UNIVERSITY OF TECHNOLOGY SYDNEY
Faculty of Engineering and Information Technology

Context-aware Image Semantic Segmentation

by

Ye Huang

A Thesis Submitted
in Fulfillment of the
Requirements for the Degree

Doctor of Philosophy

Sydney, Australia

2022

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, Ye Huang, declare that this thesis, is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the School of Electrical and Data Engineering, Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Signature: Production Note: Signature removed prior to publication.

Date: May-06-2022

ABSTRACT

Context-aware Image Semantic Segmentation

by

Ye Huang

Semantic segmentation is a fundamental task for computer vision applications. However, the existing solutions have many issues when handling difficult cases. This thesis develops three novel approaches which have improved the generalization ability of the existing solutions at significantly reduced computation costs. Extensive experiments conducted on multiple benchmark datasets have demonstrated the superior performance of the proposed approaches.

Scale-invariant: The state-of-the-art semantic segmentation solutions usually leverage different receptive fields via multiple parallel branches to handle objects of different sizes. However, employing separate kernels for individual branches degrades the generalization of the network to objects with different scales, and the computational cost increases with the increase of the number of branches. In this thesis, a novel network structure, namely *Kernel-Sharing Atrous Convolution (KSAC)*, is proposed, where branches with different receptive fields share the same kernel, i.e., letting a single kernel "see" the input feature maps more than once with different receptive fields.

Seamless dual attention: Spatial and channel attentions, modelling the semantic inter-dependencies in spatial and channel dimensions respectively, have recently been widely used for semantic segmentation. However, computing spatial attention and channel attention separately sometimes causes errors, especially in those difficult cases. In this research, a Channelized Axial Attention (CAA) is developed to seamlessly *integrate* channel attention and spatial attention into a single operation with negligible computation overhead. Furthermore, a novel grouped vectorization approach is developed to allow the proposed model to run with very little memory consumption without slowing down the computation.

Class-aware regularization: Recent segmentation methods utilizing class-level information in addition to pixel features have achieved notable success in boosting the accuracy of existing network models. However, the extracted class-level information was simply concatenated to pixel features, without being explicitly exploited to learn better pixel representation. Moreover, these approaches learn soft class centers based on coarse mask prediction, which is prone to error accumulation. Motivated by the fact that humans can recognize an object by itself no matter which other objects it appears with and aiming to use class-level information more effectively, a universal Class-Aware Regularization (CAR) approach is proposed to optimize the intra-class variance and inter-class distance during feature learning. Furthermore, the class center in the proposed approach is directly generated from ground truth instead of from the error-prone coarse prediction. The proposed CAR can be easily applied to most existing segmentation models and can largely improve their accuracy at no additional inference overhead.

Dissertation directed by Prof. Xiangjian He and Dr Wenjing Jia
School of Electrical and Data Engineering

Dedication

Dedicated to my family. Dedicated to the world peace.

Acknowledgements

First and foremost, I would like to acknowledge my supervisors Prof. Xiangjian He and Dr. Wenjing Jia for their continuous and endless supervisions and encouragements.

I would like to thank my lab mates, as well as the Dr. Qingqing Wang, Dr. Yue Xi, Dr. Lei Liu, Dr. Saeed Amirgholipour, Dr Mohammadhesam Hesamian, Zhanzhong Gu, Yuanfang Zhang and Chengpei Xu for their assistance or/and encouragements.

I also would like to appreciate the research funding supports from Byker Digital Biotechnology Co. Ltd

Finally, I am particularly grateful to all people who directly or indirectly provided the supports on my research and technical issues, including the netizens in GitHub, and the authors of the papers that I cited.

Ye Huang

Sydney, Australia, 2022.

List of Publications

Journal Papers

1. **Ye Huang**, Qingqing Wang, Wenjing Jia and Xiangjian He, See More Than Once–Kernel-Sharing Atrous Convolution for Semantic Segmentation, in Neurocomputing, Volume 443, 5 July 2021, Pages 26-34.

2. Qingqing Wang, **Ye Huang**, Wenjing Jia, Xiangjian He, Michael Blumenstein, Shujing Lyu and Yue Lu, FACLSTM: ConvLSTM with focused attention for scene text recognition, Science China Information Sciences 2020

Conference Papers

1. **Ye Huang**, Di Kang, Liang Chen, Xuefei Zhe, Wenjing Jia, Xiangjian He, Linchao Bao, CAR: Class-aware Regularizations for Semantic Segmentation, accepted by ECCV 2022.

2. **Ye Huang**, Di Kang, Wenjing Jia, Xiangjian He and Liu Liu, Channelized Axial Attention – Considering Channel Relation within Spatial Attention for Semantic Segmentation, Proceedings of the AAAI Conference on Artificial Intelligence, 36(1), 1016-1025.

3. Qingqing Wang, Wenjing Jia, Xiangjian He, Yue Lu, Michael Blumenstein, **Ye Huang** and Shujing Lyu, DeepText: Detecting text from the wild with multi-ASPP-assembled deeplab, Proceedings of 2019 International Conference on Document Analysis and Recognition

Contents

List of Figures

List of Tables

Abbreviation

- H - Height
- W - Width
- Channels - The size of last dimension of the 4D feature map.
- Encoding - Nerual network encoded 3 channels image input to the faeture map with multiple channels.
- mIOU - Mean Intersection over Union
- KSAC - Kernel Sharing Astrous Convolution
- CAA - Channelized Axial Attention
- CAR - Class-aware Regularization
- OS - Output stride [3]
- FCN - Fully Convolutional Networks [37]
- ASPP - Atrous Spatial Pyramid Pooling [4]
- PPM - Pyramid Pooling module [69]
- FPN - Feature Pyramid Networks [32]
- SA - Self-attention [53]
- ACFNet - Attentional Class Feature Network [64]
- OCR - Object-Contextual Representations [61]
- CPNet - Context Prior Network [60]

Chapter 1

Introduction

Recent advances in computer vision techniques have been largely fueled by the advances in deep learning techniques. As a classical and foundation computer vision application, semantic segmentation, also referred to as ‘dense prediction‘ or ‘pixels classification‘, assigns pixels belonging to the same object class with the same label. It is more precise compared to the similar task image classification, which usually

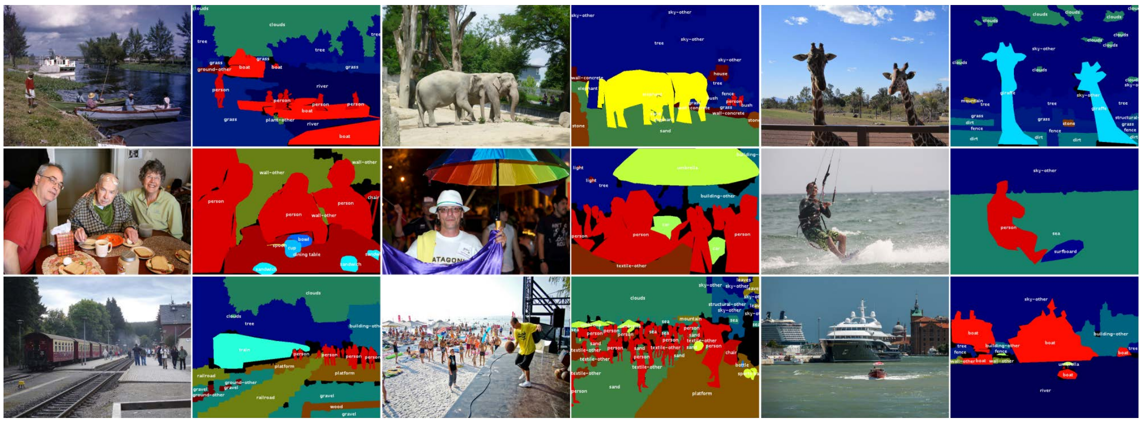


Figure 1.1 : Examples of Semantic Segmentation results on COCOStuff [1] dataset

only predicts the center of the foreground object (*e.g.*, an image that has a dog on the ground with grass will only be classified as ”dog” in image classification). On the other hand, semantic segmentation is much more challenging than image classification, because semantic segmentation requires all the pixels in the image to be accurately classified.

Semantic segmentation has a wide range of applications, including but not limited to scene understanding, medical image processing and remote sensing analysis.

It also benefits many downstream applications, *e.g.*, object detection/instance segmentation [59, 24, 6], and depth estimation.

During the segmentation procedure, deep neural networks are required to handle both local detailed and global semantic information, so as to handle objects of arbitrary sizes. To achieve such robustness, numerous efforts have been made by the research community.

1.1 Research Topics in Semantic Segmentation

1.1.1 Context Aggregation

Given a pixel from an input image with RGB channels, regardless of human or machine, it is impossible to predict its class with pixel independent transformation. For humans, we need to see more pixels and collect more information to know which object a pixel belongs to. For machines, they also need to collect more information from the spatial domain (*e.g.*, from other pixels) to enhance the cognition of that pixel, where the information collection process is called "context aggregation". Spatial context aggregation uses the encoding (*i.e.*, features) of other pixels to enhance the encoding of the pixel itself. After more information is aggregated, the encoding of that pixel now includes the context information, and the neural network is able to classify which object (class) the pixel belongs.

The problem is how to efficiently and correctly aggregate information to allow the machines to recognise the objects correctly. It is obviously meaningless to directly merge all pixel encoding together. Performing a visual perception of this world from local textures to intermediate objects/shapes and then to the entire scene is a common pattern, and it should also be the pattern of visual perception required for computer vision. Moreover, it is computationally expensive to directly aggregate the encoding of all pixels. Many methods have been proposed to tackle the context aggregation problem, from the simplest 3×3 convolutions, to the widely adopted self-attention nowadays. In this thesis, we will mainly discuss and research the context aggregation problem.

1.1.2 High Resolution and Details

To reduce computational overhead during context aggregation, the most common way is to downsample feature maps during the process of context aggregation and encoding enhancement, which is typically achieved by striding or pooling operations. The downsampled feature maps require much smaller computation resources than using the original channel number. For example, downsampling a feature map with striding $= 2$ produces feature maps four times smaller, since both height and width are reduced. Thus, downsampled feature maps can now contain four times wider encoding (channels) with the same capability of hardware. On the other hand, the distance between a pixel and its neighbouring pixels in a $2\times$ downsampled features map was shorter than before downsampling. Therefore, it is easier to conduct context aggregation and capture global context than before.

However, for image semantic segmentation, the same size as that of the input image is required for outputs/predictions. This brings a significant challenge to the downsampled feature map, since the downsampling rate achieving the best trade-off between performance and computational cost is 1/8 by using the dilation operation [3]. It means that the final feature map is required to upsample to eight times to perform the final prediction, and therefore many details will be lost, easily resulting in inaccurate object boundaries.

Recent works [45, 32, 56, 5] leverage low-level high-resolution features from the backbone to refine the details and boundary information of the downsampled high-level features during their upsampling. HRNet [52] always keeps a high-resolution branch (only 1/4 of the original size) to maintain the discrimibility of the feature of details and the accuracy of the boundaries. Boundary supervision [63, 28, 49] is another important solution, where DecoupleSegNet [28] decoupled edge and body features for the original features and gave hard supervision (*e.g.*, hard pixel mining) to the edges to enhance the robustness of edge prediction.

1.2 Early Context Aggregation based Approaches (prior to 2016)

Since the introduction of the convolution layer, the earliest spatial context aggregation operation that has been proposed 20 years ago, a huge number of works have emerged focusing on improving the context aggregation operation.

Here, the receptive field is an important concept in context aggregation, where 3×3 convolution layers encode the eight pixels around the center pixel and itself, and then use the aggregated feature from those nine pixels to augment the value of the center pixel. This means the context aggregation only considers a 3×3 receptive field.

