}^p \\ =& \mathbf{C} + (\underset{HW}{\text{softmax}}(\mathbf{Q}^c \times (\mathbf{K}^p)^{\mathrm{T}}) + \mathbf{Z}^{\mathrm{T}}) \times \mathbf{V}^p,\end{aligned} \tag{6.7}$$

where $\mathbf{Z}$ is obtained from Eq. (6.6). The update is shown in the center panel of Fig. 6.4, while the effect of *densified* attention could be found in Fig. 6.1.

Solution to Infrequent Pixel Updates. We propose to also utilize the clustering result $\mathbf{Z}$ to perform an update on the pixel features using the features of cluster centers, *i.e.*,

$$\hat{\mathbf{F}} = \mathbf{F} + \mathbf{Z} \times \mathbf{V}^c, \tag{6.8}$$

where $\mathbf{V}^c \in \mathbb{R}^{N \times D}$ is the linearly projected values from the cluster centers. This update is performed within each stacked transformer decoder, enabling frequent communication between pixel features and cluster centers (right panel of Fig. 6.4).

To this end, we have improved the transformer cross-attention module by simultaneously updating the clustering result (*i.e.*, pixel-cluster assignment), pixel features, and cluster centers. However, we notice that during the interaction between pixel features and cluster centers, pixel features are treated as bag of words [119], while the location information is not well utilized. Although learnable positional encodings (*i.e.* object queries [21]) are used for the cluster center embeddings, the positional encodings are fixed for all input images, which is suboptimal when an object query predicts masks at different locations in different input images. To resolve the issue, we propose to adopt a dynamic positional encoding conditioned on the inputs for *location-sensitive* clustering.

Location-Sensitive Clustering. To inject dynamic location information to cluster centers,

we explicitly predict a reference mask that consists of M points for each cluster center. In particular, a MLP is used to predict the reference mask out of cluster center features, followed by a sigmoid activation function. That is, we have:

$$\hat{e} = e + \text{MLP}(\mathbf{C}), \tag{6.9}$$

$$r^c = \text{sigmoid}(\hat{e}), \tag{6.10}$$

where $e \in \mathbb{R}^{N \times 2M}$ denotes an embedding projected from the cluster centers, and $r^c = [r^{c,h}, r^{c,w}] \in \mathbb{R}^{N \times 2M}$ are the reference mask represented with M pairs of coordinates $(r_i^{c,h}, r_i^{c,w})$. We utilize a residual update manner [92, 284] to predict the reference mask, with a skip-connection on the projected embedding e across stages. The location space is normalized to $[0, 1] \times [0, 1]$.

We add location information to pixel features and cluster centers through a coordinate convolution [143]. Specifically, we apply coordinate convolutions at the beginning of each transformer layer to ensure location information is considered during the clustering process, as shown below.

$$\hat{\mathbf{C}} = \text{Conv}(\text{Concat}(\mathbf{C}, r^c)), \tag{6.11}$$

$$\hat{\mathbf{F}} = \text{Conv}(\text{Concat}(\mathbf{F}, r^p)), \tag{6.12}$$

where $r^p \in \mathbb{R}^{HW \times 2}$ is the coordinates normalized to $[0, 1]$ for pixels in image space, which is fixed and not learnable.

We note that compared to the reference point used in the Deformable DETR [284], the proposed reference mask provides a rough mask shape prior for the whole object

mask. Besides, we adopt a much simpler way to incorporate the location information via coordinate convolution.

In order to learn meaningful reference mask predictions, we optimize the reference masks towards ground truth masks by proposing a mask approximation loss.

Mask Approximation Loss. We propose a loss to minimize the distance between the distribution of predicted reference points and that of points of ground-truth object masks. In detail, we utilize the Hungarian matching result to assign the ground-truth mask for each cluster center. Given the predicted M points for each cluster center, we infer their extreme points [168] and mask center. We then apply an L_1 loss to push them to be closer to their ground-truth extreme points and center. Specifically, we have

$$\begin{aligned}
\mathcal{L}_{\text{ext}} &= \frac{1}{4K}\sum_{i=1}^{K}(|\min(r_i^{c,h}) - \min(y_i^h)| + |\max(r_i^{c,h}) - \max(y_i^h)| \\
&\qquad + |\min(r_i^{c,w}) - \min(y_i^w)| + |\max(r_i^{c,w}) - \max(y_i^w)|), \\
\mathcal{L}_{\text{cen}} &= \frac{1}{2K}\sum_{i=1}^{K}(|\operatorname{avg}(r_i^{c,h}) - \operatorname{avg}(y_i^h)| + |\operatorname{avg}(r_i^{c,w}) - \operatorname{avg}(y_i^w)|), \\
\mathcal{L}_{\text{loc}} &= \mathcal{L}_{\text{ext}} + \mathcal{L}_{\text{cen}},
\end{aligned} \tag{6.13}$$

where $y = [y^h, y^w]$ are pixels on ground-truth masks and predicted reference masks have been filtered and re-ordered based on Hungarian matching results.

Finally, combining all the proposed designs results in our Clustering Mask Transformer, or CMT-DeepLab, which rethinks the current mask transformer design from the clustering perspective.

6.3.4 Network Instantiation

We instantiate CMT-DeepLab on top of MaX-DeepLab-S [217] (abbreviated as MaX-S). We first refine its architecture design. Afterwards, we enhance it with the proposed Clustering Mask Transformers.

Base Architecture. We use MaX-S [217] as our base architecture. To better align it with other state-of-the-art architecture designs [149], we use GeLU [94] activation to replace the original ReLU activation functions. Besides, we remove all transformer blocks in the pretrained backbones, which reverts the backbone from MaX-S back to Axial-ResNet-50 [218]. On top of the backbone, we append six dual-path axial-transformer blocks [217] (three at stage-5 w/ channels 2048, and the other three at stage-4 w/ channels 1024), yielding totally six axial self-attention and six cross-attention modules, which aligns with the number of attention operations used in other works [21, 44]. Additionally, we obtain a larger network backbone by scaling up the number of blocks in stage-4 of the backbone [33]. As a result, two different model variants are used: one built upon Axial-ResNet-50 backbone with number of blocks $[3, 4, 6, 3]$ (starting from stage-2), and another built upon Axial-ResNet-104 with number of blocks $[3, 4, 24, 3]$. See the supplementary material for a detailed illustration.

Loss Functions. Following [217], we use the PQ-style loss and three other auxiliary losses for the model training, including the instance discrimination loss, mask-ID cross-entropy, and semantic segmentation loss. However, we note that the instance discrimination loss proposed in [217] aims to push pixel features to be close to the feature center computed based on the ground-truth mask, instead of directly to the cluster centers. Therefore, we adopt the

pixel-wise instance discrimination loss, which learns closely aligned representations for all pixels from the same class, allowing better clustering results.

Formally, we sample a set of pixels A from the image, where we add bias to pixels' sampling probability based on the size of object mask they belong to. Thus, final sampled pixels are more balanced from objects with different scales. Afterwards, we directly perform contrastive loss on top of these pixels with multiple positive targets [110]:

$$\mathcal{L}^{insdis} = \sum_{a \in A} \frac{-1}{|P(a)|} \sum_{p \in P(a)} \log \frac{\exp\left(f_a \cdot f_p / \tau\right)}{\sum_{b \in A} \exp\left(f_a \cdot f_b / \tau\right)}, \tag{6.14}$$

where $P(a)$ is a subset of pixels of A that belongs to the same cluster (*i.e.*, object mask) with a, and $|P(a)|$ is its cardinally. We use f to denote a pixel feature vector, and τ is the temperature.

Recursive Feature Network. Motivated by DetectoRS [174] and CBNet [147], we adopt a simple strategy, named Recursive Feature Network (RFN), to increase the network capacity by stacking twice the whole model (including the backbone and added transformer blocks). There are two main differences. First, since we do not employ an FPN [137] (as in [174]), we simply connect the features at stride 4 (*i.e.*, same stride as the segmentation output). Second, we do not use the complicated fusion module proposed in [174], but simply average the features between two stacked networks, which we empirically found to be better by around 0.2% PQ.

method	backbone	TTA	params	val-set PQ	val-set PQ^{Th}	val-set PQ^{St}	test-dev PQ	test-dev PQ^{Th}	test-dev PQ^{St}
box-based panoptic segmentation methods									
Panoptic-FPN [114]	R101			40.3	47.5	29.5	-	-	-
UPSNet [240]	R50			42.5	48.5	33.4	-	-	-
UPSNet [240]	R50	✓		43.2	49.1	34.1	-	-	-
UPSNet [240]	DCN-101 [57]	✓		-	-	-	46.6	53.2	36.7
DETR [21]	R101		61.8M	45.1	50.5	37.0	46.0	-	-
DetectoRS [174]	RX-101 [237]	✓		-	-	-	49.6	57.8	37.1
center-based panoptic segmentation methods									
Panoptic-DeepLab [42]	X-71 [48]		46.7M	39.7	43.9	33.2	-	-	-
Panoptic-DeepLab [42]	X-71 [48]	✓	46.7M	41.2	44.9	35.7	41.4	45.1	35.9
Axial-DeepLab-L [218]	AX-L [218]		44.9M	43.4	48.5	35.6	43.6	48.9	35.6
Axial-DeepLab-L [218]	AX-L [218]	✓	44.9M	43.9	48.6	36.8	44.2	49.2	36.8
end-to-end panoptic segmentation methods									
MaX-DeepLab-S [217]	MaX-S [217]		61.9M	48.4	53.0	41.5	49.0	54.0	41.6
MaX-DeepLab-L [217]	MaX-L [217]		451M	51.1	57.0	42.2	51.3	57.2	42.4
MaskFormer [44]	Swin-B‡ [149]		102M	51.8	56.9	44.1	-	-	-
MaskFormer [44]	Swin-L‡ [149]		212M	52.7	58.5	44.0	53.3	59.1	44.5
CMT-DeepLab	Axial-R50‡ [218]		94.9M	53.0	57.7	45.9	53.4	58.3	46.0
CMT-DeepLab	Axial-R104‡		135.2M	54.1	58.8	**47.1**	54.5	59.6	46.9
CMT-DeepLab	Axial-R104‡-RFN		270.3M	55.1	60.6	46.8	55.4	61.0	**47.0**
CMT-DeepLab (iter 200k)	Axial-R104‡-RFN		270.3M	**55.3**	**61.0**	46.6	**55.7**	**61.6**	46.8

Table 6.1: Results comparison on COCO val and test-dev set. **TTA:** Test-time augmentation. ‡: ImageNet-22K pretraining. We provide more comparisons with concurrent works in the supplementary materials.

6.4 Experimental Results

We report main results on COCO along with state-of-the-art methods, followed by ablation studies on the architecture variants, clustering mask transformers, pretrained weights, post-processing, and scaling strategies. Finally, we analyze the working mechanism behind CMT-DeepLab with visualizations.

Implementation Details. We build CMT-DeepLab on top of MaX-DeepLab [217] with the official code-base [228]. The training strategy mainly follows MaX-DeepLab. If not specified, the model is trained with 64 TPU cores for 100k iterations with the first 5k for

((a)) Clustering update.

	PQ	PQ^{Th}	PQ^{St}
baseline	46.2	50.0	40.5
+ clustering transformer	47.1	51.0	41.1
+ pixel-wise contrastive loss	**47.5**	**51.1**	**42.1**

((b)) Location-senseitive clustering.

	PQ	PQ^{Th}	PQ^{St}
baseline	46.2	50.0	40.5
+ ref. mask pred.	46.6	50.3	40.9
+ coord-conv	**46.9**	**50.6**	**41.3**

((c)) Architecture.

clustering	location	decoder	params	PQ	PQ^{Th}	PQ^{St}
			61.9M	46.2	50.0	40.5
✓			61.9M	47.5	51.1	42.1
	✓		65.5M	46.9	50.6	41.3
		✓	91.0M	47.1	51.3	40.9
✓		✓	91.0M	48.1	51.9	42.2
✓	✓	✓	94.9M	**48.4**	**52.1**	**42.8**

((d)) Pretraining, post-processing, and scaling.

IN-22K	RFN	mask-wise merge	PQ	PQ^{Th}	PQ^{St}
			48.4	52.1	42.8
✓			49.3	53.3	43.4
✓	✓		50.1	**54.8**	43.0
✓	✓	✓	**50.6**	**54.8**	**44.3**

Table 6.2: CMT-DeepLab ablation experiments. Results are reported in an accumulative manner.

warm-up. We use batch size = 64, Adam [113] optimizer, a poly schedule learning rate of 10^{-3}. The ImageNet-pretrained [185] backbone has a learning rate multiplier 0.1. Weight decay is set to 0 and drop-path rate [101] to 0.2. The input images are resized and padded to 1281×1281 for training and inference. We use $|A| = 4096$ for pixel-wise contrastive loss and $M = 8$ for reference masks, we also tried other values but did not observe significant difference. Loss weight is 1.0 for the mask approximation loss. Other losses employ the same setting as [217]. During inference, we adopt a mask-wise merging scheme [44] to obtain the final results.

6.4.1 Main Results

Our main results on the COCO panoptic segmentation *val* set and *test-dev set* are summarized in Tab. 7.1.