AlexNet [25] and VGG [47] were two proposed deep networks and they proved that the sequential deep spatial CNN layers could increase the size of the receptive field and largely improved the accuracy of image classification. In Fully Convolutional Networks (FCNs) [37], VGG was used as the backbone to perform the pixels' dense prediction.

1.3 Fixed Range Multi-scale Context Aggregation (2017-2018)

Besides the increment of the receptive field in deep convolution layers, parallel designs have also been considered, and they use fixed-range multi-scale features to perform context aggregation in a large receptive field. The Pyramid pooling module (PPM) [69] and atrous spatial pyramid pooling (ASPP) [3] are two well-known

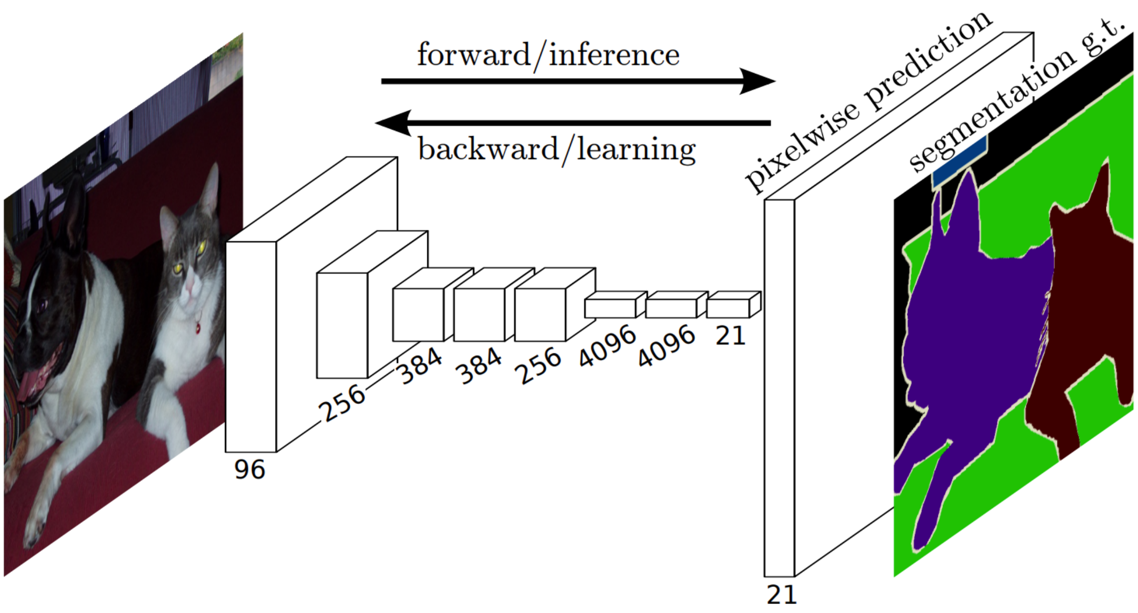


Figure 1.2 : FCN [37] based on VGG outputs pixel level predictions

implementations and have been widely used.

PPM was proposed in PSPNet [69], which expanded the receptive fields using multiple average pooling modules with different sizes, where the information from both short-range pooling and long-range pooling was aggregated. Fig 1.3 illustrates the design of PSPNet [69].

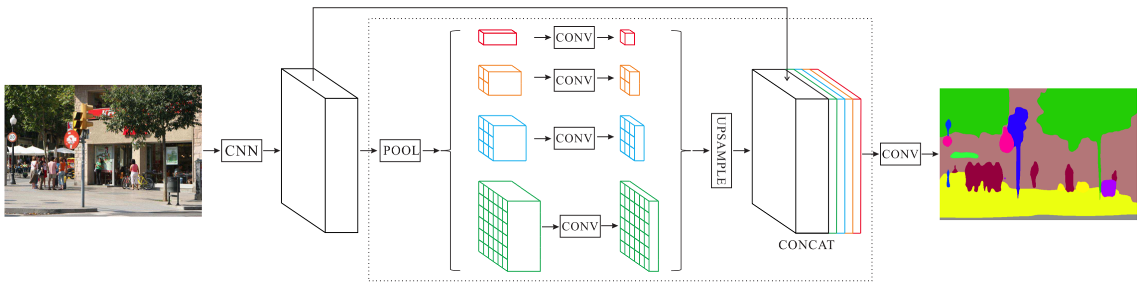


Figure 1.3 : The design of PSPNet [69], which uses PPM that contains multiple pooling operations with different size rates to aggregate the multi-scale context.

ASPP was proposed in DeepLab [3]. As shown on Fig. 1.4, instead of using pooling to merge information, it uses multiple dilated convolutions with different dilation rates to capture multi-range information. Both approaches performed very well in some early datasets with simple context, *e.g.*, Pascal VOC 2012 [13] and Cityscapes [38].

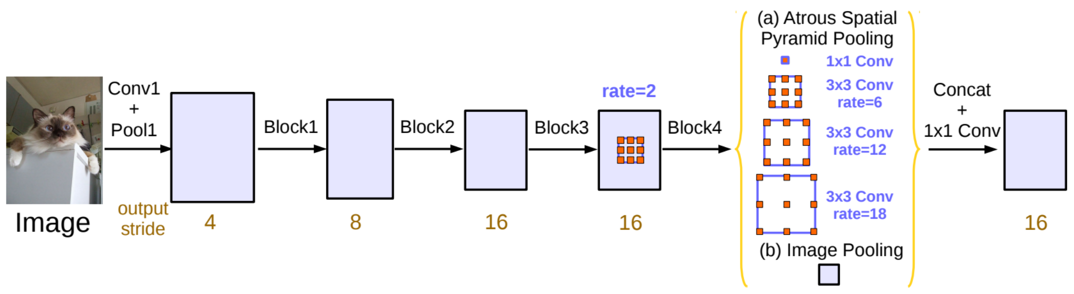


Figure 1.4 : The design of DeepLab V3 [4], which uses ASPP containing multiple dilated (atrous) convolution of different dilation rates to aggregate multi-scale context.

However, those fixed-range multi-scale context aggregation approaches have many issues, resulting in insufficient accuracy in some datasets with complex context (*e.g.*, Pascal Context [41] and COCOStuff [1]). Using ASPP as an example, with its default settings, *i.e.*, output stride = 16, and a dilation rate = (6, 12, 18) to capture short-range, mid-range and long-range information, it is obvious that most of the information outside the dilation rate = (6, 12, 18) has been ignored during the context aggregation for the center pixel. Moreover, capturing multi-range context separately with different kernels can result in scale-sensitive representations of the intra-class pixels. The intra-class pixels can end up with having different encodings only because the object it belongs to appears in different scales (*e.g.*, a large motorbike *vs* a small motorbike), so that the accuracy of the final pixel classification is affected.

DenseASPP [58] refined the structure of ASPP to allow more dilation rates to be performed to cover more pixels ignored by regular ASPP, but there were still many pixels missing, and the computational cost was increased greatly. In fact, pixel miss consideration issue has basically been solved with the introduction of self-attention (Non-Local) [53, 50], which has been applied in computer vision models, with all pixels being considered during context aggregation. The details of self-attention will be discussed in the next section.

For another issue, scale-variant representation can be caused by multi-scale context. In this thesis, Kernel-Sharing Atrous Convolution is proposed as an effective solution, which brings multi-scale feature invariant during context aggregation. See Chapter 2 for details.

1.4 Context Aggregation with Attention Mechanism (since 2018)

Regular self-attention, proposed in [50, 53], aggregates context information from all pixels and directly brings the full receptive field in spatial domain. In other words, unlike the fixed-range multi-scale context aggregation that misses many pixels outside the fixed ranges, all pixels are independently considered with self-attention. Specifically, for any pixel in the feature map that needs to aggregate information from other pixels to augment itself, self-attention firstly computes the dot-product similarity of features between all pixels and itself, where the similarity is called "attention map", and then uses the feature sum of all pixels, weighted by "attention map" to augment that pixel. The global consideration of pixels significantly improves the accuracy of neural networks on datasets with complex contexts and yields more smooth results.

There have been many works that provide improving variants of self-attention. Since self-attention in the spatial domain aggregates information from all pixels for each of the pixels in the feature map, its computation complexity is $H \times W \times H \times W$, which may not be hardware friendly for low-end devices or large images.

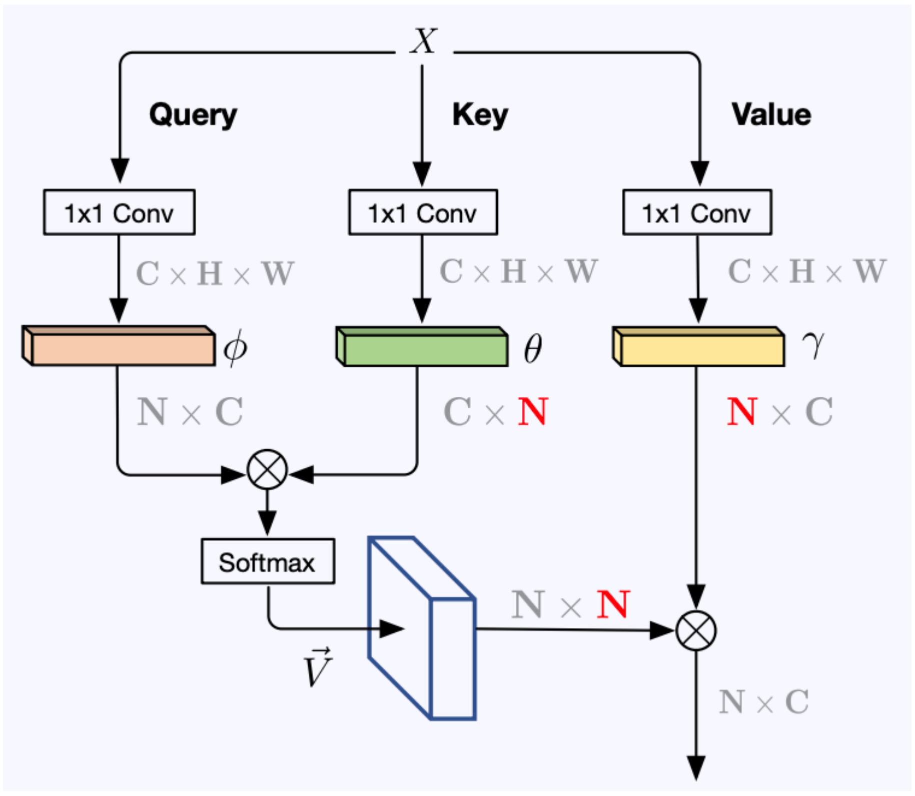


Figure 1.5 : The designs of Self-Attention [53]. It compares the dot-product similarities between each of 2 pixels on the feature map, and then uses it as the weights to aggregate context from all pixels on the feature map. Images come from [72]

Works like [72, 19, 23, 62] use sparse self-attention to reduce computational cost, for example, in CCNet [23], instead of considering all pixels at once that generates $H \times W \times H \times W$ computational overhead, criss-cross attention decomposed it into two sparse attentions. Each attention only aggregates information in "criss-cross" regions (same column or same row) for each pixel, where all pixels are considered indirectly at the second round aggregation, resulting in similar effects of regular self-attention but with much less computational complexity $(2 \times (H + W) \times H \times W)$.

Extending self-attention to other domains, such as channels, is another variant improving self-attention. DANet [14] performed channel attention and spatial attention separately and added them together to augment the final pixel feature. Thus, not only all pixels but also the dimensions are considered, so that the comprehensive coverage of context information collection is further expanded.

However, considering each dimension separately and then merging them together may cause issues (*e.g.*, feature conflicts), where the spatial attention or channel attention may focus on different aspects, so it can affect the final feature. To address this issue, we propose a “Channelized Axial-Attention” that seamlessly integrates channel and spatial attention together. See Chapter 3 for details.

1.5 Class-aware Context Aggregation (since 2020)

To further improve the accuracy and efficiency during context aggregation. The concept of class representation has been explored in many works in recent years. In fact, many works [54] have already considered the class representation in other areas of computer vision. In semantic segmentation, it started with ACFNet [64]. ACFNet aggregates the class center from the dot products between context(feature map) and coarse prediction, uses it to generate the intra-class map (distribute class-center to the corresponding pixels by dot-product with coarse prediction), and then concatenates it with the original pixel features to perform the final classification.

A similar concept also exists in OCRNet [61], where the generation from class-center to intra-class map is slightly different from ACFNet. OCRNet uses the cross-attention between the class center and the feature map to distribute the class center to the corresponding pixel.

Furthermore, CPNet [60] used a prior map (a large convolution that has a filter size equal to the $height \times width$ of the feature map) to directly generate the intra-class map, where the prior map is supervised by the ground truth with an affinity loss. It also generates an inter-class map, based on the class center of inter-class features. After that, similar to ACFNet and OCRNet, both ‘class centers’ are concatenated with the original pixel features to perform the final classification.

Obviously, the generation of class center maps is a major contribution of ACFNet, OCR and CPNet. However, their usage of the class center is simple, and it is directly combined with the original pixel feature. It is unclear how much effect the class center map brings to final classification results. For example, the final classifier may focus more on the original features of pixels, and reduce the effectiveness of the aggregated class center information. Moreover, during the context aggregation of class-center, ACFNet and OCRNet rely on coarse prediction, and CPNet relies on a prior map (which is another form of coarse prediction), so they can be affected by incorrect information.

To address the above issues, This thesis proposed a "Class-aware Regularization" module (CAR), which uses a class center to supervise the context aggregations of the existing networks. In short, our CAR uses the real ground truth to compute the class center during training to avoid aggregating incorrect information, and uses it to supervise any existing networks with three goals: 1) reducing pixels' intra-class distance, 2) reducing inter-class center-to-center dependency, and 3) reducing pixels' inter-class dependency. With those three targets, the context aggregation process in existing networks is implicitly class-aware, with highly optimized inter-class and intra-class features. Please refer to Chapter 4 for the details.

1.6 Latest Research Status (2022)

Up until now, the attention mechanism and class-aware context aggregation are still the most pupular research topics combined with the Transformer [12] architecture. Different from the traditional FCN [37] based pipelines that use Transformer as the backbone for semantic segmentation [48], many works [44] have also extended OCR following the decoder design in DETR [2] that has improved the class representations (class center) by multiple iterations.

Furthermore, by using the decoder concept in DETR [2] and the mask concept in MaX-DeepLab [51], MaskFormer [8] used the mask classification concept to replace the traditional pixel classification and has achieved notable improvement on semantic segmentation.

However, the issues of DETR [2] also appeared in its derived methods, such as the slow convergence issue in MaskFormer [8]. Although the masked attention in Mask2Former [7] has greatly improved its convergence speed, considering the training iteration and computational cost required, it is still much more inefficient compared with the previous FCN and CNN based methods.

While concluding this thesis, we noticed that our proposed CAA [22](in Chapter 3 3) and CAR [21](in Chapter 4 4) can contribute a more efficient Transformer architecture with similar computational cost and converge speed as the prior FCN and CNN based methods. We will leave this as a future work.

1.7 Research Problems and Contributions

- **Scale-invariant:** As described in Subsection 1.3, solutions addressing issues with fixed-range multi-scale context, such as ASPP, used different dilated convolution operatoins to aggregation multi-range context, which affects to the performance due to the variant features of intra-class pixels. To address this, the thesis proposes “Kernel sharing Atrous Convolution(KSAC)” (see Chapter 2 for details). The detailed contributions are as follows.

  – We propose a sharing strategy to aggregate multi-ranges context with multiple branches but retain scale-invariant characteristic.

  – Based on the sharing strategy, the computational cost of multiple branches is significantly reduced compared to ASPP.

  – The proposed KSAC has achieved top 10 accuracies and appeared in the top 10 algorithms in Pascal VOC 2012 testing set.

- **Seamless dual attention:** As described in Subsection 1.4, dual-attention [14] preformed channel attention and spatial attention separately to model the inter-dependence in the channel and spatial domains, so it has caused feature conflicts because two separated attentions focusing on different aspects in different domains are simply added together. To tackle this, this thesis proposes "Channelized Axial-Attention" (See Chapter 3 for details) to perform the attention in spatial and channel domains seamlessly. Our detailed contributions are as follows.

  - We are the first to explicitly *identify* the potential conflicts between spatial and channel attention in existing dual attention designs by *visualizing* the effects of each attention on the final result.
  - We propose a novel Channelized Axial Attention, which breaks down the axial attention into more basic parts and inserts channel attention in between, integrating spatial attention and channel attention together seamlessly and efficiently, with only a minor computation overhead compared to the original axial attention.
  - To balance the computation speed and GPU memory usage, a grouped vectorization approach for computing the channelized attentions is proposed. This is particularly advantageous when processing large images.
  - Experiments on three challenging benchmark datasets, including PASCAL Context [13], COCO-Stuff [1] and Cityscapes [38], demonstrate the superiority of our approach over the state-of-the-art approaches.

- **Class-aware Regularization:** As described in Subsection 1.5, existing class-center based approaches extracted class centers in various different ways but the centres were simply concatenated with the original features so they lack efficiency. Moreover, the extracted class centers rely on the coarse prediction outcome, which may contain errors. We propose “Class-aware regularization” (See Chapter 4 for details) to explicitly improve the robustness of existing networks, where the class center is computed by the real ground truth during training. Our detailed contributions in this chapter are as follows.

– We propose a universal class-aware regularization module that can be integrated into various segmentation models to largely improve the accuracy.

– We devise three novel regularization terms to achieve more separable and less class-dependent feature representations by minimizing the intra-class variance and maximizing the inter-class distance.

– We calculate the class centers directly from ground truth during training, thus avoiding the error accumulation issue of the existing methods and introducing no computational overhead during inference.

– We provide image-level feature-similarity heatmaps to visualize the learned inter-class features with our CAR are indeed less related to each other.

Chapter 2

Kernel-Sharing Atrous Convolution

As a classical computer vision application, semantic segmentation assigns pixels belonging to the same object class with the same label. During the segmentation procedure, deep networks are required to handle both local detailed and global semantic information, so as to handle objects of arbitrary sizes. To achieve such robustness, numerous efforts have been made by the research community. For example, the Fully Convolutional Network (FCN) [37] and U-Net [45] combined the low-resolution feature maps with the high-resolution ones via concatenation or an element-wise addition operation to extract both detailed and contextual features, and the PSPNet [69], on the other hand, utilized multiple pooling layers in parallel to extract richer information. In particular, in the DeepLab family [3, 4, 5], a more powerful and successful Atrous Spatial Pyramid Pooling (ASPP) structure was proposed to exploit different receptive fields via multiple parallel convolutional branches of different atrous rates (*i.e.*, the dilation size of the convolution kernel; see [3] for details) to extract features for both small and large objects. The ASPP structure improves the networks' generalizability significantly. Thanking the superiority of this parallel concatenation strategy, ASPP has been widely used and further improved by other works, such as DenseASPP net [58].

However, though ASPP and other similar parallel strategies have improved, to some extent, the robustness of their models to objects' scale variability, they still suffer from other limitations. First, the lack of communication among branches compromises the generalizability of individual kernels, as illustrated in Fig. 2.1.

Specifically, kernels in the convolutional branches with small atrous rates or high-resolution feature maps are able to learn detailed information and handle small semantic classes well. However, for large semantic classes, these kernels are incapable of learning features that concern a broader range of context. In contrast, kernels in branches with big atrous rates or low-resolution feature maps are able to extract features with large receptive fields, but may miss much detailed information. Therefore, the generalizability of kernels is limited. On the other hand, the number of samples contributing to training individual branches are reduced since small (or big) objects are only effective for the training of branches with small (or big) atrous rates. Therefore, the representations of similar pixels at different ranges will have different effect on the center pixel's representations. Secondly, it is intuitive that by using parallel branches with separate kernels, the computational cost increases with the increase of the number of parallel branches.

To tackle the above mentioned problems, in this chapter, we propose a novel network structure, namely Kernel-Sharing Atrous Convolution (KSAC), as shown in Fig. 2.2, where multiple branches with different atrous rates can share a single kernel effectively. With this sharing strategy, the shared kernel is able to scan the input feature maps more than once with both small and large receptive fields, and thus be able to *see* both local detailed and global contextual information extracted from objects of small or big sizes. In other words, the information learned with different atrous rates is also shared. Moreover, since objects of various sizes can all contribute to the training of the shared kernel, the number of effective training samples increases, resulting in the improved representation ability of the shared kernel. On the other hand, the computational cost is significantly reduced with the sharing mechanism, and the implementation of the our KSAC is very simple.

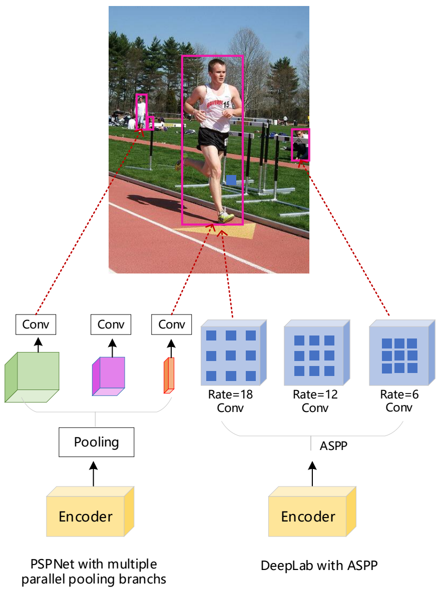


Figure 2.1 : The multi-branch-like solutions used in PSPNet and DeepLab for improving models' robustness to objects' scale variability.

2.1 Related Work and Issues

2.1.1 Fully Convolutional Network

The Fully Convolutional Network (FCN) proposed in [37] was a watershed in the development of semantic segmentation techniques. It was the first publication that successfully applied deep neural networks to spatially dense prediction tasks. The fully-connected layers in deep networks require fixed-size inputs, which conflict with the arbitrary-size inputs of semantic segmentation tasks. FCN solves this problem by transforming the fully connected layers into convolutional layers, allowing networks to produce arbitrary sized heatmaps. In addition, FCN uses the skip connections to fuse global semantic information with local appearance information so that more accurate predictions can be produced. Based on the reported results on the benchmark dataset VOC 2012, FCN has made a major breakthrough for

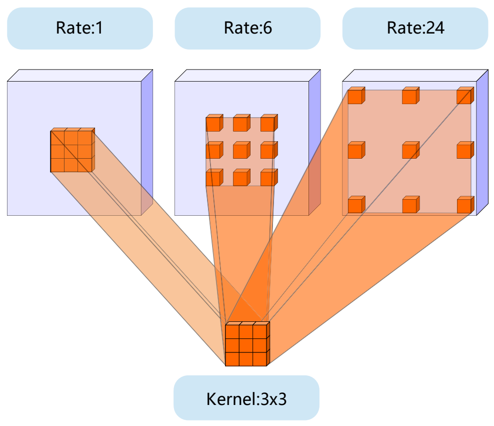


Figure 2.2 : Illustration of our proposed Kernel-Sharing Atrous Convolution structure. The single 3×3 kernel is shared by three parallel branches with different atrous rates.

the problem of semantic segmentation, and outperformed state-of-the-art methods dramatically.