Val Set. We compare our validation set results with box-based, center-based, and end-to-end

res.	backbone	iters	PQ	PQ^{Th}	PQ^{St}
641	Axial-R50	100k	50.1	53.5	44.9
641	Axial-R50	200k	50.6	54.5	44.8
1281	Axial-R50	100k	53.0	57.7	45.9
1281	Axial-R50	200k	53.5	58.5	45.9
641	Axial-R104	100k	51.7	55.4	46.4
641	Axial-R104	200k	52.2	56.4	46.0
1281	Axial-R104	100k	54.1	58.8	47.1
1281	Axial-R104-RFN	100k	55.1	60.6	46.8

Table 6.3: Ablation on **input resolution, backbone, and training iterations**. ImageNet-22K, mask-wise merge are used for all results.

panoptic segmentation methods. It is noticeable that CMT-DeepLab, built upon a smaller backbone Axial-ResNet-50, already surpasses all other box-based and center-based methods by a large margin. More importantly, when compared with its end-to-end baseline MaX-DeepLab-S [217], we observe a significant improvement of 4.6% PQ. Our small model even surpasses previous state-of-the-art method MaX-DeepLab-L [217], which has more than $5\times$ parameters, by 1.9% PQ. Compared to recently proposed MaskFormer [44], CMT-DeepLab still shows a significant advantage of 1.2% PQ and 1.4% PQ while being more light-weight over the small and large model variant, respectively. The significant improvement illustrates the importance of introducing the concept of clustering into transformer, which leads to a denser attention preferred by the segmentation task. Our CMT-DeepLab with a deeper backbone Axial-ResNet-104 improves the *single-scale* performance to 54.1% PQ, outperforming *multi-scale* Axial-DeepLab [218] by 10.2% PQ. Moreover, we enhance the model with the proposed RFN, which further improves the PQ to 55.3%.

Test-dev Set. We verify the transfer-ability of CMT-DeepLab on *test-dev* set, which shows consistently better results compared to other methods. Especially, the small version of

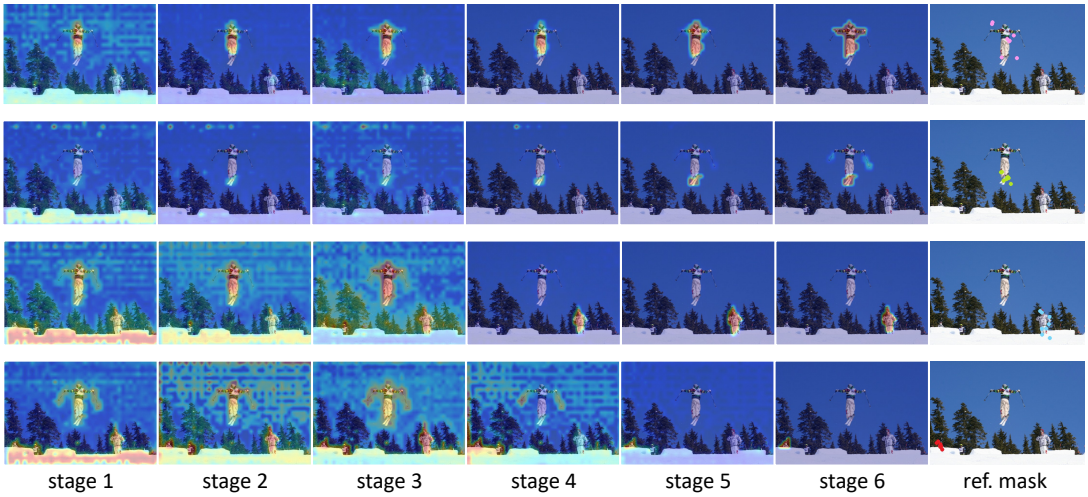

Figure 6.5: Visualization of clustering results at different stages (*i.e.*, transformer layers), with last column for reference masks. The clustering results, providing denser attention maps, are close-to-random at the beginning and are gradually refined to focus on corresponding object.

CMT-DeepLab with Axial-R50 backbone outperforms DETR [21] by 7.4% PQ, MaX-DeepLab-S [217] by 4.4% PQ, and MaX-DeepLab-L [217] by 2.1% PQ. Additionally, employing a deeper backbone Axial-R104 can boost the PQ score by 1.1% PQ. On top of it, using the proposed RFN further improves PQ to 55.7%, surpassing MaskFormer [44] with Swin-L [149] backbone by 2.4% PQ.

6.4.2 Ablation Studies

Herein, we evaluate the effectiveness of different components of the proposed CMT-DeepLab. For all the following experiments, we use MaX-DeepLab-S [217] with GeLU [94] activation function as our baseline. This improved baseline has a 0.3% higher PQ compared to the original MaX-DeepLab-S. If not specified, we perform all ablation studies with the Axial-R50 backbone [92, 218], ImageNet-1K [185] pretrained, crop size 641 $\times$ 641, and 100k training iterations.

Clustering Mask Transformer. We start with adding the design variants of Clustering Mask Transformer step by step, as summarized in Tab. 6.1(a). Regarding the object queries as cluster centers, and adding a clustering-style update can improve the PQ by 0.9%, illustrating the effectiveness of the cluster center perspective and the importance of including more pixels into the cluster center updates. Next, we utilize pixel-wise contrastive loss instead of the original instance-wise contrastive loss, resulting in another 0.4% PQ improvement, as it provides a better supervision signal from a clustering perspective. In short, re-designing the transformer layer from a clustering perspective leads to a 1.3% PQ improvement overall.

Location-Sensitive Clustering. Location information plays an important role in the clustering process, as shown in Tab. 6.1(b). Each cluster center needs to predict a reference mask without using pixel features (*i.e.*, appearance information), which requires cluster centers to include more location information in the feature embedding and thus benefits clustering. Adding reference masks prediction alone brings a gain of 0.4% PQ. Using the coordinate convolution (coord-conv) [143] to include the reference mask information yields another 0.3% PQ improvement. In sum, the location-sensitive clustering brings up the PQ score by 0.7%.

Stronger Decoder. We study the effect of using a stronger decoder design [21, 44]. We remove all transformer layers from the pretrained backbone, which reverts the MaX-S backbone [217] to Axial-ResNet-50 [218]. Then we stack more axial-blocks with transformer module in the decoder part. More specifically, we use six self-attention modules and six cross-attention modules in total for the decoder, which aligns to the design of DETR [21]. As shown in Tab. 6.1(c), this stronger decoder brings 0.9% PQ improvement

(47.1% *vs*. 46.2%).

As shown in Tab. 6.1(c), these improvements are complementary to each other, while combining them together can further boost the performance. Adding all of them leads to CMT-DeepLab, which improves 2.2% PQ over the MaX-DeepLab-S-GeLU baseline. We note that the major cost comes from the stronger decoder, which accounts for the increase of 29.1M parameters, while clustering update and location-sensitive clustering improve the PQ by 1.3% and 0.7%, respectively, with neglectable extra parameters.

Pretraining, Post-processing, and Scaling. We further verify the effect of better pretraining, post-processing, and scaling-up, with results summarized in Tab. 6.1(d) and Tab. 6.3. Specifically, we find that using ImageNet-22K for pretraining can improve the performance by 0.9% PQ. Furthermore, we empirically find that using the mask-wise merge strategy [44] to obtain panoptic results, compared to the simple per-pixel strategy [217], improves PQ by 0.5%. Next, we scale up CMT-DeepLab from different dimensions. With a longer training strategies (from 100k to 200k iterations), we observe a consistent 0.5% PQ improvement over various settings, where the improvement mainly comes from PQ^{Th} (*i.e.*, thing classes), indicating that the model needs a longer training schedule to better segment thing objects. We also find that using a larger input resolution (from 641 to 1281) significantly boosts the performance by more than 2% PQ. Besides, increasing the model size by using a deeper backbone or stacking the model with RFN can improve the performance by 1.6% and 1.0%, respectively.

Visualization. In Fig. 6.5, we visualize the clustering results in each stage as well as the learned reference masks. As shown in the figure, the clustering results, starting with a close-to-random assignment, gradually learn to focus on the target instances. For example, in the

last two rows of Fig. 6.5, the clustering results firstly focus on all the ‘person’ instances and the background ‘snow’, and then they start to concentrate on the specific person instance, showing a refinement from “semantic segmentation" to “instance segmentation". Moreover, as shown in the last column of Fig. 6.5, the learned reference mask provides a reasonable prior for the object mask.

6.5 Conclusion

In this chapter, we have introduced CMT-DeepLab, which rethinks object queries, used in the current mask transformers for panoptic segmentation, from a clustering perspective. Considering object queries as cluster centers, our framework additionally incorporates the proposed cluster center update in the cross-attention module, which significantly enriches the learned cross-attention maps and further facilitates the segmentation prediction. As a result, CMT-DeepLab achieves new state-of-the-art performance on the COCO dataset, and sheds light on the working mechanism behind mask transformers for segmentation tasks.

Chapter 7

k-means Mask Transformer

In this chapter, we take one more step in rethinking the mask transformer from a clustering process. More specifically, we consider k-means clustering, based on which we re-design the transformer module with a simple change on the activation function, which leads to k-means transformer decoder. Although the change is very simple, it proves to be very effective. The proposed kMaX-DeepLab not only significantly boosts the performance, but also produces more plausible attention map to better understand the working mechanism behind the model.

7.1 Introduction

Transformers [214] are receiving a growing attention in the computer vision community. On the one hand, the transformer encoder, with multi-head self-attention as the central component, demonstrates a great potential for building powerful network architectures in various visual recognition tasks [218, 63, 149]. On the other hand, the transformer decoder, with multi-head cross-attention at its core, provides a brand-new approach to

tackling complex visual recognition problems in an end-to-end manner, dispensing with hand-designed heuristics.

Recently, the pioneering work DETR [21] introduces the first end-to-end object detection system with transformers. In this framework, the pixel features are firstly extracted by a convolutional neural network [120], followed by the deployment of several transformer encoders for feature enhancement to capture long-range interactions between pixels. Afterwards, a set of learnable positional embeddings, named object queries, is responsible for interacting with pixel features and aggregating information through several interleaved cross-attention and self-attention modules. In the end, the object queries, decoded by a Feed-Forward Network (FFN), directly correspond to the final bounding box predictions. Along the same direction, MaX-DeepLab [217] proves the success of transformers in the challenging panoptic segmentation task [115], where the prior arts [114, 240, 41] usually adopt complicated pipelines involving hand-designed heuristics. The essence of this framework lies in converting the object queries to mask embedding vectors [107, 205, 227], which are employed to yield a set of mask predictions by multiplying with the pixel features.

The end-to-end transformer-based frameworks have been successfully applied to multiple computer vision tasks with the help of transformer decoders, especially the cross-attention modules. However, the working mechanism behind the scenes remains unclear. The cross-attention, which arises from the Natural Language Processing (NLP) community, is originally designed for language problems, such as neural machine translation [200, 5], where both the input sequence and output sequence share a similar short length. This implicit assumption becomes problematic when it comes to certain vision problems, where the cross-attention is performed between object queries and spatially flattened pixel features

with an exorbitantly large length. Concretely, usually a small number of object queries is employed (*e.g.*, 128 queries), while the input images can contain thousands of pixels for the vision tasks of detection and segmentation. Each object query needs to learn to highlight the most distinguishable features among the abundant pixels in the cross-attention learning process, which subsequently leads to slow training convergence and thus inferior performance [284, 72].

In this work, we make a crucial observation that the cross-attention scheme actually bears a strong similarity to the traditional k-means clustering [151] by regarding the object queries as cluster centers with learnable embedding vectors. Our examination of the similarity inspires us to propose the novel **k**-means **Ma**sk **X**former (kMaX-DeepLab), which rethinks the relationship between pixel features and object queries, and redesigns the cross-attention from the perspective of k-means clustering. Specifically, when updating the cluster centers (*i.e.*, object queries), our kMaX-DeepLab performs a different operation. Instead of performing *softmax* on the large spatial dimension (image height times width) as in the original Mask Transformer's cross-attention [217], our kMaX-DeepLab performs *argmax* along the cluster center dimension, similar to the k-means pixel-cluster assignment step (with a *hard* assignment). We then update cluster centers by aggregating the pixel features based on the pixel-cluster assignment (computed by their feature affinity), similar to the k-means center-update step. In spite of being conceptually simple, the modification has a striking impact: on COCO *val* set [138], using the standard ResNet-50 [92] as backbone, our kMaX-DeepLab demonstrates a significant improvement of **5.2%** PQ over the original cross-attention scheme at a negligible cost of extra parameters and FLOPs.

When comparing to state-of-the-art methods, our *k*MaX-DeepLab with the simple ResNet-50 backbone already outperforms MaX-DeepLab [217] with MaX-L [217] backbone by **1.9%** PQ, while requiring **7.9** and **22.0** times fewer parameters and FLOPs, respectively. Our *k*MaX-DeepLab with ResNet-50 also outperforms MaskFormer [44] with the strong ImageNet-22K pretrained Swin-L [149] backbone, and runs **4.4** times faster. Using the modern ConvNeXt-L [150] as backbone, our *k*MaX-DeepLab further sets a new state-of-the-art performance on the COCO *val* set [138] with 58.0% PQ. It also outperforms other state-of-the-art methods on the Cityscapes *val* set [54], achieving 68.4% PQ, 83.5% mIoU, 44.0% AP, without using any test-time augmentation or extra dataset pretraining [138, 164]. Finally, *k*MaX-DeepLab also advances the new state-of-the-art performance on ADE20K [277] with 50.9% PQ and 55.2% mIoU.