Thus, since the introduction of FCN, all of the subsequent deep networks designed for semantic segmentation have followed the fully convolutional approach. An example is the most widely used medical image segmentation network U-Net [45], where concatenation is used to combine low-level features with high-level features in the skip operation, instead of element-wise addition used in FCN.

2.1.2 DeepLab Family

Models from the DeepLab family [3, 4, 5] have championed the semantic segmentation solutions, thanking the advanced network architecture as well as the huge training datasets used.

In the early version of DeepLab [3], atrous convolution (*aka,* 'dilated convolution') was proposed to expand network's receptive fields without shrinking the feature maps' resolutions, and this was achieved by inserting zeros into the kernels. Additionally, they also employed the fully connected Conditional Random Fields (CRFs) to obtain more accurate boundary predictions and designed a key technique, ASPP (Atrous Spatial Pyramid Pooling), which exploited multiple parallel branches with different atrous rates to generate multi-scale feature maps to handle scale variability. This technique has been retained in all of the subsequent DeepLab versions due to its extraordinary performance. In particular, DeepLabv3 [4] augmented ASPP with image-level features by encoding global context to further boost the segmentation performance. Moreover, DeepLabv3+ [5] embedded the ASPP to a more efficient encoder-decoder architecture, *i.e.,* Xception, and achieved the best performance in the semantic segmentation task. Besides, the authors of DeepLab explored more efficient convolution operators like depthwise separable convolution in MobileNet [46] and more effective network structures via the Neural Architecture Search (NAS) techniques in [33]

However, though ASPP has achieved remarkable performance boost, it still has the limitation because it cannot capture well the representations of similar pixels at different ranges from the center pixels, as explained earlier. Therefore, in this work, we propose a novel Kernel-Sharing Atrous Convolution to handle the scale variability problem more effectively. According to the comprehensive experiments conducted on three benchmark datasets, including PASCAL VOC 2012 [13], ADE20K [71] and Cityscapes [38], our proposed KSAC achieves much better performance than ASPP with also reduced computational cost (see the detailed figures in the FLOPs column of Table 1 for details).

2.1.3 Other Semantic Segmentation Models

In addition to the aforementioned models, there are many other outstanding deep networks designed for semantic segmentation. For instance, the PSPNet proposed in [69] aggregated the global context information via a pyramid pooling module, together with their proposed pyramid scene parsing network. DenseASPP [58] argued that the scale-axis of ASPP was not dense enough for the autonomous driving scenario, so they designed a more powerful DenseASPP structure, where a group of atrous convolutional layers were connected in a densely connected way. Considering the importance of global contextual information, a Context Encoding Module was proposed in [65] to capture the semantic context of scenes and enhance the class-dependent feature maps. This method improved the segmentation results with only a slightly increased computational cost when compared with the FCN structure [37].

In recent years, self-attention [53] and its variants [67] and [14] have been widely reported in various semantic segmentation works. With the self-attention mechanism, the similarities between each pair of pixels are used to optimize the representations of each pixel based on the entire feature map, breaking the locally finite receptive fields in the conventional convolution operation. The works reported in [23, 72, 19, 61, 15] show that performing attention on partial feature maps for each pixel can achieve comparable performance of the full attention but at a much reduced computational cost. The computation of self-attention breaks the limitation of 2D feature maps, but also brings some information loss, such as pixel locations, which may be one of the reasons that skip connections and auxiliary loss are widely used in self-attention based models [67, 27].

Clearly, improving the correctness of the representations for pixels belonging to objects of arbitrary sizes has been an intrinsic goal for recent semantic segmentation techniques. Existing attempts have explored various possibilities to take into consideration both global context and local appearance information. In this work, we propose an effective sharing strategy, *i.e.,* Kernel-Sharing Atrous Convolution (KSAC). Our experimental results demonstrate the superiority of this idea in terms of improving the segmentation quality, reducing the network complexity and considering a wider range of context. Next, the technical details of our proposed KSAC, together with our motivations and justification, are presented.

2.2 Proposed Solution

As introduced above, thanking the development of techniques including atrous convolution, depthwise separable convolution, ASPP and Xception, etc, the DeepLab [5] has achieved the highest performance for the task of semantic segmentation and become the most significant and successful multi-branch structures. In our work, for fair comparison with the well-known ASPP structure, we base our proposed KSAC on the DeepLab framework and replace the ASPP module with our KSAC. More details are presented below.

2.2.1 Atrous Spatial Pyramid Pooling

The receptive field of a filter represents the range of context that can be viewed when calculating features as input for the subsequent layers. A large receptive field enables the network to consider wider range context and more semantic information, which is vital to handling large sized objects. In contrast, a small receptive field is good for capturing local detailed information, which can help to generate more refined boundaries and more accurate predictions, especially for small objects. However, the receptive fields are fixed in traditional convolution operators (*e.g.,* a 3×3 kernel has a fixed receptive field of 3×3). Atrous convolution allows us to expand the receptive fields of filters flexibly by setting various atrous rates for the traditional convolutional layer and inserting zeros into the filters accordingly.

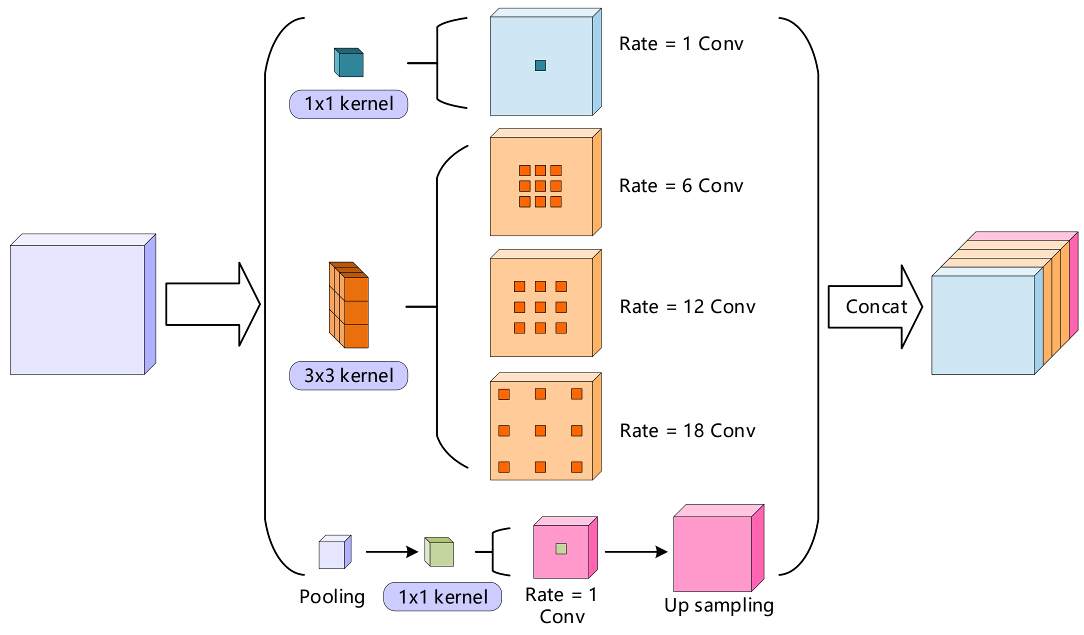


Figure 2.3 : The detailed architecture of our proposed Kernel-Sharing Atrous Convolution with $rate = (6, 12, 18)$

Furthermore, in the ASPP structure [3], to handle objects with arbitrary sizes, multiple atrous convolution layers with different atrous rates were used in parallel, and their outputs were combined to integrate information extracted with various receptive fields. However, as analyzed above, this design does harm to the generalizability of kernels in individual branches and also increases the computation burden. To address this issue, we propose a novel sharing mechanism (*i.e.*, KSAC) to improve the semantic segmentation performance of the existing models.

2.2.2 Atrous Convolution with Shared Kernel

As shown in Fig. 2.3, our proposed KSAC is composed of three components, *i.e.*, a 1×1 convolutional layer, a global average pooling layer followed by a 1×1 convolutional layer to obtain the image-level features, and a pyramid atrous convolutional module with a shared 3×3 kernel and atrous rates $(6, 12, 18)$. Note that the batch

Algorithm 1 Kernel-Sharing Atrous Convolution

```
Require: $I$: Input channels, $T$: Input feature maps, $C$: Output channels, $R$: Atrous rates
1: $shape \leftarrow [3, 3, I, C]$
2: $K \leftarrow$ KERNEL($shape$)                ▷ generate shared kernel
3: for $r \in R$ do
4:     $T'_r \leftarrow$CONV2D($T, r, K$)
5:     $B_r \leftarrow$ BATCHNORM($T'_r$)
6: end for
Ensure: $\sum_{r \in R} ReLU(B_r)$
```

normalization layers are used after each convolutional layer. Algorithm 1 shows the implementation details of our KSAC.

As demonstrated in our experiments, our proposed sharing strategy not only helps reduce computational cost but also improves the segmentation performance. This improvement can be explained from two aspects. Firstly, the generalization ability of the shared kernels are enhanced by learning both local detailed features for small objects and global semantically rich features for large objects, which is realized via varying the atrous rates. Secondly, the number of effective training samples is increased by sharing information, which improves the representation ability of the shared kernels. As described in Fig. 2.1, kernels with small atrous rates in ASPP cannot extract features comprehensively enough for large objects, while those with large atrous rates are ineffective on extracting local and fine details for small objects. Therefore, kernels in individual branches can only be trained effectively by some objects in the training images. In contrast, in our proposed KSAC, all of the objects in the training images are contributive samples for the training of the shared kernel. Note that, essentially, this kernel-sharing's purpose is to conduct

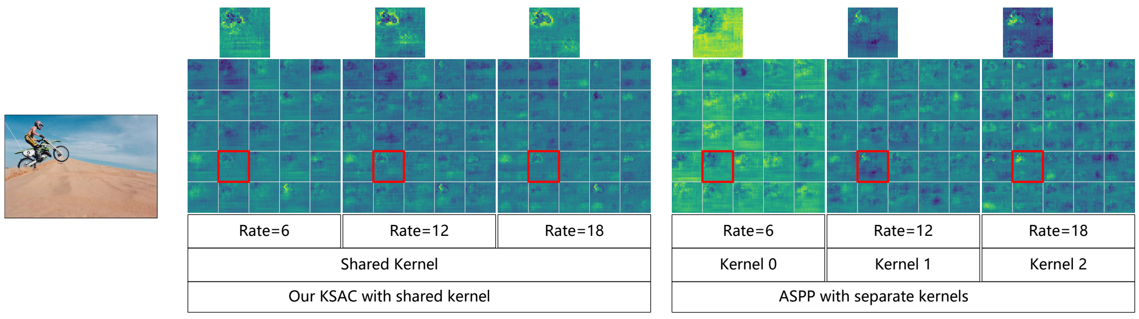


Figure 2.4 : Visualization of the feature maps extracted by kernels of KSAC and ASPP. Here, 25 feature maps are presented for each rate, and we enlarge the ones indicated by red bounding boxes on the top of the figure. Apparently, edges and contours extracted by the shared kernel of our KSAC are much clearer than those extracted by separate kernels of ASPP, for both large atrous rate and small atrous rate. Readers are suggested to zoom in to see more details.

'feature' augmentation inside the network by sharing kernels among branches. Like data augmentation performed in the pre-processing stage, feature augmentation performed inside the network can help to enhance the representation ability of the shared kernels.

To better understand the enhanced generalization and representation abilities of our KSAC, we visualize the feature maps learned by its shared kernel, and compare these feature maps with those generated by ASPP's separate kernels, as shown in Fig. 2.4. Obviously, no matter whether it is for branches with small atrous rates (or small receptive fields) or for large atrous rates (large receptive fields), the feature maps produced by our KSAC are much more comprehensive, expressive and discriminative than those generated by ASPP. Specifically, as illustrated in Fig. 2.4, the edges (local detailed information) and contours (global semantic) detected by our KSAC are much clearer than those detected by ASPP.

Moreover, as pointed out in [5], the DeepLab model has achieved the best performance under the setting $rate = (6, 12, 18)$ in ASPP. However, when an additional parallel branch with $rate = 24$ was added, the performance actually dropped slightly by 0.12%. That is to say, ASPP is not able to produce better performance through capturing a longer range of context. In contrast, according to our experimental results, the performance obtained with our proposed KSAC can be further improved with the setting $rate = (1, 6, 12, 18, 24)$. This demonstrates that, compared with ASPP, our proposed KSAC is more effective in terms of capturing longer ranges of context with larger atrous rates and wider parallel atrous convolutional branches.

2.3 Experiment Setting

To demonstrate the effectiveness of our proposed KSAC sharing mechanism, we evaluated its performance on three benchmark datasets, *i.e.*, PASCAL VOC 2012 [13], ADE20K [71] and Cityscapes [38], and compared its performance with those of the state-of-the-art approaches, in terms segmentation accuracy, model size, robustness as well as efficiency and GPU memory usage. In this section, details of involved datasets, model implementation and training protocol are presented.

2.3.1 Datasets and Data Augmentation

In this work, the same as in [4, 27], we use the benchmark datasets Semantic Boundaries Dataset (SBD) and COCO for pre-training, and PASCAL VOC 2012 for fine-tuning and evaluation. We also conduct experiments on ADE20K and Cityscapes, where no segmentation dataset is used for pre-training.

2.3.1.1 PASCAL VOC 2012

Pascal VOC 2012 is created for multiple purposes, including detection, recognition and segmentation, etc. There are a large number of images provided in this dataset, but only about 4,500 of them are labeled with high quality for segmentation. In particular, the PASCAL VOC 2012 segmentation dataset consists of about

1,500 annotated training images, 1,500 annotated validation images and 1,500 unannotated test images.

2.3.1.2 SBD

SBD [17] is a third party extension of the PASCAL VOC 2012 dataset and composed of about 8,500 annotated training images and 2,800 annotated validation images. Among the released images, more than 1,000 of them are picked directly from the official PASCAL VOC 2012 validation set. Therefore, in order to use the SBD dataset for the training and accurately evaluate the performance of related models with the PASCAL VOC 2012 validation set, we remove these images from the SBD and merge the rest of the training and validation images to create the SBD 'trainaug' dataset.

2.3.1.3 COCO

COCO [1] is a huge dataset created for multiple tasks. As mentioned in the literature, additional improvement can be made if the model is pre-trained with the COCO dataset. Therefore, following the practice in [5], we select about 60K training images from the COCO dataset to include images containing classes defined in PASCAL VOC 2012 and with an annotation region greater than 1,000. Moreover, any classes that are not defined in PASCAL VOC 2012 are treated as background.

2.3.1.4 ADE20K

ADE20K [71] is a large-scale scene parsing dataset containing 150 stuff, *i.e.,* object category labels. This dataset is split into 20k, 2k and 3k for training, validation and testing, respectively.

2.3.1.5 Cityscapes

Cityscapes [38] has 19 classes. Its fine set contains high quality pixel-level annotations of 5,000 images, where there are 2,975, 500 and 1525 images in the Training, Validation, and Test sets, respectively. The same as in other works [26, 14], we crop the training images to 769×769 during training, and keep full-resolution 1025×2049 during inference. We report our results on the Test set.

To fairly compare our proposed model with other existing works, we also apply some widely adopted data augmentation strategies in training, including horizontally flipping with 50% probability, randomly scaling the images with a scaling factor between 0.5 and 2.0 and at a step size of 0.25, padding and randomly cropping the scaled images to a size of 513×513.

2.3.2 Implementation Details

In our experiments, the batch size is set to 32 and 16 for the MobileNetV2-based models and Xception-based models, respectively. Additionally, our networks are optimized with the SGD optimizer, and in the first pre-training stage, they are trained on the mixed dataset of COCO, SBD and VOC for 300K iterations with a learning rate of 1e-3. Then, the learning rate is adjusted to 4e-4 and related networks are continually trained on the SBD and VOC mixed datasets for another 40K iterations. Finally, our models are fine-tuned on the VOC training set with a learning rate of 2e-4. For the evaluation on ADE20K, backbones pre-trained on ImageNet are loaded and no semantic segmentation dataset is used for pre-training. In particular, our networks are trained only on the ADE20K training set with an initial learning rate of 2e-3 and a momentum of 0.9. Then, according to the segmentation loss, the learning rate is manually reduced in half about every 30K iterations.

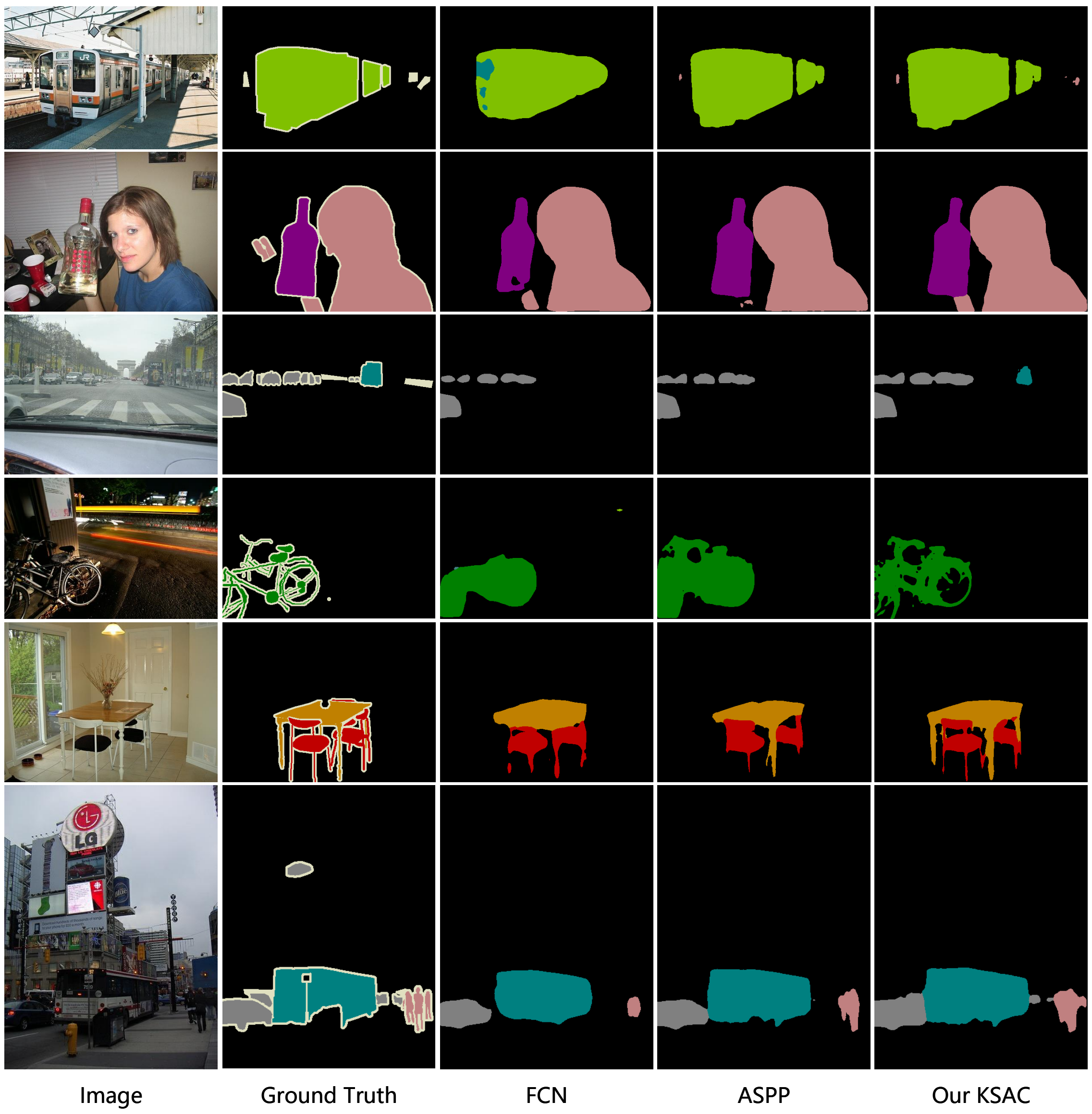


Figure 2.5 : Comparison of the segmentation results obtained by FCN, ASPP and our KSAC on the Pascal VOC 2012 validation set.

Head		Backbone	D	OS	COCO	MF	mIOU (%)	FLOPs
ASPP	**Ours**							
		ResNet-50 [18]		16			70.99	39.94G
✓		ResNet-50 [18]		16			74.91	57.37G
	✓	ResNet-50 [18]		16			76.23	54.63G
✓		ResNet-101 [18]		16			77.20	76.61G
✓		ResNet-101 [18]		8			78.51	257.74G
✓		ResNet-101 [18]		8		✓	79.77	-
✓		ResNet-101 [18]	✓	8			79.35	277.99G
	✓	ResNet-101 [18]		16			77.77	73.87G
	✓	ResNet-101 [18]		16		✓	79.63	-
	✓	ResNet-101 [18]		8			78.78	245.09G
	✓	ResNet-101 [18]		8		✓	80.10	-
	✓	ResNet-101 [18]	✓	8			79.70	265.34G
✓		Xception65 [10]	✓	16			79.79	83.50G
✓		Xception65 [10]	✓	16	✓		82.20	83.50G
✓		Xception65 [10]	✓	16	✓	✓	83.34	-
✓		Xception65 [10]	✓	8	✓		82.45	244.54G
✓		Xception65 [10]	✓	8	✓	✓	83.58	-
	✓	Xception65 [10]	✓	16			80.22	80.77G
	✓	Xception65 [10]	✓	16	✓		83.92	80.77G
	✓	Xception65 [10]	✓	16	✓	✓	**85.96**	-
	✓	Xception65 [10]	✓	8	✓		84.13	231.89G
	✓	Xception65 [10]	✓	8	✓	✓	**86.09**	-
✓		MobileNetV2 [46]		16	✓		75.70	-
	✓	MobileNetV2 [46]		16	✓		76.30	-

Table 2.1 : Experimental results obtained on PASCAL VOC 2012 validation set with different inference strategies when using ASPP and our proposed KSAC, and ResNet-50, ResNet-101, Xception65 or MobileNetV2 as the backbone. **KSAC:** Using our proposed KSAC. **ASPP:** Using the standard ASPP structure proposed in DeepLabv3+ [4]. **D:** Concatenating the OS = 4 feature maps from the backbone during the upsampling of the logits. **MF:** Employing multi-scale (MS) and left-right flipping on the inputs during the evaluation. **COCO:** Model is pre-trained on COCO dataset [1]. The Performance is evaluated from the aspect of **mIOU (%)** and the number of FLOPS (Floating Point Operations per Second), respectively.

2.4 Experiment Results

2.4.1 Improved mIOU

To demonstrate the effectiveness of our proposed KSAC, we first compare it with ASPP, the most successful multi-scale structure that has played a key role in the DeepLab family. The comparison results are shown in Table 2.1. Moreover, additional comparative visualizations are presented in Fig. 2.5, showing our results and the failure segmentation predictions generated by the baseline models. Note that the combination of ASPP, Xception and Decoder has exactly the same architecture as that DeepLabv3+ [5]. In addition, the ASPP module and our KSAC module used in Table 2.1 are with the same atrous rate setting, *i.e.*, (6, 12, 18), which is the standard setting of DeepLabv3+ [5].