7.2 Related Works

Transformers. Transformer [214] and its variants [117, 222, 159, 46, 13, 264, 83, 2] have advanced the state-of-the-art in natural language processing tasks [61, 191, 58] by capturing relations across modalities [5] or in a single context [45, 214]. In computer vision, transformer encoders or self-attention modules are either combined with Convolutional Neural Networks (CNNs) [225, 17] or used as standalone backbones [180, 100, 218, 63, 149]. Both approaches have boosted various vision tasks, such as image classification [39, 12, 180, 100, 130, 218, 63, 149, 256, 245], image generation [169, 95], object detection [225, 193, 180, 99, 21, 284], video recognition [225, 39, 4, 67], semantic segmentation [34, 271, 102, 70, 288, 283, 273, 235, 24], and panoptic segmentation [218].

Mask transformers for segmentation. Besides the usage as backbones, transformers

are also adopted as task decoders for image segmentation. MaX-DeepLab [217] proposed **Ma**sk **X**formers (MaX) for end-to-end panoptic segmentation. Mask transformers predict class-labeled object masks and are trained by Hungarian matching the predicted masks with ground truth masks. The essential component of mask transformers is the conversion of object queries to mask embedding vectors [107, 205, 227], which are employed to generate predicted masks. Both Segmenter [198] and MaskFormer [44] applied mask transformers to semantic segmentation. K-Net [267] proposed dynamic kernels for generating the masks. CMT-DeepLab [253] proposed to improve the cross-attention with an additional clustering update term. Panoptic Segformer [135] strengthened mask transformer with deformable attention [284], while Mask2Former [43] further boosted the performance with masked cross-attention along with a series of technical improvements including cascaded transformer decoder, deformable attention [284], uncertainty-based pointly supervision [116], *etc*. These mask transformer methods generally outperform box-based methods [114] that decompose panoptic segmentation into multiple surrogate tasks (*e.g*., predicting masks for each detected object bounding box [89], followed by fusing the instance segments ('thing') and semantic segments ('stuff') [29] with merging modules [128, 173, 140, 248, 240, 126]). Moreover, mask transformers showed great success in the video segmentation problems [111, 40, 127].

Clustering methods for segmentation. Traditional image segmentation methods [151, 282, 1] typically cluster image intensities into a set of masks or superpixels with gradual growing or refinement. However, it is challenging for these traditional methods to capture high-level semantics. Modern clustering-based methods usually operate on semantic

segments [27, 32, 35] and group 'thing' pixels into instance segments with various representations, such as instance center regression [108, 211, 165, 247, 42, 218, 133], Watershed transform [215, 6], Hough-voting [10, 122, 216], or pixel affinity [109, 148, 196, 71, 103].

Recently, CMT-DeepLab [253] discussed the similarity between mask transformers and clustering algorithms. However, they only used the clustering update as a complementary term in the cross-attention. In this work, we further discover the underlying similarity between mask transformers and the k-means clustering algorithm, resulting in a simple yet effective k-means mask transformer. Finally, we note that several recent works [152, 241, 253, 281] revisited the relationship between query and key in the attention operation. They applied the cross-attention softmax operation along the query dimension and showed promising results.

7.3 Method

In this section, we first overview the mask-transformer-based segmentation framework presented by MaX-DeepLab [217]. We then revisit the transformer cross-attention [214] and the k-means clustering algorithm [151], and reveal their underlying similarity. Afterwards, we introduce the proposed **k**-means **Ma**sk **X**former (kMaX-DeepLab), which redesigns the cross-attention from a clustering perspective. Even though simple, kMaX-DeepLab effectively and significantly improves the segmentation performance.

7.3.1 Mask-Transformer-Based Segmentation Framework

Transformers [214] have been effectively deployed to segmentation tasks. Without loss of generality, we consider panoptic segmentation [115] in the following problem formulation,

which can be easily generalized to other segmentation tasks.

Problem statement. Panoptic segmentation aims to segment the image $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$ into a set of non-overlapping masks with associated semantic labels:

$$\{y_i\}_{i=1}^{K} = \{(m_i, c_i)\}_{i=1}^{K}. \tag{7.1}$$

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.

Starting from DETR [21] and MaX-DeepLab [217], approaches to panoptic segmentation shift to a new end-to-end paradigm, where the prediction directly matches the format of ground-truth with N masks (N is a fixed number and $N \geq K$) and their semantic classes:

$$\{\hat{y}_i\}_{i=1}^{N} = \{(\hat{m}_i, \hat{p}_i(c))\}_{i=1}^{N}, \tag{7.2}$$

where $\hat{p}_i(c)$ denotes the semantic class prediction confidence for the corresponding mask, which includes 'thing' classes, 'stuff' classes, and the void class $\varnothing$.

The N masks are predicted based on the N object queries, which aggregate information from the pixel features through a transformer decoder, consisting of self-attention and cross-attention modules.

The object queries, updated by multiple transformer decoders, are employed as mask embedding vectors [107, 205, 227], which will multiply with the pixel features to yield the final prediction $\mathbf{Z} \in \mathbb{R}^{HW \times N}$ that consists of N masks. That is,

$$\mathbf{Z} = \underset{N}{\text{softmax}}(\mathbf{F} \times \mathbf{C}^{\mathrm{T}}), \tag{7.3}$$

where $\mathbf{F} \in \mathbb{R}^{HW \times D}$ and $\mathbf{C} \in \mathbb{R}^{N \times D}$ refers to the final pixel features and object queries,

respectively. D is the channel dimension of pixel features and object queries. We use underscript N to indicate the axis to perform softmax.

7.3.2 Relationship between Cross-Attention and k-means Clustering

Although the transformer-based segmentation frameworks successfully connect object queries and mask predictions in an end-to-end manner, the essential problem becomes how to transform the object queries, starting from learnable embeddings (randomly initialized), into meaningful mask embedding vectors.

Cross-attention. The cross-attention modules are used to aggregate affiliated pixel features to update object queries. Formally, we have

$$\hat{\mathbf{C}} = \mathbf{C} + \underset{HW}{\mathrm{softmax}}(\mathbf{Q}^c \times (\mathbf{K}^p)^{\mathrm{T}}) \times \mathbf{V}^p, \tag{7.4}$$

where $\mathbf{C} \in \mathbb{R}^{N \times D}$ refers to N object queries with D channels, and $\hat{\mathbf{C}}$ denotes the updated object queries. We use the underscript HW to represent the axis for softmax on spatial dimension, and superscripts p and c to indicate the feature projected from the pixel features and object queries, respectively. $\mathbf{Q}^c \in \mathbb{R}^{N \times D}, \mathbf{K}^p \in \mathbb{R}^{HW \times D}, \mathbf{V}^p \in \mathbb{R}^{HW \times D}$ stand for the linearly projected features for query, key, and value. For simplicity, we ignore the multi-head mechanism and feed-forward network (FFN) in the equation.

As shown in Eq. (7.4), when updating the object queries, a *softmax* function is applied to the image resolution (HW), which is typically in the range of thousands of pixels for the task of segmentation. Given the huge number of pixels, it can take many training iterations to learn the attention map, which starts from a uniform distribution at the beginning (as the queries are randomly initialized). Each object query has a difficult time to identify

the most distinguishable features among the abundant pixels in the early stage of training. This behavior is very different from the application of transformers to natural language processing tasks, *e.g.*, neural machine translation [200, 5], where the input and output sequences share a similar short length. Vision tasks, especially segmentation problems, present another challenge for efficiently learning the cross-attention.

Discussion. Similar to cross-attention, self-attention needs to perform a *softmax* function operated along the image resolution. Therefore, learning the attention map for self-attention may also take many training iterations. An efficient alternative, such as axial attention [218] or local attention [149] is usually applied on high resolution feature maps, and thus alleviates the problem, while a solution to cross-attention remains an open question for research.

***k*-means clustering.** In Eq. (7.4), the cross-attention computes the affinity between object queries and pixels (*i.e.*, $\mathbf{Q}^c \times (\mathbf{K}^p)^{\mathrm{T}}$), which is converted to the attention map through the spatial-wise softmax (operated along the image resolution). The attention map is then used to retrieve (and weight accordingly) affiliated pixel features to update the object queries. Surprisingly, we observe that the whole process is actually similar to the classic k-means clustering algorithm [151], which works as follows:

$$\mathbf{A} = \underset{N}{\operatorname{argmax}}(\mathbf{C} \times \mathbf{P}^{\mathrm{T}}), \tag{7.5}$$

$$\hat{\mathbf{C}} = \mathbf{A} \times \mathbf{P}, \tag{7.6}$$

where $\mathbf{C} \in \mathbb{R}^{N \times D}$, $\mathbf{P} \in \mathbb{R}^{HW \times D}$, and $\mathbf{A} \in \mathbb{R}^{N \times HW}$ stand for cluster centers, pixel features, and clustering assignments, respectively.

Comparing Eq. (7.4), Eq. (7.5), and Eq. (7.6), we notice that the k-means clustering algorithm is parameter-free and thus no linear projection is needed for query, key, and value. The updates on cluster centers are not in a residual manner. Most importantly, k-means adopts a *cluster-wise argmax* (*i.e.*, argmax operated along the cluster dimension) instead of the spatial-wise softmax when converting the affinity to the attention map (*i.e.*, weights to retrieve and update features).

This observation motivates us to reformulate the cross-attention in vision problems, especially image segmentation. From a clustering perspective, image segmentation is equivalent to grouping pixels into different clusters, where each cluster corresponds to a predicted mask. However, the cross-attention mechanism, also attempting to group pixels to different object queries, instead employs a different *spatial-wise softmax* operation from the *cluster-wise argmax* as in k-means. Given the success of k-means, we hypothesize that the cluster-wise argmax is a more suitable operation than the spatial-wise softmax regarding pixel clustering, since the cluster-wise argmax performs the hard assignment and efficiently reduces the operation targets from thousands of pixels (HW) to just a few cluster centers (N), which (we will empirically prove) speeds up the training convergence and leads to a better performance.

7.3.3 k-means Mask Transformer

Herein, we first introduce the crucial component of the proposed k-means Mask Transformer, *i.e.*, k-means cross-attention. We then present its meta architecture and model instantiation.

***k*-means cross-attention.** The proposed *k-means cross-attention* reformulates the cross-attention in a manner similar to k-means clustering:

$$\hat{\mathbf{C}} = \mathbf{C} + \underset{N}{\text{argmax}}(\mathbf{Q}^c \times (\mathbf{K}^p)^{\mathrm{T}}) \times \mathbf{V}^p. \tag{7.7}$$

Comparing Eq. (7.4) and Eq. (7.7), the spatial-wise softmax is now replaced by the cluster-wise argmax. As shown in Fig. 7.1, with such a simple yet effective change, a typical transformer decoder could be converted to a kMaX decoder. Unlike the original cross-attention, the proposed k-means cross-attention adopts a different operation (*i.e.*, cluster-wise argmax) to compute the attention map, and does not require the multi-head mechanism [214]. However, the cluster-wise argmax, as a hard assignment to aggregate pixel features for the cluster center update, is not a differentiable operation, posing a challenge during training. We have explored several methods (*e.g.*, Gumbel-Softmax [106]), and discover that a simple deep supervision scheme turns out to be most effective. In particular, in our formulation, the affinity logits between pixel features and cluster centers directly correspond to the softmax logits of segmentation masks (*i.e.*, $\mathbf{Q}^c \times (\mathbf{K}^p)^{\mathrm{T}}$ in Eq. (7.7) corresponds to $\mathbf{F} \times \mathbf{C}^{\mathrm{T}}$ in Eq. (7.3)), since the cluster centers aim to group pixels of similar affinity together to form the predicted segmentation masks. This formulation allows us to add deep supervision to every kMaX decoder, in order to train the parameters in the k-means cross-attention module.

Meta architecture. Fig. 7.2 shows the meta architecture of our proposed kMaX-DeepLab, which contains three main components: pixel encoder, enhanced pixel decoder, and kMaX decoder. The pixel encoder extracts the pixel features either by a CNN [92] or a transformer [149] backbone, while the enhanced pixel decoder is responsible for

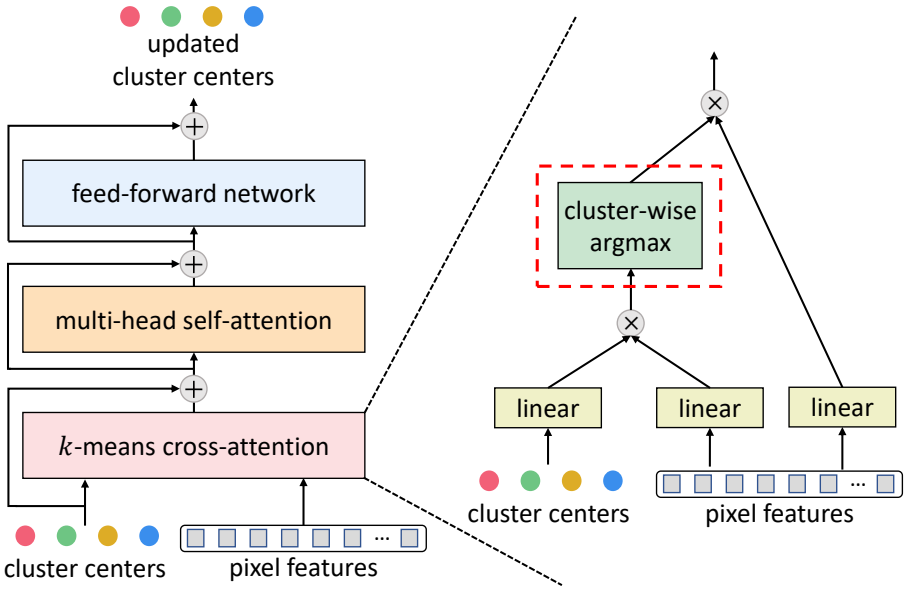

Figure 7.1: To convert a typical transformer decoder into our kMaX decoder, we simply replace the original cross-attention with our k-means cross-attention (*i.e.*, with the only simple change *cluster-wise argmax* high-lighted in red)

recovering the feature map resolution as well as enhancing the pixel features via transformer encoders [214] or axial attention [218]. Finally, the kMaX decoder transforms the object queries (*i.e.*, cluster centers) into mask embedding vectors from the k-means clustering perspective.

Model instantiation. We build kMaX based on MaX-DeepLab [217] with the official code-base [228]. We divide the whole model into two paths: the pixel path and the cluster path, which are responsible for extracting pixel features and cluster centers, respectively. Fig. 7.3 details our kMaX-DeepLab instantiation with two example backbones.

Pixel path. The pixel path consists of a pixel encoder and an enhanced pixel decoder. The pixel encoder is an ImageNet-pretrained [185] backbone, such as ResNet [92], MaX-S [217] (*i.e.*, ResNet-50 with axial attention [218]), and ConvNeXt [150]. Our enhanced

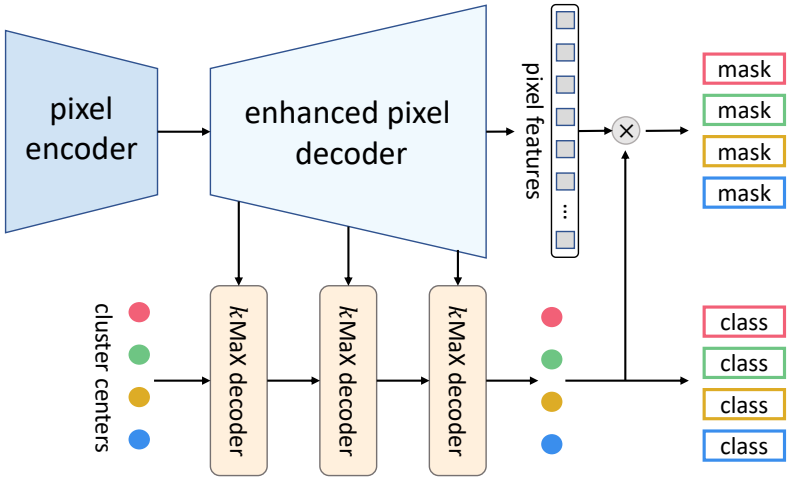

Figure 7.2: The meta architecture of k-means Mask Transformer consists of three components: pixel encoder, enhanced pixel decoder, and kMaX decoder. The pixel encoder is any network backbone. The enhanced pixel decoder includes transformer encoders to enhance the pixel features, and upsampling layers to generate higher resolution features. The series of kMaX decoders transform cluster centers into (1) mask embedding vectors, which multiply with the pixel features to generate the predicted masks, and (2) class predictions for each mask.

pixel decoder consists of several axial attention blocks [218] and bottleneck blocks [92].

Cluster path. The cluster path contains totally six kMaX decoders, which are evenly distributed among features maps of different spatial resolutions. Specifically, we deploy two kMaX decoders each for pixel features at output stride 32, 16, and 8, respectively.

Loss functions. Our training loss functions mostly follow the setting of MaX-DeepLab [217]. We adopt the same PQ-style loss, auxiliary semantic loss, mask-id cross-entropy loss, and pixel-wise instance discrimination loss [253].

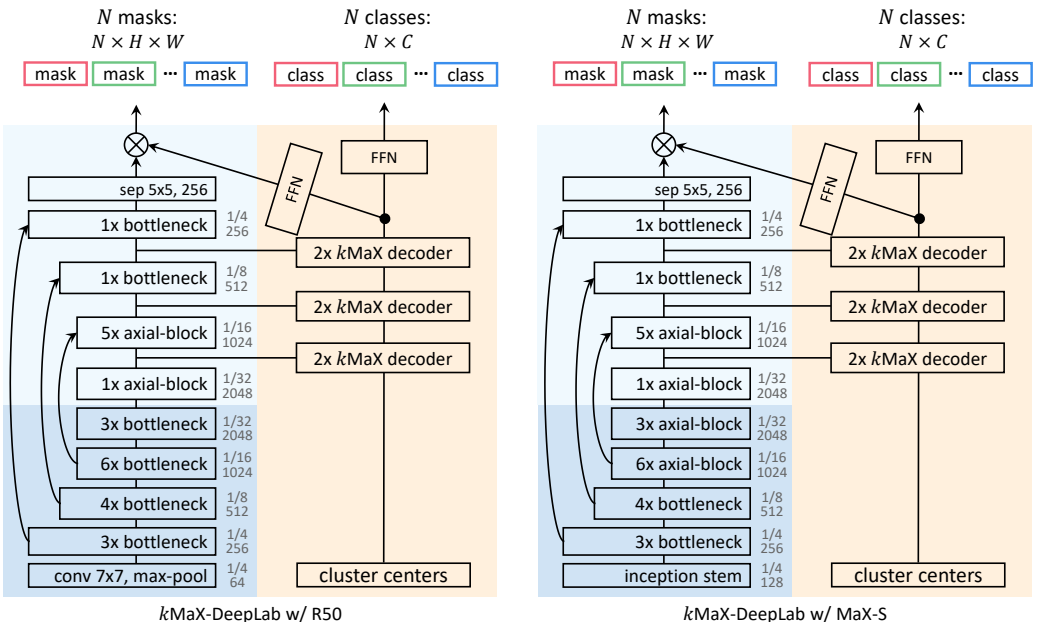

Figure 7.3: An illustration of kMaX-DeepLab with ResNet-50 and MaX-S as backbones. The hidden dimension of FFN is 256. The design of kMaX-DeepLab is general to different backbones by simply updating the pixel encoder (marked in dark-blue). The enhanced pixel decoder and kMaX decoder are colored in light-blue and yellow, respectively

7.4 Experimental Results

In this section, we first provide our implementation details. We report our main results on COCO [138], Cityscapes [54], and ADE20K [277]. We also provide visualizations to better understand the clustering process of the proposed kMaX-DeepLab. The ablation studies are provided in the appendix.

7.4.1 Implementation Details

The meta architecture of the proposed kMaX-DeepLab contains three main components: the pixel encoder, enhanced pixel decoder, and kMaX decoder, as shown in Fig. 7.2. We

provide the implementation details of each component below.

Pixel encoder. The pixel encoder extracts pixel features given an image. To verify the generality of *k*MaX-DeepLab across different pixel encoders, we experiment with ResNet-50 [92], MaX-S [217] (*i.e.*, ResNet-50 with axial attention [218] in the 3rd and 4th stages), and ConvNeXt [150].

Enhanced pixel decoder. The enhanced pixel decoder recovers the feature map resolution and enriches pixel features via self-attention. As shown in Fig. 7.3, we adopt one axial block with channels 2048 at output stride 32, and five axial blocks with channels 1024 at output stride 16. The axial block is a bottleneck block [92], but the 3×3 convolution is replaced by the axial attention [218]. We use one bottleneck block at output stride 8 and 4, respectively. We note that the axial blocks play the same role (*i.e.*, feature enhancement) as the transformer encoders in other works [21, 44, 253], where we ensure that the total number of axial blocks is six for a fair comparison to previous works [21, 44, 253].

Cluster path. As shown in Fig. 7.3, we deploy six *k*MaX decoders, where each two are placed for pixel features (enhanced by the pixel decoders) with output stride 32, 16, 8, respectively. Our design uses six transformer decoders, aligning with the previous works [21, 44, 253], though some recent works [43, 135] adopt more transformer decoders to achieve a stronger performance.

Training and testing. We mainly follow MaX-DeepLab [217] for training settings. The ImageNet-pretrained [185] backbone has a learning rate multiplier 0.1. For regularization and augmentations, we adopt drop path [101], random color jittering [55], and panoptic copy-paste augmentation, which is an extension from instance copy-paste augmentation [68, 74] by augmenting both 'thing' and 'stuff' classes. AdamW [113, 155]

optimizer is used with weight decay 0.05. The k-means cross-attention adopts cluster-wise argmax, which aligns the formulation of attention map to segmentation result. It therefore allows us to directly apply deep supervision on the attention maps. These auxiliary losses attached to each kMaX decoder have the same loss weight of 1.0 as the final prediction, and Hungarian matching result based on the final prediction is used to assign supervisions for all auxiliary outputs. During inference, we adopt the same mask-wise merging scheme used in [44, 267, 135, 253] to obtain the final segmentation results.

COCO dataset. If not specified, we train all models with batch size 64 on 32 TPU cores with 150k iterations (around 81 epochs). The first 5k steps serve as the warm-up stage, where the learning rate linearly increases from 0 to 5×10^{-4}. The input images are resized and padded to 1281×1281. Following MaX-DeepLab [217], the loss weights for PQ-style loss, auxiliary semantic loss, mask-id cross-entropy loss, instance discrimination loss are 3.0, 1.0, 0.3, and 1.0, respectively. The number of cluster centers (*i.e.*, object queries) is 128, and the final feature map resolution has output stride 4 as in MaX-DeepLab [217].

We have also experimented with doubling the number of object queries to 256 for kMaX-DeepLab with ConvNeXt-L, which however leads to a performance loss. Empirically, we adopt a **drop query** regularization, where we randomly drop half of the object queries (*i.e.*, 128) during each training iteration, and all queries (*i.e.*, 256) are used during inference. With the proposed drop query regularization, doubling the number of object queries to 256 consistently brings 0.1% PQ improvement under the large model regime.

Cityscapes dataset. We train all models with batch size 32 on 32 TPU cores with 60k iterations. The first 5k steps serve as the warm-up stage, where learning rate linearly increases from 0 to 3×10^{-4}. The inputs are padded to 1025×2049. The loss weights for

PQ-style loss, auxiliary semantic loss, mask-id cross-entropy loss, and instance discrimination loss are 3.0, 1.0, 0.3, and 1.0, respectively. We use 256 cluster centers, and add an additional bottleneck block in the pixel decoder to produce features with output stride 2.

ADE20K dataset. We adopt the same setting as the COCO dataset, except that the model is trained for 100k iterations. We experiment with both 641×641 and 1281×1281 input resolutions. Our inference is run w.r.t. the whole input image, instead of in the sliding window manner (which may yield a better performance at the cost of more FLOPs).

7.4.2 Main Results

Our main results on the COCO [138], Cityscapes [54], and ADE20K [277] *val* set are summarized in Tab. 7.1, Tab. 7.2, and Tab. 7.3, respectively.