As we can see from Table 2.1, under the same configuration, by replacing ASPP with our proposed KSAC, the mIOU figures have been improved for all the models with ResNets, Xception and MobileNetV2 backbones. In particular, when the Xception backbone is used, our proposed KSAC has achieved the highest mIOU (85.96% for OS=16 and 86.09% for OS=8), which are 2.62% and 2.51% higher than DeepLabv3+ (83.34% when OS=16 and 83.58% when OS = 8).

To further illustrate the superiority of our proposed KSAC, we also compare its performance with those of other state-of-the-art approaches. As listed in Table 2.3, our KSAC outperforms all of the listed methods on both the validation set and test set of PASCAL VOC 2012. Note that, for a fair comparison, we only compare our KSAC with methods using ResNet-101, ResNerXt-131 or Xception-65 as their backbones.

From the above comparison results, we can conclude that our proposed KSAC structure is more robust and effective than the ASPP structure, and by seeing the input feature maps multiple times with different receptive fields, the networks' generalization and representation abilities have been significantly improved.

2.4.2 Reduced Computational Cost

Reducing computational cost is another advantage of our KSAC. Specifically, the typical ASPP with dilation rates = (6, 12, 18) requires three 3×3 convolutions, so that all pixels perform $3 \times 3 \times 3$ dot product operations. In KSAC, thanking the weight sharing, only $1 \times 3 \times 3$ dot product operations are needed.

Table 2.1 presents the computational cost for each segmentation head. As it can be seen, with our proposed sharing mechanism, the FLOPs count of our KSAC is significantly smaller than that of the ASPP. In particular, when inferencing with an output stride = 16, about 2.73G FLOPs is saved, and when inferencing with an output stride = 8, about 12.65G FLOPs is saved. Note that, if we leave out the major FLOPs contributed by the computations independent to ASPP or KSAC module such as the backbone, the advantages of our KSAC over ASPP is more significant, with a 15% save on computational cost.

2.4.3 Capability of Handling Wider Range of Context

As claimed in [4], the DeepLabv3 model achieved the best performance when three parallel branches with atrous rates of of 6, 12, and 18 being used in the ASPP module, while an additional parallel branch with $rate = 24$ resulted in a slight drop (0.12%) of the performance. In contrast, our proposed KSAC is able to take the benefit of a wider range of context to further improve the segmentation performance. As shown in Table 2.2, when added with two atrous convolution branches with rates 1 and 24 in our KSAC structure, the mIOU is further improved from 85.96% to 87.01%. Specifically, since the newly added branches share the same kernel with the original three branches, no additional parameters are added.

Atrous Rates		Testing Strategy		
(6, 12, 18)	(1, 6, 12, 18, 24)	Multi-Scale	Flip	mIOU (%)
✓				83.92
✓		✓	✓	**85.96**
	✓			84.50
	✓	✓	✓	**87.01**

Table 2.2 : Experimental results of our proposed KSAC on PASCAL VOC 2012 validation set with different settings of atrous rates.

	mIOU (%)	
Methods	Validation	Test
PSPNet [69]	-	85.4
EMA (ResNet-101) [27]	-	87.7
ExFuse [68]	85.8	87.9
SANet [70]	-	86.1
SDN [16]	84.8	86.6
CFNet [67]	-	87.2
DeepLabv3 [4]	82.7	85.7
DeepLabv3+ [5]	83.6	87.8
Our KSAC	**87.0**	**88.1**

Table 2.3 : Comparison results with other approaches on the PASCAL VOC 2012 validation and test sets.

2.4.4 Improved Speed with Less GPU Memory Usage

The existing approaches, such as [14, 67], have trained their models and obtained their final results on Pascal VOC 2012 by setting the output stride (OS) to 8. Technically, $OS = 8$ means the output size of the encoder is 1/8 of the original image size, in order to yield more detailed feature maps compared to $OS = 16$, as described in [4]. However, this also means a significant drop on speed and more GPU memory consumption. According to our experiments and the data provided in [5], a setting of $OS = 8$ will result in the speed dropped by about three times and the GPU memory usage increased by nearly four times, compared to $OS = 16$. By contrast, in our KSAC, the sharing kernel mechanism gives rise to stronger generalizability and representation ability, so, with $OS = 16$, we are able to achieve the segmentation results similar to those of ASPP when $OS = 8$.

In Table 2.1, we conducted experiments with $OS = 16$, as well as $OS = 8$ for ResNet-101 and Xception65. The results show that $OS = 8$ can further improve the performance without adding the decoder. It is also worth of noting that, after adding the decoder, the improvement of $OS = 8$ over $OS = 16$ becomes very small (86.09% vs 85.96%).

Therefore, we can say that our KSAC has achieved high performance without dramatic speed loss and extra GPU memory cost.

2.4.5 Experiment Results on ADE20K

To further demonstrate the effectiveness and robustness of our proposed KSAC, we also evaluate its performance on the ADE20K dataset. Finally, our KSAC achieves mIOUs of 43.20% and 45.47% for single scale test and multi-scale test, respectively, outperforming all of the listed approaches as shown in Table 2.4. Note that, because of the hardware limitation, we resize our training images to a size of 513×513, instead of 1000×1000 used in other works, and a smaller size is usually harmful to the segmentation performance.

Methods	mIOU (%)
RefineNet [31]	40.7
UperNet [56]	42.66
PSPNet [69]	43.29
DSSPN [30]	43.68
EncNet [65]	44.65
CFNet [67]	44.89
Our KSAC	**45.47**

Table 2.4 : Comparison results with other approaches on the ADE20K validation set for multi-scale prediction.

2.5 Summary

In this chapter, aiming to address the scale variability problem in semantic segmentation, we have proposed a novel and effective network structure namely Kernel-Sharing Atrous Convolution (KSAC), where different branches share one single kernel with different atrous rates, *i.e.,* let a single kernel see the input feature maps more than once with different receptive fields. Experimental results conducted on the benchmark PASCAL VOC 2012, ADE20K and Cityscapes have demonstrated the superiority of our proposed KSAC. KSAC has not only effectively improved the segmentation performance but also significantly reduced the computational cost with less GPU memory usage. Additionally, compared with the well-known ASPP structure, our KSAC can also capture a wider range of context without downgrading performance. However, our KSAC has a limitation that cannot be ignored. In this work, we have empirically set the dilation rates to capture the multi-scale contextual information based on experiments. However, such fixed rates cannot handle scenarios where the data are significantly different from the training dataset. In this case, setting dynamic dilation rates (*e.g.*, generating dilation rates during run-time based on axial pixel representations) may be helpful. We leave this exploration as a future work.

Chapter 3

Channelized Axial Attention

Most of the existing semantic segmentation approaches [61, 58, 14, 27] have adopted a pipeline similar to the one that is defined by Fully Convolutional Networks (FCNs) [37] using fully convolutional layers to output the pixel-level segmentation results of input images. These approaches have achieved state-of-the-art performance. After the FCNs, there have been many approaches dedicated to extracting enhanced pixel representations from the backbone. Earlier approaches, including PSPNet [69] and DeepLab [5], used a Pyramid Pooling Module or an Atrous Spatial Pyramid Pooling module to expand the receptive field to enhance the representation capabilities. Recently, many works focus on using the attention mechanisms to enhance pixel representations. The first set of attempts in this direction included Squeeze and Excitation Networks (SENets) [20] that introduced a simple yet effective channel attention module to explicitly model the interdependencies between channels. Meanwhile, spatial attention relied on self-attention proposed in [53, 50] to model long-range dependencies in spatial domain, so as to produce more correct pixel representations. For each pixel in the feature maps, spatial attention "corrects" its representation with the representations of other pixels depending on their similarity. In contrast, channel attention identifies important channels based on all spatial locations and reweights the extracted features.

Parallel dual attention (*e.g.*, [14]) was proposed to gain the advantages of both spatial attention and channel attention. This approach directly fused their results with an element-wise addition (see Fig. 3.1(a)). Although they have achieved im-

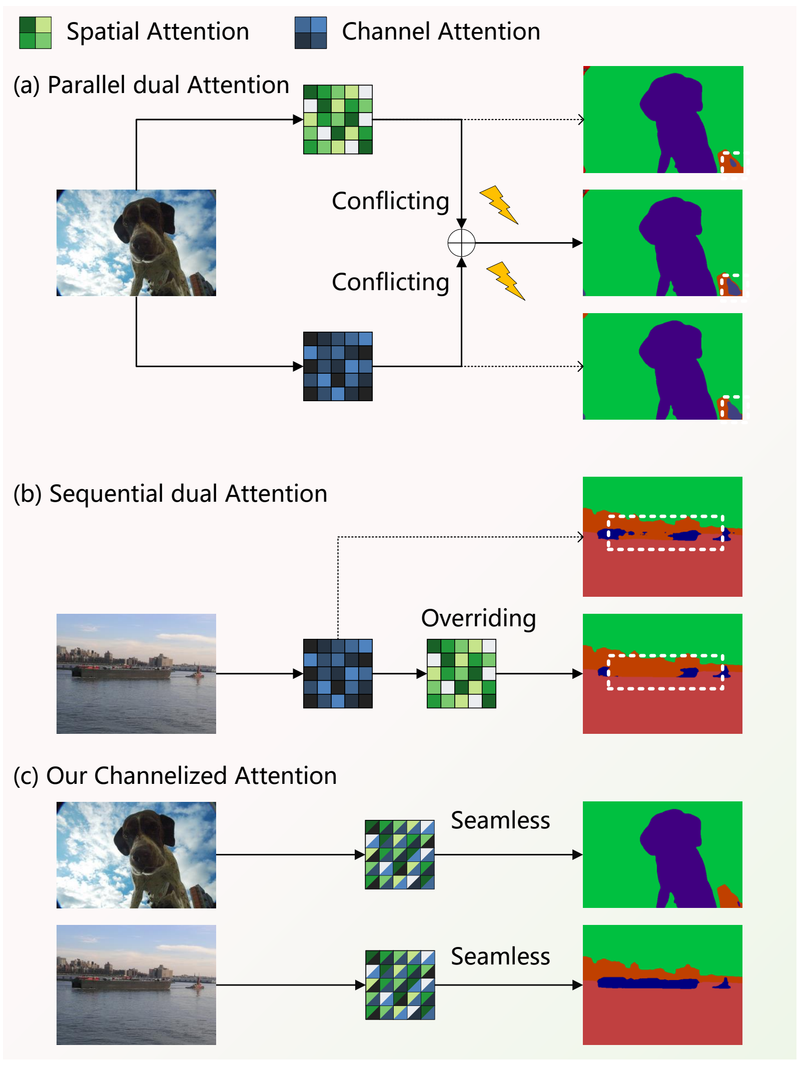

Spatial Attention
Channel Attention
(a) Parallel dual Attention
Conflicting
Conflicting
(b) Sequential dual Attention
Overriding
(c) Our Channelized Attention
Seamless
Seamless

Figure 3.1 : Different dual attention designs: (a) **Parallel dual attention** sums the results from spatial and channel attentions directly, which may cause conflicts because spatial and channel attentions are focusing on different aspects. (b) **Sequential dual attention** performs spatial attention after channel attention, where the spatial attention may override correct features extracted by the channel attention. (c) **Our channelized attention** seamlessly merges the spatial and channel attentions into a single operation (see Sect. 3.3.2), removing the potential conflicting issue caused by **a** or **b**.

proved performance, the relationship between the contributions of spatial and channel attentions to the final results is unclear. Moreover, calculating the two attentions separately not only increases the computational complexity, but may also result in conflicting importance of feature representations. For example, some channels may appear to be important in spatial attention for a pixel that belongs to a partial region in the feature maps. However, channel attention may have its own perspective, which is calculated by summing up the similarities over the entire feature maps, and weakens the impact of spatial attention.