COCO *val* set. In Tab. 7.1, we compare our *k*MaX-DeepLab with other transformer-based panoptic segmentation methods on COCO *val* set. Notably, with a simple ResNet-50 backbone, *k*MaX-DeepLab already achieves 53.0% PQ, surpassing *most* prior arts with stronger backbones. Specifically, *k*MaX-DeepLab outperforms MaskFormer [44] and K-Net [267], all with the ResNet-50 backbone as well, by a large margin of **6.5%** and **5.9%**, while maintaining a similar level of computational costs. Our *k*MaX-DeepLab with ResNet-50 even surpasses the largest variants of MaX-DeepLab [217] by **1.9%** PQ (while using **7.9**× fewer parameters and **22.0**× fewer FLOPs), and MaskFormer (while using **3.7**× fewer parameters and **4.7**× fewer FLOPs) by 0.3% PQ, respectively. With a stronger backbone MaX-S [217], *k*MaX-DeepLab boosts the performance to 56.2% PQ, outperforming MaX-DeepLab with the same backbone by **7.8%** PQ. Our *k*MaX-DeepLab

method	backbone	params	FLOPs	FPS	PQ	PQ^{Th}	PQ^{St}
MaskFormer [44]	ResNet-50 [92]	45M	181G	17.6	46.5	51.0	39.8
K-Net [267]	ResNet-50 [92]	-	-	-	47.1	51.7	40.3
CMT-DeepLab [253]	ResNet-50 [92]	-	-	-	48.5	-	-
Panoptic SegFormer [135]	ResNet-50 [92]	51M	214G	7.8	49.6	54.4	42.4
Mask2Former [43]	ResNet-50 [92]	44M	226G	8.6	51.9	57.7	43.0
kMaX-DeepLab	ResNet-50 [92]	57M	168G	22.8	**53.0**	**58.3**	**44.9**
MaX-DeepLab [217]	MaX-S [217]	62M	324G	-	48.4	53.0	41.5
CMT-DeepLab	MaX-S† [217]	95M	396G	8.1	53.0	57.7	45.9
kMaX-DeepLab	MaX-S† [217]	74M	240G	16.9	**56.2**	**62.2**	**47.1**
MaskFormer [44]	Swin-B (W12)† [149]	102M	411G	8.4	51.8	56.9	44.1
CMT-DeepLab [253]	Axial-R104† [253]	135M	553G	6.0	54.1	58.8	47.1
Panoptic SegFormer [135]	PVTv2-B5† [223]	105M	349G	-	55.4	61.2	46.6
Mask2Former [43]	Swin-B (W12)† [149]	107M	466G	-	56.4	62.4	47.3
kMaX-DeepLab	ConvNeXt-B† [150]	122M	380G	11.6	**57.2**	**63.4**	**47.8**
MaX-DeepLab [217]	MaX-L [217]	451M	3692G	-	51.1	57.0	42.2
MaskFormer [44]	Swin-L (W12)† [149]	212M	792G	5.2	52.7	58.5	44.0
K-Net [267]	Swin-L (W7)† [149]	-	-	-	54.6	60.2	46.0
CMT-DeepLab [253]	Axial-R104-RFN† [175]	270M	1114G	3.2	55.3	61.0	46.6
Panoptic SegFormer [135]	Swin-L (W7)† [149]	221M	816G	-	55.8	61.7	46.9
Mask2Former [43]	Swin-L (W12)† [149]	216M	868G	4.0	57.8	64.2	48.1
kMaX-DeepLab	ConvNeXt-L† [150]	232M	744G	6.7	57.9	64.0	48.6
kMaX-DeepLab*	ConvNeXt-L† [150]	232M	749G	6.6	58.0	64.2	48.6
kMaX-DeepLab‡	ConvNeXt-L† [150]	232M	744G	6.7	**58.1**	**64.3**	**48.8**

Table 7.1: COCO *val* set results. Our FLOPs and FPS are evaluated with the input size 1200×800 and a Tesla V100-SXM2 GPU. †: ImageNet-22K pretraining. $\star$: Using 256 object queries with drop query regularization. ‡: Using COCO *unlabeled* set

with MaX-S backbone also improves over the previous state-of-art K-Net with Swin-L [149] by **1.6%** PQ. To further push the envelope, we adopt the modern CNN backbone ConvNeXt [150] and set new state-of-the-art results of 57.2% PQ with ConvNeXt-B and 58.0% PQ with ConvNeXt-L, outperforming K-Net with Swin-L by a significant margin of **3.4%** PQ.

When compared to more recent works (CMT-DeepLab [253], Panoptic SegFormer [135], and Mask2Former [43]), *k*MaX-DeepLab still shows great performances without the advanced modules, such as deformable attention [284], cascaded transformer decoder [43],

and uncertainty-based pointly supervision [116]. As different backbones are utilized for each method (*e.g.*, PVTv2 [223], Swin [149], and ConvNeXt [150]), we start with a fair comparison using the ResNet-50 backbone. Our *k*MaX-DeepLab with ResNet-50 achieves a significant better performance compared to CMT-DeepLab, Panoptic SegFormer and Mask2Former by a large margin of **4.5%**, **3.4%**, and **1.1%** PQ, respectively. Additionally, our model runs almost **3**× faster than them (since *k*MaX-DeepLab enjoys a simple design without deformable attention). When employing stronger backbones, *k*MaX-DeepLab with ConvNeXt-B outperforms CMT-DeepLab with Axial-R104, Panoptic SegFormer with PVTv2-B5, and Mask2Former with Swin-B (window size 12) by **3.1%**, **1.8%**, and **0.8%** PQ, respectively, while all models have a similar level of cost (parameters and FLOPs). When scaling up to the largest backbone for each method, *k*MaX-DeepLab outperforms CMT-DeepLab, and Panoptic SegFormer significantly by **2.7%** and **2.2%** PQ. Although we already perform better than Mask2Former with Swin-L (window size 12), we notice that *k*MaX-DeepLab benefits much less than Mask2Former when scaling up from base model to large model (+0.7% for *k*MaX-DeepLab but +1.4% for Mask2Former), indicating *k*MaX-DeepLab's strong representation ability and that it may overfit on COCO *train* set with the largest backbone. Therefore, we additionally perform a simple experiment to alleviate the over-fitting issue by generating pseudo labels [26] on COCO *unlabeled* set. Adding pseudo labels to the training data slightly improves *k*MaX-DeepLab, yielding a PQ score of **58.1%** (the drop query regularization is not used here and the number of object query remains 128).

Cityscapes *val* set. In Tab. 7.2, we compare our *k*MaX-DeepLab with other state-of-art methods on Cityscapes *val* set. Our reported PQ, AP, and mIoU results use the same

panoptic model to provide a comprehensive comparison. Notably, *k*MaX-DeepLab with ResNet-50 backbone already surpasses most baselines, while being more efficient. For example, *k*MaX-DeepLab with ResNet-50 achieves **1.3%** PQ higher performance compared to Panoptic-DeepLab [42] (Xception-71 [48] backbone) with **20%** computational cost (FLOPs) reduced. Moreover, it achieves a similar performance to Axial-DeepLab-XL [218], while using **3.1**× fewer parameters and **5.6**× fewer FLOPs. *k*MaX-DeepLab achieves even higher performances with stronger backbones. Specifically, with MaX-S backbone, it performs on par with previous state-of-the-art Panoptic-DeepLab with SWideRNet [33] backbone, while using **7.2**× fewer parameters and **17.2**× fewer FLOPs. Additionally, even only trained with panoptic annotations, our *k*MaX-DeepLab also shows superior performance in instance segmentation (AP) and semantic segmentation (mIoU). Finally, we provide a comparison with the recent work Mask2Former [43], where the advantage of our *k*MaX-DeepLab becomes even more significant. Using the ResNet-50 backbone for a fair comparison, *k*MaX-DeepLab achieves **2.2%** PQ, **1.2%** AP, and **2.2%** mIoU higher performance than Mask2Former. For other backbone variants with a similar size, *k*MaX-DeepLab with ConvNeXt-B is **1.9%** PQ higher than Mask2Former with Swin-B (window size 12). Notably, *k*MaX-DeepLab with ConvNeXt-B already obtains a PQ score that is **1.4%** higher than Mask2Former with their best backbone. With ConvNeXt-L as backbone, *k*MaX-DeepLab sets a new state-of-the-art record of 68.4% PQ without any test-time augmentation or COCO [138]/Mapillary Vistas [164] pretraining.

ADE20K *val* set. In Tab. 7.3, we summarize *k*MaX-DeepLab's performance on ADE20K against other state-of-the-art methods. We report PQ and mIoU results with the same panoptic model, where *k*MaX-DeepLab consistently shows better performance.

method	backbone	params	FLOPs	FPS	PQ	AP	mIoU
Panoptic-DeepLab [42]	Xception-71 [48]	47M	548G	5.7	63.0	35.3	80.5
Axial-DeepLab [218]	Axial-ResNet-L [218]	45M	687G	-	63.9	35.8	81.0
Axial-DeepLab [218]	Axial-ResNet-XL [218]	173M	2447G	-	64.4	36.7	80.6
CMT-DeepLab [253]	MaX-S [217]	-	-	-	64.6	-	81.4
Panoptic-DeepLab [42]	SWideRNet-(1,1,4.5) [33]	536M	10365G	1.0	66.4	40.1	82.2
Mask2Former [43]	ResNet-50 [92]	-	-	-	62.1	37.3	77.5
Mask2Former [43]	Swin-B (W12)† [149]	-	-	-	66.1	42.8	82.7
Mask2Former [43]	Swin-L (W12)† [149]	-	-	-	66.6	43.6	82.9
SETR [273]	ViT-L† [63]	-	-	-	-	-	79.3
SegFormer [235]	MiT-B5 [235]	85M	1460G	2.5	-	-	82.4
Mask R-CNN [89]	ResNet-50 [92]	-	-	-	-	31.5	-
PANet [144]	ResNet-50 [92]	-	-	-	-	36.5	-
kMaX-DeepLab	ResNet-50 [92]	56M	434G	9.0	64.3	38.5	79.7
kMaX-DeepLab	MaX-S† [217]	74M	602G	6.5	66.4	41.6	82.1
kMaX-DeepLab	ConvNeXt-B† [150]	121M	858G	5.2	68.0	43.0	83.1
kMaX-DeepLab	ConvNeXt-L† [150]	232M	1673G	3.1	**68.4**	**44.0**	**83.5**

Table 7.2: Cityscapes *val* set results. We only consider methods without extra data [138, 164] and test-time augmentation for a fair comparison. We evaluate FLOPs and FPS with the input size 1025×2049 and a Tesla V100-SXM2 GPU. Our instance (AP) and semantic (mIoU) results are based on the same panoptic model (*i.e.*, no task-specific fine-tuning). †: ImageNet-22K pretraining

Specifically, with ResNet-50 and input size 641×641, *k*MaX-DeepLab attains 41.5% PQ. Increasing the input size to 1281×1281 further improves the performance to 42.3% PQ, significantly outperforming the prior state-of-the-art Mask2Former with ResNet-50 backbone by 2.6% PQ. Finally, *k*MaX-DeepLab equipped with the modern ConvNeXt-L backone achieves a new state-of-the-art performance of 50.9% PQ, signicantly surpassing MaskFormer, Panoptic-DeepLab with SWideRNet, and Mask2Former by 16.2%, 13.0%, and 2.8% PQ, respectively.

Visualizations. In Fig. 7.4, we provide a visualization of pixel-cluster assignments at each *k*MaX decoder and final prediction, to better understand the working mechanism behind *k*MaX-DeepLab. Another benefit of *k*MaX-DeepLab is that with the cluster-wise argmax, visualizations can be directly drawn as segmentation masks, as the pixel-cluster

method	backbone	params	FLOPs	FPS	PQ	mIoU
MaskFormer [44]	ResNet-50 [92]	-	-	-	34.7	-
Panoptic-DeepLab [42]	SWideRNet-(1,1.5,3) [33]	-	-	-	37.4	50.4
Panoptic-DeepLab [42]	SWideRNet-(1,1,4) [33]	-	-	-	37.9	50.0
Mask2Former [43]	ResNet-50 [92]	-	-	-	39.7	46.1
Mask2Former [43]	Swin-L (W12)† [149]	-	-	-	48.1	54.5
*k*MaX-DeepLab (641)	ResNet-50 [92]	57M	75G	38.7	41.5	45.0
*k*MaX-DeepLab (1281)	ResNet-50 [92]	57M	295G	14.4	42.3	45.3
*k*MaX-DeepLab (641)	ConvNeXt-L† [150]	232M	333G	14.0	48.7	54.8
*k*MaX-DeepLab (1281)	ConvNeXt-L† [150]	232M	1302G	4.0	50.9	55.2

Table 7.3: ADE20K *val* set results. Our FLOPs and FPS are evaluated with the input size (641×641 or 1281×1281) and a Tesla V100-SXM2 GPU. †: ImageNet-22K pretraining. The input size for *k*MaX-DeepLab is shown in the parentheses

assignments are exclusive to each other with cluster-wise argmax. Noticeably, the major clustering update happens in the first three stages, which already updates cluster centers well and generates reasonable clustering results, while the following stages mainly focus on refining details. This coincides with our observation that 3 *k*MaX decoders are sufficient to produce good results. Besides, we observe that 1st clustering assignment tends to produce over-segmentation effects, where many clusters are activated and then combined or pruned in the later stages. Moreover, though there exist many fragments in the first round of clustering, it already surprisingly distinguishes different semantics, especially some persons are already well clustered, which indicates that the initial clustering is not only based on texture or location, but also depends on the underlying semantics. Another visualization is shown in Fig. 7.5, where we observe that *k*MaX-DeepLab behaves in a part-to-whole manner to capture an instance. More experimental results (*e.g.*, ablation studies, test set results) and visualizations are available in the appendix.

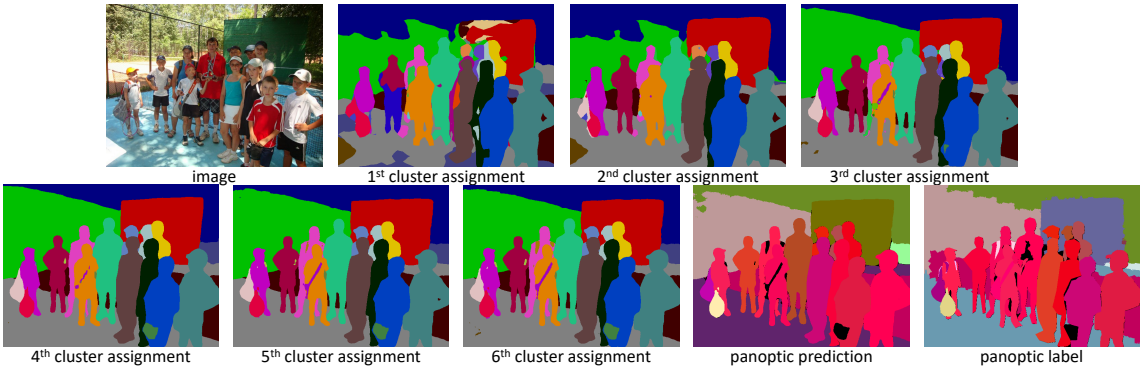

Figure 7.4: Visualization of kMaX-DeepLab (ResNet-50) pixel-cluster assignments at each kMaX decoder stage, along with the final panoptic prediction. In the cluster assignment visualization, pixels with same color are assigned to the same cluster and their features will be aggregated for updating corresponding cluster centers

7.5 Conclusion

In this chapter, we have presented a novel end-to-end framework, called k-means Mask Transformer (kMaX-DeepLab), for segmentation tasks. kMaX-DeepLab rethinks the relationship between pixel features and object queries from the clustering perspective. Consequently, it simplifies the mask-transformer model by replacing the multi-head cross attention with the proposed single-head k-means clustering. We have tailored the transformer-based model for segmentation tasks by establishing the link between the traditional k-means clustering algorithm and cross-attention. We hope our work will inspire the community to develop more vision-specific transformer models.

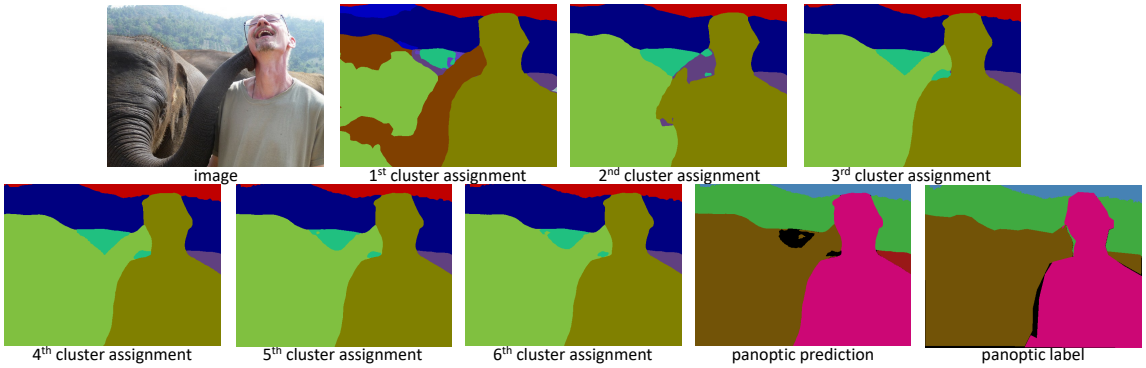

Figure 7.5: Visualization of *k*MaX-DeepLab (ResNet-50) pixel-cluster assignments at each *k*MaX decoder stage, along with the final panoptic prediction. *k*MaX-DeepLab shows a behavior of recognizing objects starting from their parts to their the whole shape in the clustering process. For example, the elephant's top head, body, and nose are separately clustered at the beginning, and they are gradually merged in the following stages

Chapter 8

Conclusion

8.1 Summary

In this dissertation, I have presented a total of 6 research projects. All of them aim at providing a path towards better segmentation models in 2D or 3D images. In Chapter 2, we study a disentangled search space to enable a consistent neural architecture search for 3D segmentation models. In Chapter 3, we further propose to push the search space towards channel level, which enables a more fine-grained model configuration and brings better accuracy-cost trade-off. In Chapter 4, we introduce a coarse-to-fine mechanism and mask-guided matting, which refines a coarse segmentation mask with matting-level details. In Chapter 5, we also show that the coarse-to-fine mechanism, specifically, glance and gaze, can help build a strong hierarchical vision transformer backbone and serves as a strong feature extractor. In Chapter 6, we rethink the relationship among mask transformer, panoptic segmentation, and traditional clustering algorithm, which motivates us to reformulate the cross-attention and proposes clustering mask transformer. Furthermore in Chapter 7, we focus on the k-means clustering and modify the attention mechanism

accordingly. With a simple change, we achieve much better results with more plausible attention visualizations.

8.2 Future Work

In this section, I will discuss a few potential future research objectives.

8.2.1 Unsupervised Object-Centric Tokenization.

While we have illustrated promising ways to generate strong pixel-level representation [255] and grouping pixel-level representation to object-level representation [252, 254], these methods heavily rely on the dataset and annotations. Therefore, it is an interesting question about whether object-centric tokenization may emerge under a large-scale unsupervised manner, which provides a more general representation and avoids dataset bias (*e.g.*, we observe that kMaX-DeepLab performs extremely well on “person", due to it is one most common and challenging class on COCO dataset).

The unsupervised methods include but are not limited to (a) for obtaining a strong pixel-level feature, we may expect a pretraining in the style of pre-text tasks [166, 76], contrastive learning [37, 88], and masked image modeling [11, 87], (b) for grouping pixel-level representation into object-level, it can be promising to draw inspirations from VQ-VAE [213], SAVI++ [64], which provide a reasonable way to pretrain with image/mask reconstruction.

8.2.2 Unified Model across Tasks and Modalities.

One most important features of transformer is to provide the possibility of a unified model for different tasks and modalities. *E.g.*, Perceiver IO [105] proposes a shared architecture that can be used to tackle different tasks, Omnivore [77] tries to unify the image and video within a single model. More recently, Unified-IO [157] shows promising results to handle language and vision while supporting various tasks using a single model.

Though these methods show good and promising results, it is noticeable that they heavily rely on a pretrained VQ-VAE [213] and VQ-GAN [65] to handle the vision tasks. Nonetheless, the tokenization with these models is not semantic enough, which actually incurs huge computation and memory costs, as well as the need of training a strong transformer encoder as well. In this case, building a unified model with object-centric tokenization can be promising in terms of both efficiency and performance.

Bibliography

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE TPAMI*, 2012.

[2] Joshua Ainslie, Santiago Ontanon, Chris Alberti, Philip Pham, Anirudh Ravula, and Sumit Sanghai. Etc: Encoding long and structured data in transformers. In *EMNLP*, 2020.

[3] Yağiz Aksoy, Tae-Hyun Oh, Sylvain Paris, Marc Pollefeys, and Wojciech Matusik. Semantic soft segmentation. *ACM Transactions on Graphics (TOG)*, 37(4):1–13, 2018.

[4] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *ICCV*, 2021.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[6] Min Bai and Raquel Urtasun. Deep watershed transform for instance segmentation. In *CVPR*, 2017.

[7] Yutong Bai, Qing Liu, Lingxi Xie, Weichao Qiu, Yan Zheng, and Alan L Yuille. Semantic part detection via matching: Learning to generalize to novel viewpoints from limited training data. In *ICCV*, pages 7535–7545, 2019.

[8] Yutong Bai, Angtian Wang, Adam Kortylewski, and Alan Yuille. Coke: Localized contrastive learning for robust keypoint detection. *arXiv preprint arXiv:2009.14115*, 2020.

[9] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *ICLR*, 2017.

[10] Dana H Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 1981.

[11] Hangbo Bao, Li Dong, and Furu Wei. Beit: Bert pre-training of image transformers. 2022.

[12] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. In *ICCV*, 2019.

[13] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv:2004.05150*, 2020.

[14] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *ICML*, pages 550–559, 2018.

[15] Ujwal Bonde, Pablo F Alcantarilla, and Stefan Leutenegger. Towards bounding-box free panoptic segmentation. *arXiv:2002.07705*, 2020.

[16] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Smash: one-shot model architecture search through hypernetworks. *ICLR*, 2018.

[17] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *CVPR*, 2005.

[18] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *ICLR*, 2019.

[19] Shaofan Cai, Xiaoshuai Zhang, Haoqiang Fan, Haibin Huang, Jiangyu Liu, Jiaming Liu, Jiaying Liu, Jue Wang, and Jian Sun. Disentangled image matting. In *ICCV*, pages 8819–8828, 2019.

[20] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, 2018.

[21] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.

[22] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, pages 6299–6308, 2017.

[23] Hao Chen, Qi Dou, Lequan Yu, Jing Qin, and Pheng-Ann Heng. Voxresnet: Deep voxelwise residual networks for brain segmentation from 3d mr images. *NeuroImage*, 170:446–455, 2018.

[24] Jieneng Chen, Yongyi Lu, Qihang Yu, Xiangde Luo, Ehsan Adeli, Yan Wang, Le Lu, Alan L Yuille, and Yuyin Zhou. Transunet: Transformers make strong encoders for medical image segmentation. 2021.

[25] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[26] 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.

[27] 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.

[28] 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*, 40(4):834–848, 2017.

[29] 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.

[30] 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. *PAMI*, 40(4):834–848, 2018.

[31] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[32] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv:1706.05587*, 2017.

[33] Liang-Chieh Chen, Huiyu Wang, and Siyuan Qiao. Scaling wide residual networks for panoptic segmentation. *arXiv:2011.11675*, 2020.

[34] Liang-Chieh Chen, Yi Yang, Jiang Wang, Wei Xu, and Alan L Yuille. Attention to scale: Scale-aware semantic image segmentation. In *CVPR*, 2016.

[35] 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.

[36] Qifeng Chen, Dingzeyu Li, and Chi-Keung Tang. Knn matting. *TPAMI*, 35(9):2175–2188, 2013.

[37] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

[38] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. *ICCV*, 2019.

[39] Yunpeng Chen, Yannis Kalantidis, Jianshu Li, Shuicheng Yan, and Jiashi Feng. Aˆ 2-nets: Double attention networks. In *NeurIPS*, 2018.

[40] Bowen Cheng, Anwesa Choudhuri, Ishan Misra, Alexander Kirillov, Rohit Girdhar, and Alexander G Schwing. Mask2former for video instance segmentation. *arXiv:2112.10764*, 2021.

[41] 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.

[42] Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-DeepLab: A Simple, Strong, and Fast Baseline for Bottom-Up Panoptic Segmentation. In *CVPR*, 2020.

[43] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. *CVPR*, 2022.

[44] Bowen Cheng, Alexander G Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. In *NeurIPS*, 2021.

[45] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. In *EMNLP*, 2016.

[46] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv:1904.10509*, 2019.

[47] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[48] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017.

[49] Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Conditional positional encodings for vision transformers. *Arxiv*

preprint 2102.10882, 2021.

[50] Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Jixiang Li. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. *arXiv preprint arXiv:1907.01845*, 2019.

[51] Yung-Yu Chuang, Brian Curless, David H Salesin, and Richard Szeliski. A bayesian approach to digital matting. In *CVPR*, volume 2, pages II–II. IEEE, 2001.

[52] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3D u-net: learning dense volumetric segmentation from sparse annotation. In *MICCAI*, 2016.

[53] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020.

[54] 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.

[55] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. In *CVPR*, 2019.

[56] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

[57] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, 2017.

[58] 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.

[59] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[60] Heiner Deubel and Werner X. Schneider. Saccade target selection and object recognition: Evidence for a common attentional mechanism. *Vision Research*, 36(12):1827–1837, 1996.

[61] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[62] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *CVPR*, pages 1761–1770, 2019.

[63] 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*, 2021.

[64] Gamaleldin F Elsayed, Aravindh Mahendran, Sjoerd van Steenkiste, Klaus Greff, Michael C Mozer, and Thomas Kipf. Savi++: Towards end-to-end object-centric

learning from real-world videos. *arXiv:2206.07764*, 2022.

[65] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021.

[66] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010.

[67] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *ICCV*, 2021.

[68] Hao-Shu Fang, Jianhua Sun, Runzhong Wang, Minghao Gou, Yong-Lu Li, and Cewu Lu. Instaboost: Boosting instance segmentation via probability map guided copy-pasting. In *ICCV*, 2019.

[69] Marco Forte and François Pitié. f, b, alpha matting. *arXiv preprint arXiv:2003.07711*, 2020.

[70] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *CVPR*, 2019.

[71] 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.

[72] Peng Gao, Minghang Zheng, Xiaogang Wang, Jifeng Dai, and Hongsheng Li. Fast convergence of detr with spatially modulated co-attention. In *ICCV*, 2021.

[73] Eduardo SL Gastal and Manuel M Oliveira. Shared sampling for real-time alpha matting. In *Computer Graphics Forum*, volume 29, pages 575–584. Wiley Online Library, 2010.

[74] Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D Cubuk, Quoc V Le, and Barret Zoph. Simple copy-paste is a strong data augmentation method for instance segmentation. In *CVPR*, 2021.

[75] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *CVPR*, pages 7036–7045, 2019.

[76] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. 2018.

[77] Rohit Girdhar, Mannat Singh, Nikhila Ravi, Laurens van der Maaten, Armand Joulin, and Ishan Misra. Omnivore: A single model for many visual modalities. In *CVPR*, 2022.

[78] Felix Gonda, Donglai Wei, Toufiq Parag, and Hanspeter Pfister. Parallel separable 3d convolution for video and volumetric data understanding. *BMVC*, 2018.