Sequential dual attention, which combines channel attention and spatial attention in a sequential manner (Fig. 3.1(a)) has similar issues. For example, channel attention can *ignore* the partial region representation obtained from the overall perspective. However, this partial region representation may be required by spatial attention. Thus, directly fusing the spatial and channel attention results may yield incorrect importance weights for pixel representations. In Sect. 3.4, we develop an approach to visualize the impact of the conflicting feature representation on the final segmentation results.

In order to overcome the aforementioned issues, we propose Channelized Axial Attention (CAA), which breaks down the axial attention into more basic parts and inserts channel attention into them, combining spatial attention and channel attention together seamlessly and efficiently. Specifically, when applying the axial attention maps to the input signal [53], we capture the intermediate results of the dot product before they are summed up along the corresponding axes. Capturing these intermediate results allows channel attention to be integrated for each column and each row, instead of computing on the mean or sum of the features in the entire feature maps. We also develop a novel grouped vectorization approach to maximize the computation speed in limited GPU memory.

In summary, our contributions in this chapter are as follows.

- We are the first to explicitly *identify* the potential conflicts between spatial and channel attention in existing dual attention designs by *visualizing* the effects of each attention on the final result.
- We propose a novel Channelized Axial Attention, which breaks down the axial attention into more basic parts and inserts channel attention in between, integrating spatial attention and channel attention together seamlessly and efficiently, with only a minor computation overhead compared to the original axial attention.
- To balance the computation speed and GPU memory usage, a grouped vectorization approach for computing the channelized attentions is proposed. This is particularly advantageous when processing large images.
- Experiments on three challenging benchmark datasets, including PASCAL Context [13], COCO-Stuff [1] and Cityscapes [38], demonstrate the superiority of our approach over the state-of-the-art approaches.

3.1 Related Work and issues

Spatial attention. Non-local networks [53] and Transformer [50] introduced the self-attention mechanism to examine the pixel relationship in the spatial domain. It usually calculates dot-product similarity or cosine similarity to obtain the similarity measurement between every two pixels in feature maps, and recalculates the feature representation of each pixel according to its similarity with others. Self-attention has successfully addressed the feature map coverage issue of multiple fixed-range approaches [3, 69], but it has also introduced huge computation costs for computing the complete feature map. This means that, for each pixel in the feature maps, its attention similarity affects all other pixels. Recently, many approaches [23, 72] have been developed to optimize the GPU memory costs of spatial self-attention.

Channel Attention. Channel attention [20] examined the relationships between channels, and enhanced the important channels so as to improve performance. SENets [20] conducted a global average pooling to get mean feature representations, and then went through two fully connected layers, where the first one reduced channels and the second one recovered the original channels, resulting in channel-wise weights according to the importance of channels.

In DANet [14], channel-wise relationships were modelled by a 2D attention matrix, similar to the self-attention used in the spatial domain, except that it computed the attention with a dimension of $C \times C$ rather than $(H \times W) \times (H \times W)$ (here, C represents the number of channels, and H and W represent the height and width of the feature maps, respectively).

Spatial Attention + Channel Attention. Combining spatial attention and channel attention can provide fully optimized pixel representations in a feature map. However, it is not easy to enjoy both advantages seamlessly. In DANet [14], the results of the channel attention and spatial attention are directly added together. Supposing that there is a pixel belonging to a semantic class that has a tiny region in the feature maps, spatial-attention can find its similar pixels.

However, channel representation of the semantic class with a partial region of the feature maps may not be important in the perspective of entire feature maps, so it may be ignored when conducting channel attention computations. Computing self-attention and channel attention separately (as illustrated in Fig. 3.1(a)) can cause conflicting results, and thus weaken their performance when both results are summarized together. Similarly, in the cascaded model (see Fig. 3.1(b)), the spatial attention module after the channel attention module may pick up an incorrect pixel representation enhanced by channel attention, because channel attention computes channel importance according to the entire feature maps.

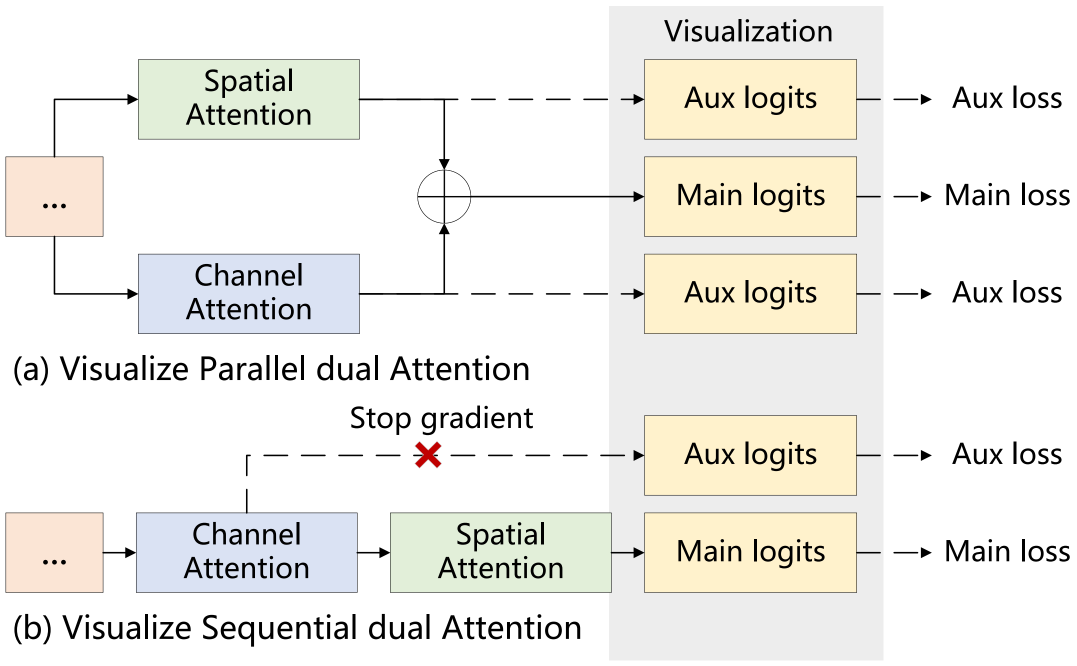


Figure 3.2 : Our designs for visualizing the effects of dual attentions in parallel and sequential.

3.2 Exploring Conflicting Features

As we have analyzed earlier in Sect. 3.1, computing spatial and channel attentions separately can cause conflicting features. In our experiments, to illustrate this feature conflicting issue faced by existing dual attention approaches, we designed a simple way to visualize the effects of spatial attention and channel attentions on pixel representation.

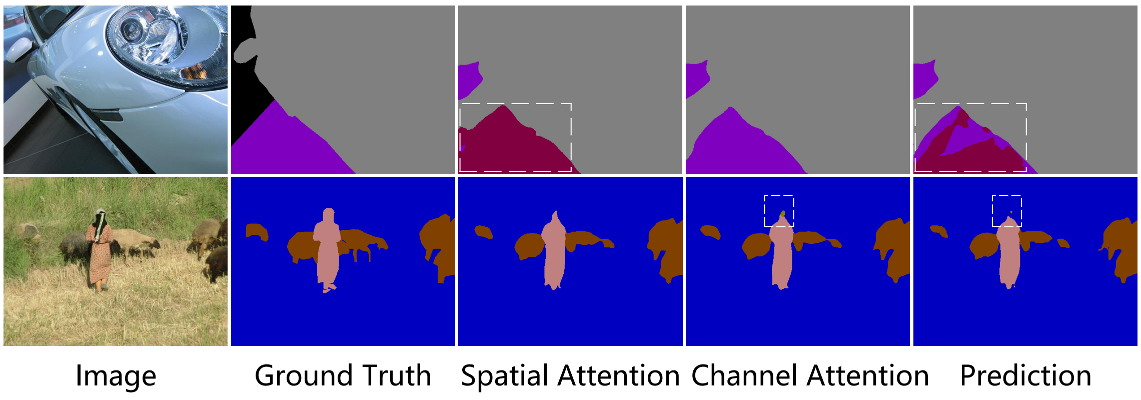


Figure 3.3 : Conflicting features in parallel dual attention designs. **Top:** The bad spatial attention representation negatively influences the good channel attention representation. **Bottom:** The bad channel attention representation negatively influences the good spatial attention representation. See the boxed areas.

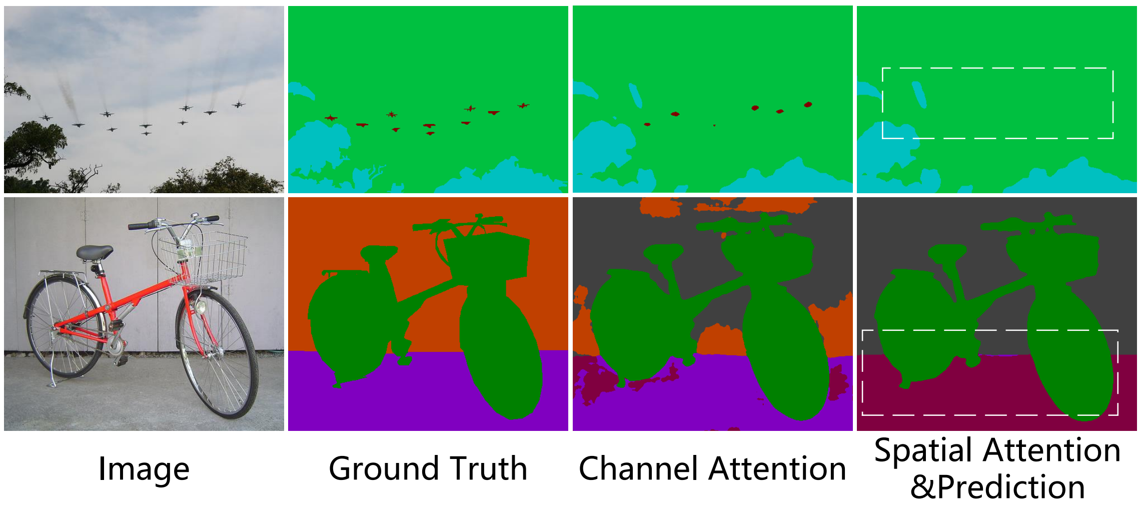


Figure 3.4 : In sequential dual attention designs, the spatial attention representation (the 4th column) ignores the correct channel attention representation (the 3rd column).

3.2.1 Visualizing Conflicts

For a parallel dual attention design such as DANet [14], since it has two auxiliary losses for each of spatial attention and channel attention, we directly use their logits during inference to generate corresponding segmentation results and compare them with the result generated by the main logits. For a sequential dual attention design, we add an extra branch that directly uses the pixel representation obtained from channel attention to perform the segmentation logits. Note that, since the original sequential design does not have independent logits after the channel attention module, we stop the gradient from back-propagating to the main branch, to ensure that our newly added branch has no effect on the main branch.

3.2.2 Examples of Conflicting Features

To visualize the impact of the feature conflicting issue in the existing dual attention designs (see Sect. 3.1), we present examples of the segmentation results obtained with the conflicting features in the parallel dual attention design (see Fig. 3.3) and the sequential dual attention design (see Fig. 3.4). As observed from Fig. 3.3, the parallel design of dual attention directly sums up the pixel representations obtained from spatial attention and channel attention. With this approach, the advantages of the pixel representations obtained from one can be weakened by the other.

The sequential way of combining the dual attentions avoids taking their average but still has its own problem. As shown in Fig. 3.4, the pixel representation obtained from the spatial attention ignores the correct representation obtained from the channel attention, and worsens the prediction.