[79] Ariel Gordon, Elad Eban, Ofir Nachum, Bo Chen, Hao Wu, Tien-Ju Yang, and Edward Choi. Morphnet: Fast & simple resource-constrained structure learning of deep networks. In *CVPR*, pages 1586–1595, 2018.

[80] 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.

[81] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *ICCV*, volume 1, page 5, 2017.

[82] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. *arXiv preprint arXiv:1904.00420*, 2019.

[83] Ankit Gupta and Jonathan Berant. Gmat: Global memory augmentation for transformers. *arXiv:2006.03274*, 2020.

[84] Vikas Gupta and Shanmuganathan Raman. Automatic trimap generation for image matting. In *2016 International Conference on Signal and Information Processing (IConSIP)*, pages 1–5. IEEE, 2016.

[85] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *arXiv preprint arXiv:2103.00112*, 2021.

[86] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In *ECCV*, 2014.

[87] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

[88] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

[89] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

[90] Kaiming He, Christoph Rhemann, Carsten Rother, Xiaoou Tang, and Jian Sun. A global sampling method for alpha matting. In *CVPR*, pages 2049–2056. IEEE, 2011.

[91] Kaiming He, Jian Sun, and Xiaoou Tang. Fast matting using large kernel matting laplacian matrices. In *CVPR*, pages 2165–2172. IEEE, 2010.

[92] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[93] Xuming He, Richard S Zemel, and Miguel Á Carreira-Perpiñán. Multiscale conditional random fields for image labeling. In *CVPR*, 2004.

[94] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[95] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv:1912.12180*, 2019.

[96] Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry. Augment your batch: Improving generalization through instance repetition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8129–8138, 2020.

[97] Qiqi Hou and Feng Liu. Context-aware image matting for simultaneous foreground and alpha estimation. In *ICCV*, pages 4130–4139, 2019.

[98] Chang-Lin Hsieh and Ming-Sui Lee. Automatic trimap generation for digital image matting. In *2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, pages 1–5. IEEE, 2013.

[99] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In *CVPR*, 2018.

[100] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *ICCV*, 2019.

[101] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *ECCV*, 2016.

[102] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *ICCV*, 2019.

[103] 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.

[104] Fabian Isensee, Jens Petersen, Andre Klein, David Zimmerer, Paul F Jaeger, Simon Kohl, Jakob Wasserthal, Gregor Koehler, Tobias Norajitra, Sebastian Wirkert, et al. nnu-net: Self-adapting framework for u-net-based medical image segmentation. *arXiv:1809.10486*, 2018.

[105] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. 2022.

[106] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *ICLR*, 2017.

[107] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. In *NeurIPS*, 2016.

[108] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, 2018.

[109] Margret Keuper, Evgeny Levinkov, Nicolas Bonneel, Guillaume Lavoué, Thomas Brox, and Bjorn Andres. Efficient decomposition of image and mesh graphs by lifted multicuts. In *ICCV*, 2015.

[110] 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.

[111] Dahun Kim, Jun Xie, Huiyu Wang, Siyuan Qiao, Qihang Yu, Hong-Seok Kim, Hartwig Adam, In So Kweon, and Liang-Chieh Chen. Tubeformer-deeplab: Video mask transformer. In *CVPR*, 2022.

[112] Sungwoong Kim, Ildoo Kim, Sungbin Lim, Woonhyuk Baek, Chiheon Kim, Hyungjoo Cho, Boogeon Yoon, and Taesup Kim. Scalable neural architecture search for 3d medical image segmentation. *arXiv preprint arXiv:1906.05956*, 2019.

[113] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[114] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *CVPR*, 2019.

[115] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In *CVPR*, 2019.

[116] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *CVPR*, 2020.

[117] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *ICLR*, 2020.

[118] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

[119] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[120] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[121] Philip Lee and Ying Wu. Nonlocal matting. In *CVPR*, pages 2193–2200. IEEE, 2011.

[122] Bastian Leibe, Ales Leonardis, and Bernt Schiele. Combined object categorization and segmentation with an implicit shape model. In *Workshop on statistical learning in computer vision, ECCV*, 2004.

[123] Anat Levin, Dani Lischinski, and Yair Weiss. A closed-form solution to natural image matting. *TPAMI*, 30(2):228–242, 2007.

[124] Anat Levin, Alex Rav-Acha, and Dani Lischinski. Spectral matting. *TPAMI*, 30(10):1699–1712, 2008.

[125] Jie Li, Allan Raventos, Arjun Bhargava, Takaaki Tagawa, and Adrien Gaidon. Learning to fuse things and stuff. *arXiv:1812.01192*, 2018.

[126] Qizhu Li, Xiaojuan Qi, and Philip HS Torr. Unifying training and inference for panoptic segmentation. In *CVPR*, 2020.

[127] Xiangtai Li, Wenwei Zhang, Jiangmiao Pang, Kai Chen, Guangliang Cheng, Yunhai Tong, and Chen Change Loy. Video k-net: A simple, strong, and unified baseline for video segmentation. In *CVPR*, 2022.

[128] 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.

[129] Yingwei Li, Xiaojie Jin, Jieru Mei, Xiaochen Lian, Linjie Yang, Cihang Xie, Qihang Yu, Yuyin Zhou, Song Bai, and Alan Yuille. Autonl: Neural architecture search for lightweight non-local networks in mobile vision. In *CVPR*, 2020.

[130] Yingwei Li, Xiaojie Jin, Jieru Mei, Xiaochen Lian, Linjie Yang, Cihang Xie, Qihang Yu, Yuyin Zhou, Song Bai, and Alan L Yuille. Neural architecture search for lightweight non-local networks. In *CVPR*, 2020.

[131] Yaoyi Li and Hongtao Lu. Natural image matting via guided contextual attention. In *AAAI*, volume 34, pages 11450–11457, 2020.

[132] Yingwei Li, Qihang Yu, Mingxing Tan, Jieru Mei, Peng Tang, Wei Shen, Alan Yuille, and Cihang Xie. Shape-texture debiased neural network training. In *ICLR*, 2020.

[133] Yanwei Li, Hengshuang Zhao, Xiaojuan Qi, Liwei Wang, Zeming Li, Jian Sun, and Jiaya Jia. Fully convolutional networks for panoptic segmentation. In *CVPR*, 2021.

[134] Yingwei Li, Zhuotun Zhu, Yuyin Zhou, Yingda Xia, Wei Shen, Elliot K Fishman, and Alan L Yuille. Volumetric medical image segmentation: A 3d deep coarse-to-fine

framework and its adversarial examples. In *Deep Learning and Convolutional Neural Networks for Medical Imaging and Clinical Informatics*, pages 69–91. Springer, 2019.

[135] Zhiqi Li, Wenhai Wang, Enze Xie, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, Tong Lu, and Ping Luo. Panoptic segformer. *CVPR*, 2022.

[136] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *ICCV*, pages 7083–7093, 2019.

[137] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[138] 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.

[139] 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.

[140] Huanyu Liu, Chao Peng, Changqian Yu, Jingbo Wang, Xu Liu, Gang Yu, and Wei Jiang. An end-to-end network for panoptic segmentation. In *CVPR*, 2019.

[141] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *ICLR*, 2019.

[142] Jinlin Liu, Yuan Yao, Wendi Hou, Miaomiao Cui, Xuansong Xie, Changshui Zhang, and Xian-sheng Hua. Boosting semantic human matting with coarse annotations. In *CVPR*, pages 8563–8572, 2020.

[143] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *NeurIPS*, 2018.

[144] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *CVPR*, 2018.

[145] Siqi Liu, Daguang Xu, S Kevin Zhou, Thomas Mertelmeier, Julia Wicklein, Anna Jerebko, Sasa Grbic, Olivier Pauly, Weidong Cai, and Dorin Comaniciu. 3d anisotropic hybrid network: Transferring convolutional features from 2d images to 3d anisotropic volumes. *MICCAI*, 2017.

[146] Siqi Liu, Daguang Xu, S Kevin Zhou, Olivier Pauly, Sasa Grbic, Thomas Mertelmeier, Julia Wicklein, Anna Jerebko, Weidong Cai, and Dorin Comaniciu. 3d anisotropic hybrid network: Transferring convolutional features from 2d images to 3d anisotropic volumes. In *MICCAI*, pages 851–858. Springer, 2018.

[147] 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.

[148] 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.

[149] 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.

[150] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *CVPR*, 2022.

[151] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[152] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *NeurIPS*, 2020.

[153] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[154] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[155] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[156] Hao Lu, Yutong Dai, Chunhua Shen, and Songcen Xu. Indices matter: Learning to index for deep image matting. In *ICCV*, pages 3266–3275, 2019.

[157] Jiasen Lu, Christopher Clark, Rowan Zellers, Roozbeh Mottaghi, and Aniruddha Kembhavi. Unified-io: A unified model for vision, language, and multi-modal tasks. *arXiv:2206.08916*, 2022.

[158] Chenxu Luo and Alan Yuille. Grouped spatial-temporal aggretation for efficient action recognition. In *ICCV*, 2019.

[159] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In *EMNLP*, 2015.

[160] Sebastian Lutz, Konstantinos Amplianitis, and Aljosa Smolic. Alphagan: Generative adversarial networks for natural image matting. *BMVC*, 2018.

[161] Jieru Mei, Yingwei Li, Xiaochen Lian, Xiaojie Jin, Linjie Yang, Alan Yuille, and Jianchao Yang. Atomnas: Fine-grained end-to-end neural architecture search. *ICLR*, 2020.

[162] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *3DV*, 2016.

[163] Aliasghar Mortazi and Ulas Bagci. Automatically designing cnn architectures for medical image segmentation. In *MLMI*, pages 98–106. Springer, 2018.

[164] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *ICCV*, 2017.

[165] Davy Neven, Bert De Brabandere, Marc Proesmans, and Luc Van Gool. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In *CVPR*, 2019.

[166] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016.

[167] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, et al. Attention u-net: Learning where to look for the pancreas. *MIDL*, 2018.

[168] Dim P Papadopoulos, Jasper RR Uijlings, Frank Keller, and Vittorio Ferrari. Extreme clicking for efficient object annotation. In *ICCV*, 2017.

[169] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *ICML*, 2018.

[170] Mathias Perslev, Erik Bjørnager Dam, Akshay Pai, and Christian Igel. One network to segment them all: A general, lightweight system for accurate 3d medical image segmentation. In *MICCAI*, pages 30–38. Springer, 2019.

[171] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *ICML*, 2018.

[172] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

[173] Lorenzo Porzi, Samuel Rota Bulò, Aleksander Colovic, and Peter Kontschieder. Seamless scene segmentation. In *CVPR*, 2019.

[174] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. *arXiv:2006.02334*, 2020.

[175] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. In *CVPR*, 2021.

[176] Yu Qiao, Yuhao Liu, Xin Yang, Dongsheng Zhou, Mingliang Xu, Qiang Zhang, and Xiaopeng Wei. Attention-guided hierarchical structure aggregation for image matting. In *CVPR*, pages 13676–13685, 2020.

[177] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *ICCV*, pages 5533–5541, 2017.

[178] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10428–10436, 2020.

[179] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. *arXiv preprint arXiv:1906.05909*, 2019.

[180] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. In *NeurIPS*, 2019.

[181] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *AAAI*, volume 33, pages 4780–4789, 2019.

[182] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.

[183] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 2015.

[184] Holger R Roth, Le Lu, Amal Farag, Hoo-Chang Shin, Jiamin Liu, Evrim B Turkbey, and Ronald M Summers. Deeporgan: Multi-level deep convolutional networks for automated pancreas segmentation. In *MICCAI*, 2015.

[185] 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*, 115:211–252, 2015.

[186] Michael S Ryoo, AJ Piergiovanni, Mingxing Tan, and Anelia Angelova. Assemblenet: Searching for multi-stream neural connectivity in video architectures. *arXiv preprint arXiv:1905.13209*, 2019.

[187] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.

[188] Soumyadip Sengupta, Vivek Jayaram, Brian Curless, Steven M Seitz, and Ira Kemelmacher-Shlizerman. Background matting: The world is your green screen. In *CVPR*, pages 2291–2300, 2020.

[189] Alexander Sergeev and Mike Del Balso. Horovod: fast and easy distributed deep learning in tensorflow. *arXiv preprint arXiv:1802.05799*, 2018.

[190] Ehsan Shahrian, Deepu Rajan, Brian Price, and Scott Cohen. Improving image matting using comprehensive sampling sets. In *CVPR*, pages 636–643, 2013.

[191] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *NAACL*, 2018.

[192] Xiaoyong Shen, Xin Tao, Hongyun Gao, Chao Zhou, and Jiaya Jia. Deep automatic portrait matting. In *ECCV*, pages 92–107. Springer, 2016.

[193] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Efficient attention: Attention with linear complexities. In *WACV*, 2021.

[194] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

[195] Amber L Simpson, Michela Antonelli, Spyridon Bakas, Michel Bilello, Keyvan Farahani, Bram van Ginneken, Annette Kopp-Schneider, Bennett A Landman, Geert Litjens, Bjoern Menze, et al. A large annotated medical image dataset for the development and evaluation of segmentation algorithms. *arXiv:1902.09063*, 2019.

[196] Konstantin Sofiiuk, Olga Barinova, and Anton Konushin. Adaptis: Adaptive instance selection network. In *ICCV*, 2019.

[197] Dimitrios Stamoulis, Ruizhou Ding, Di Wang, Dimitrios Lymberopoulos, Bodhi Priyantha, Jie Liu, and Diana Marculescu. Single-path nas: Designing hardware-efficient convnets in less than 4 hours. *ECML PKDD*, 2019.

[198] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *ICCV*, 2021.

[199] Jian Sun, Jiaya Jia, Chi-Keung Tang, and Heung-Yeung Shum. Poisson matting. In *SIGGRAPH*, pages 315–321. 2004.

[200] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *NeurIPS*, 2014.

[201] Nima Tajbakhsh, Jae Y Shin, Suryakanth R Gurudu, R Todd Hurst, Christopher B Kendall, Michael B Gotway, and Jianming Liang. Convolutional neural networks for medical image analysis: Full training or fine tuning? *TMI*, 35(5):1299–1312, 2016.

[202] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[203] Mingxing Tan and Quoc V Le. Mixconv: Mixed depthwise convolutional kernels. *BMVC*, 2019.

[204] Jingwei Tang, Yagiz Aksoy, Cengiz Oztireli, Markus Gross, and Tunc Ozan Aydin. Learning-based sampling for natural image matting. In *CVPR*, pages 3055–3063, 2019.

[205] Zhi Tian, Chunhua Shen, and Hao Chen. Conditional convolutions for instance segmentation. In *ECCV*, 2020.

[206] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *CVPR*, 2014.

[207] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020.

[208] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, pages 4489–4497, 2015.

[209] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *ICCV*, pages 5552–5561, 2019.

[210] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*,

pages 6450–6459, 2018.

[211] 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.

[212] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[213] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. 2017.

[214] 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.

[215] Luc Vincent and Pierre Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE TPAMI*, 1991.

[216] Haochen Wang, Ruotian Luo, Michael Maire, and Greg Shakhnarovich. Pixel consensus voting for panoptic segmentation. In *CVPR*, 2020.

[217] 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.

[218] 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.

[219] Jue Wang and Michael F Cohen. Optimized color sampling for robust matting. In *CVPR*, pages 1–8. IEEE, 2007.

[220] Jue Wang and Michael F Cohen. *Image and video matting: a survey*. Now Publishers Inc, 2008.

[221] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, pages 20–36. Springer, 2016.

[222] Sinong Wang, Belinda Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv:2006.04768*, 2020.

[223] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvtv2: Improved baselines with pyramid vision transformer. *arXiv:2106.13797*, 2021.

[224] Wenhai 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. *arXiv preprint arXiv:2102.12122*, 2021.

[225] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018.

[226] Xiaolong Wang and Abhinav Gupta. Videos as space-time region graphs. In *ECCV*, pages 399–417, 2018.

[227] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. SOLOv2: Dynamic and fast instance segmentation. In *NeurIPS*, 2020.

[228] 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 L. Yuille, Florian Schroff, Hartwig Adam, and Liang-Chieh Chen. DeepLab2: A TensorFlow Library for Deep Labeling. *arXiv: 2106.09748*, 2021.

[229] Yu Weng, Tianbao Zhou, Yujie Li, and Xiaoyu Qiu. Nas-unet: Neural architecture search for medical image segmentation. *IEEE Access*, 7:44247–44257, 2019.

[230] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. *arXiv preprint arXiv:2103.15808*, 2021.

[231] Yingda Xia, Fengze Liu, Dong Yang, Jinzheng Cai, Lequan Yu, Zhuotun Zhu, Daguang Xu, Alan Yuille, and Holger Roth. 3d semi-supervised learning with uncertainty-aware multi-view co-training. *WACV*, 2020.

[232] Yingda Xia, Lingxi Xie, Fengze Liu, Zhuotun Zhu, Elliot K Fishman, and Alan L Yuille. Bridging the gap between 2d and 3d organ segmentation with volumetric fusion net. In *MICCAI*, pages 445–453. Springer, 2018.

[233] Yingda Xia, Qihang Yu, Linda Chu, Satomi Kawamoto, Seyoun Park, Fengze Liu, Jieneng Chen, Zhuotun Zhu, Bowen Li, Zongwei Zhou, et al. The felix project: Deep networks to detect pancreatic neoplasms. *medRxiv*, pages 2022–09, 2022.

[234] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 418–434, 2018.

[235] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. In *NeurIPS*, 2021.

[236] Lingxi Xie and Alan Yuille. Genetic cnn. In *ICCV*, pages 1379–1388, 2017.

[237] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

[238] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *ECCV*, pages 305–321, 2018.

[239] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *ICCV*, pages 1395–1403, 2015.

[240] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. Upsnet: A unified panoptic segmentation network. In *CVPR*, 2019.

[241] Jiarui Xu, Shalini De Mello, Sifei Liu, Wonmin Byeon, Thomas Breuel, Jan Kautz, and Xiaolong Wang. Groupvit: Semantic segmentation emerges from text supervision. In *CVPR*, 2022.

[242] Ning Xu, Brian Price, Scott Cohen, and Thomas Huang. Deep image matting. In *CVPR*, pages 2970–2979, 2017.

[243] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient differentiable architecture search. *ICLR*, 2020.

[244] Antoine Yang, Pedro M Esperança, and Fabio M Carlucci. Nas evaluation is frustratingly hard. *ICLR*, 2020.

[245] Chenglin Yang, Yilin Wang, Jianming Zhang, He Zhang, Zijun Wei, Zhe Lin, and Alan Yuille. Lite vision transformer with enhanced self-attention. In *CVPR*, 2022.

[246] Dong Yang, Holger Roth, Ziyue Xu, Fausto Milletari, Ling Zhang, and Daguang Xu. Searching learning strategy with reinforcement learning for 3d medical image segmentation. In *MICCAI*, pages 3–11. Springer, 2019.

[247] 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.

[248] Yibo Yang, Hongyang Li, Xia Li, Qijie Zhao, Jianlong Wu, and Zhouchen Lin. Sognet: Scene overlap graph network for panoptic segmentation. In *AAAI*, 2020.

[249] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[250] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. Bignas: Scaling up neural architecture search with big single-stage models. *ECCV*, 2020.

[251] Qihang Yu, Yingwei Li, Jieru Mei, Yuyin Zhou, and Alan Yuille. Cakes: Channel-wise automatic kernel shrinking for efficient 3d networks. In *AAAI*, 2021.

[252] Qihang Yu, Huiyu Wang, Dahun Kim, Siyuan Qiao, Maxwell Collins, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Cmt-deeplab: Clustering mask transformers for panoptic segmentation. In *CVPR*, 2022.

[253] Qihang Yu, Huiyu Wang, Dahun Kim, Siyuan Qiao, Maxwell Collins, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Cmt-deeplab: Clustering mask transformers for panoptic segmentation. In *CVPR*, 2022.

[254] Qihang Yu, Huiyu Wang, Siyuan Qiao, Maxwell Collins, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. k-means mask transformer. In *ECCV*, 2022.

[255] Qihang Yu, Yingda Xia, Yutong Bai, Yongyi Lu, Alan L Yuille, and Wei Shen. Glance-and-gaze vision transformer. In *NeurIPS*, 2021.

[256] Qihang Yu, Yingda Xia, Yutong Bai, Yongyi Lu, Alan L Yuille, and Wei Shen. Glance-and-gaze vision transformer. *NeurIPS*, 2021.

[257] Qihang Yu, Lingxi Xie, Yan Wang, Yuyin Zhou, Elliot K Fishman, and Alan L Yuille. Recurrent saliency transformation network: Incorporating multi-stage visual cues for small organ segmentation. In *CVPR*, pages 8280–8289, 2018.

[258] Qihang Yu, Dong Yang, Holger Roth, Yutong Bai, Yixiao Zhang, Alan L Yuille, and Daguang Xu. C2fnas: Coarse-to-fine neural architecture search for 3d medical image segmentation. In *CVPR*, 2020.

[259] Qihang Yu, Dong Yang, Holger Roth, Yutong Bai, Yixiao Zhang, Alan L Yuille, and Daguang Xu. C2fnas: Coarse-to-fine neural architecture search for 3d medical image segmentation. In *CVPR*, pages 4126–4135, 2020.

[260] Qihang Yu, Dong Yang, Holger Roth, Yutong Bai, Yixiao Zhang, Alan L Yuille, and Daguang Xu. C2fnas: Coarse-to-fine neural architecture search for 3d medical image segmentation. In *CVPR*, pages 4126–4135, 2020.

[261] Qihang Yu, Jianming Zhang, He Zhang, Yilin Wang, Zhe Lin, Ning Xu, Yutong Bai, and Alan Yuille. Mask guided matting via progressive refinement network. In *CVPR*, 2021.

[262] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021.

[263] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032, 2019.

[264] 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.

[265] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[266] He Zhang, Jianming Zhang, Federico Perazzi, Zhe Lin, and Vishal M Patel. Deep image compositing. *WACV*, 2021.

[267] Wenwei Zhang, Jiangmiao Pang, Kai Chen, and Chen Change Loy. K-net: Towards unified image segmentation. In *NeurIPS*, 2021.

[268] Yunke Zhang, Lixue Gong, Lubin Fan, Peiran Ren, Qixing Huang, Hujun Bao, and Weiwei Xu. A late fusion cnn for digital matting. In *CVPR*, pages 7469–7478, 2019.

[269] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10076–10085, 2020.

[270] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017.

[271] 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.

[272] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. *arXiv preprint arXiv:2012.15840*, 2020.

[273] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *CVPR*, 2021.

[274] Yuanjie Zheng and Chandra Kambhamettu. Learning based digital matting. In *ICCV*, pages 889–896. IEEE, 2009.

[275] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13001–13008, 2020.

[276] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *ECCV*, pages 803–818, 2018.

[277] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, 2017.

[278] Yuyin Zhou, Yingwei Li, Zhishuai Zhang, Yan Wang, Angtian Wang, Elliot K Fishman, Alan L Yuille, and Seyoun Park. Hyper-pairing network for multi-phase pancreatic ductal adenocarcinoma segmentation. In *MICCAI*, pages 155–163. Springer, 2019.

[279] Yuyin Zhou, Lingxi Xie, Wei Shen, Yan Wang, Elliot K Fishman, and Alan L Yuille. A fixed-point model for pancreas segmentation in abdominal ct scans. In *MICCAI*, pages 693–701. Springer, 2017.

[280] Yuyin Zhou, Qihang Yu, Yan Wang, Lingxi Xie, Wei Shen, Elliot K Fishman, and Alan L Yuille. 2d-based coarse-to-fine approaches for small target segmentation in abdominal ct scans. In *Deep Learning and Convolutional Neural Networks for Medical Imaging and Clinical Informatics*, pages 43–67. Springer, 2019.

[281] Yi Zhou, Hui Zhang, Hana Lee, Shuyang Sun, Pingjun Li, Yangguang Zhu, ByungIn Yoo, Xiaojuan Qi, and Jae-Joon Han. Slot-vps: Object-centric representation learning for video panoptic segmentation. In *CVPR*, 2022.

[282] Song Chun Zhu and Alan Yuille. Region competition: Unifying snakes, region growing, and bayes/mdl for multiband image segmentation. *IEEE TPAMI*, 1996.

[283] Xizhou Zhu, Dazhi Cheng, Zheng Zhang, Stephen Lin, and Jifeng Dai. An empirical study of spatial attention mechanisms in deep networks. In *ICCV*, 2019.

[284] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *ICLR*, 2021.

[285] Zhuotun Zhu, Chenxi Liu, Dong Yang, Alan Yuille, and Daguang Xu. V-nas: Neural architecture search for volumetric medical image segmentation. *3DV*, 2019.

[286] Zhuotun Zhu, Yingda Xia, Wei Shen, Elliot K Fishman, and Alan L Yuille. A 3d coarse-to-fine framework for automatic pancreas segmentation. *3DV*, 2018.

[287] Zhuotun Zhu, Yingda Xia, Wei Shen, Elliot K. Fishman, and Alan L. Yuille. A 3d coarse-to-fine framework for volumetric medical image segmentation. In *3DV*, 2018.

[288] Zhen Zhu, Mengde Xu, Song Bai, Tengteng Huang, and Xiang Bai. Asymmetric non-local neural networks for semantic segmentation. In *CVPR*, 2019.

[289] Mohammadreza Zolfaghari, Kamaljeet Singh, and Thomas Brox. Eco: Efficient convolutional network for online video understanding. In *ECCV*, pages 695–712, 2018.

[290] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *ICLR*, 2017.

[291] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *CVPR*, pages 8697–8710, 2018.

Vita

Qihang Yu is completing his Ph.D. degree of Computer Science at the Johns Hopkins University, under the supervision of Bloomberg Distinguished Professor Alan L. Yuille. Qihang received his B.S. degree in Computer Science from Peking University in 2018. Qihang's research interests lie in the fields of computer vision, deep learning, and medical image analysis.