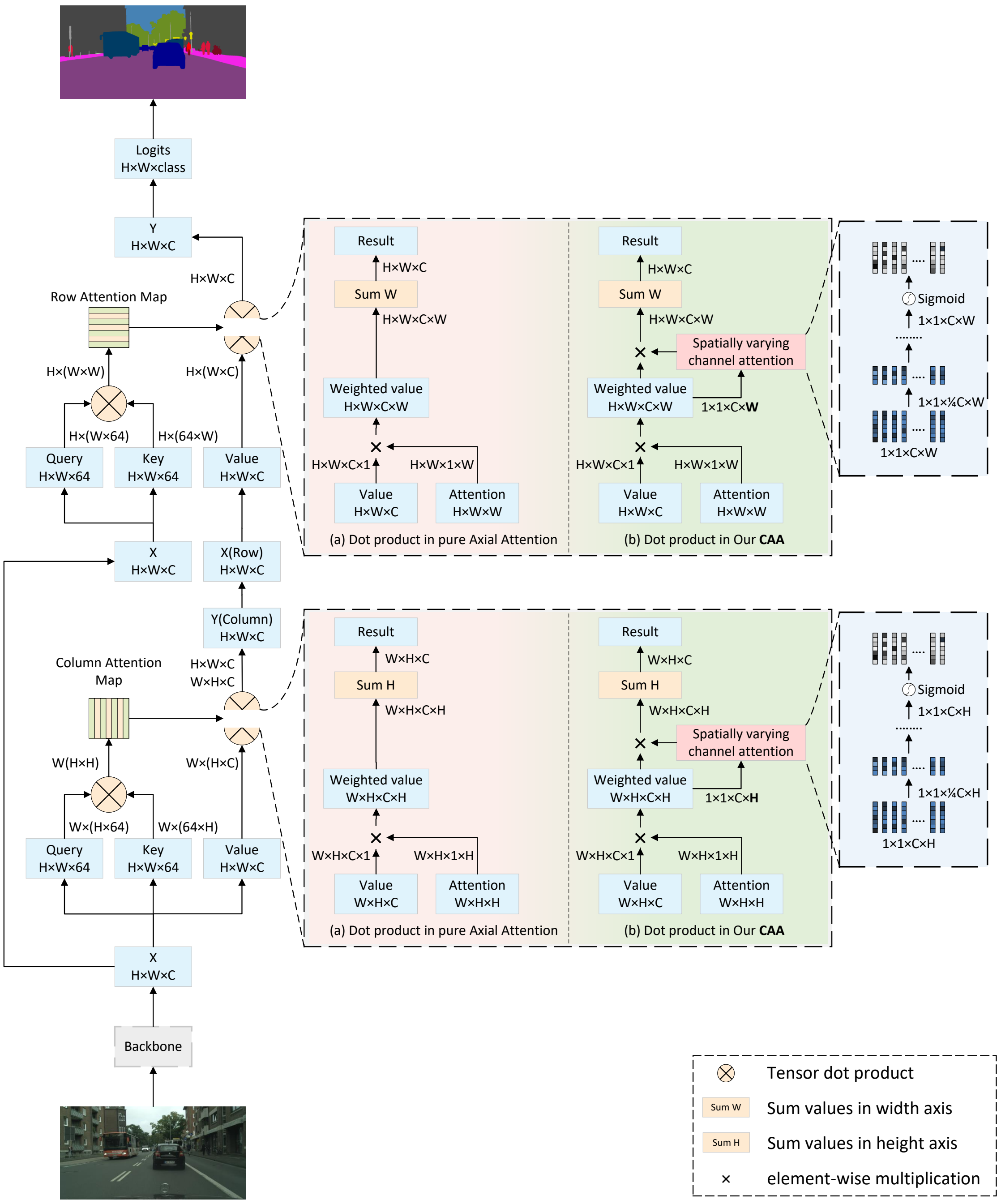


Figure 3.5 : The detailed architecture of the proposed CAA. We present the way to apply channel attention seamlessly in (**b**). We mark the independent spatial dimension in the **bold** style. This allows channel attention to also consider spatially unique information. *Note that*, in our design, the "*value*" for Row attention is obtained from the result of Column attention. See Eq. 3.11 for details.

3.3 Methods

3.3.1 Preliminaries

3.3.1.1 Formulation of the Spatial Self-attention

Following Non Local [53] and Stand Alone Self Attention [42], a 2D self-attention operation in spatial domain can be defined by:

$$\mathbf{y}_{i,j} = \sum_{\forall m,n} f(\mathbf{x}_{i,j}, \mathbf{x}_{m,n}) g(\mathbf{x}_{m,n}). \tag{3.1}$$

Here, a pairwise function f computes the similarity between the pixel representation $\mathbf{x}_{i,j}$ (*query*) at position (i, j) and the pixel representation $\mathbf{x}_{m,n}$ (*key*) at all other possible positions (m, n). The unary function g maps the original representation at position (m, n) to a new domain (*value*). In our work, we use the similarity function [53] as f, *i.e.*,

$$f(\mathbf{x}_{i,j}, \mathbf{x}_{m,n}) = \text{softmax}_{m,n}(\theta(\mathbf{x}_{i,j})^T \theta(\mathbf{x}_{m,n})), \tag{3.2}$$

where θ is a 1×1 convolution layer transforming the feature maps $\mathbf{x}$ to a new domain to calculate dot-product similarity [53] between every two pixels. Note that, following a common practice [26], we use the same 1×1 convolution weights for both query and key. Then, these similarities are used as the weights (Eq. (3.1)) to aggregate features of all pixels, producing an enhanced pixel representation $\mathbf{y}_{i,j}$ at position (i, j).

3.3.1.2 Formulation of the Axial Attention

From the above equations, we can see the computational complexity of the self-attention module is $O(H^2W^2)$, which requires large computation resources and prevents real-time applications, such as autopilot. Several subsequent works [23, 19] focused on reducing the computational complexity while maintaining high accuracy.

In this work, we adopt axial-attention to perform spatial attention. In axial attention, the attention map is calculated for the column and row that cover the current pixel, reducing the computational complexity to $O(HW^2 + H^2W)$.

For convenience, we call the attention values calculated along the Y axis "*column attention*", and the attention values calculated along the X axis "*row attention*". Similar to Eq. 3.2, we define axial similarity functions by:

$$A_{\text{col}}(\mathbf{x}_{i,j}, \mathbf{x}_{m,j}) = \text{softmax}_m \left(\theta(\mathbf{x}_{i,j})^T \theta(\mathbf{x}_{m,j})\right) \ , \ m \in [H]^*, \tag{3.3}$$

and

$$A_{\text{row}}(\mathbf{x}_{i,j}, \mathbf{x}_{i,n}) = \text{softmax}_n \left(\phi(\mathbf{x}_{i,j})^T \phi(\mathbf{x}_{i,n})\right) \ , \ n \in [W]. \tag{3.4}$$

Note that we use different feature transformations (θ, ϕ) for the row and column attention calculations.

With the column and row attention maps A_{col} and A_{row}, the final value weighted by the column and row attention maps can be represented by:

$$\mathbf{y}_{i,j} = \sum_{\forall n} \left(A_{\text{row}}(\mathbf{x}_{i,j}, \mathbf{x}_{i,n}) (\sum_{\forall m} A_{\text{col}}(\mathbf{x}_{i,j}, \mathbf{x}_{m,j}) g(\mathbf{x}_{m,n})) \right). \tag{3.5}$$

3.3.2 Channelized Axial Attention

In order to address the feature conflicting issue of the existing dual attention designs, we propose a novel *Channelized* Axial Attention (CAA), which seamlessly combines the advantages of spatial attention and channel attention.

As mentioned in the above sections, feature conflicts may be caused by the different interests of spatial and channel attentions. Ideally, channel attention should not ignore the regional features that are interesting to spatial attention. Conversely, spatial attention should consider channel relation as well.

*We use $i \in [n]$ to denote that i is generated from $[n] = \{1, 2, ..., n\}$.

Thus, we propose to compute channel attention within spatial attention. Specifically, we firstly break down spatial attention into more basic parts (Sect. 3.3.2.1). Then, spatially varying channel attention is generated with $\boldsymbol{\alpha}_{i,j,m,n}$ and $\boldsymbol{\beta}_{i,j,n}$. In this way, channel attention is incorporated into spatial attention and spatial attention will not be ignored when small objects exist, seamlessly and effectively combining spatial and channel attention together.

3.3.2.1 Breaking Down Axial Attention.

For convenience, we firstly define two variables $\boldsymbol{\alpha}_{i,j,m,n}$ and $\boldsymbol{\beta}_{i,j,n}$ to represent the *intermediate weighted features* as follows.

$$\boldsymbol{\alpha}_{i,j,m,n} = A_{\text{col}}(\mathbf{x}_{i,j}, \mathbf{x}_{m,j})g(\mathbf{x}_{m,n}) \tag{3.6}$$

and

$$\boldsymbol{\beta}_{i,j,n} = A_{\text{row}}(\mathbf{x}_{i,j}, \mathbf{x}_{i,n}) \sum_{\forall m} \boldsymbol{\alpha}_{i,j,m,n}. \tag{3.7}$$

Thus, Eq. (3.5) can be rewritten as:

$$\mathbf{y}_{i,j} = \sum_{\forall n} \boldsymbol{\beta}_{i,j,n} = \sum_{\forall n} A_{\text{row}}(\mathbf{x}_{i,j}, \mathbf{x}_{i,n}) \left(\sum_{\forall m} \boldsymbol{\alpha}_{i,j,m,n} \right). \tag{3.8}$$

Eqs. (3.6), (3.7) and (3.8) show that the computation of the dot product is composed of two steps: 1) *Reweighting*: re-weighting features on selected locations by column attention as in Eq. (3.6) and row attention as in Eq. (3.7), and 2) *Summation*: summing the elements along row and column axes in Eq. (3.8). Note that, this breakdown is also applicable to regular self-attention.

3.3.2.2 Spatially Varying Channel Attention.

With the intermediate results $\boldsymbol{\alpha}_{i,j,m,n}$ and $\boldsymbol{\beta}_{i,j,n}$ in Eqs. (3.6) and (3.7), channel relation can be applied inside spatial attention, seamlessly combining them into one operation. In addition, channel attention is now independently conducted on each column or row (on each pixel in regular self-attention) and provides spatial perspective for the channel relation modeling, resulting in our *spatially varying channel attention*. Enhanced with spatially varying channel attentions, now C_{col} and C_{row} are written as:

$$C_{\text{col}}(\boldsymbol{\alpha}_{i,j,m,n}) = \text{Sigmod}\left(\text{ReLU}(\frac{\sum_{\forall m,j}(\boldsymbol{\alpha}_{i,j,m,n})}{H \times W}\omega_{c1})\omega_{c2}\right)\boldsymbol{\alpha}_{i,j,m,n}, \tag{3.9}$$

and

$$C_{\text{row}}(\boldsymbol{\beta}_{i,j,n}) = \text{Sigmod}\left(\text{ReLU}(\frac{\sum_{\forall i,n}(\boldsymbol{\beta}_{i,j,n})}{H \times W}\omega_{r1})\omega_{r2}\right)\boldsymbol{\beta}_{i,j,n}, \tag{3.10}$$

where Sigmod$(\cdot)$ is the learned channel attention, and ω_{c1}, ω_{c2}, ω_{r1} and ω_{r2} are the learned relationships between different channels according to $\boldsymbol{\alpha}_{i,j,m,n}$ and $\boldsymbol{\beta}_{i,j,n}$.

Thus, instead of directly using $\boldsymbol{\alpha}_{i,j,m,n}$ and $\boldsymbol{\beta}_{i,j,n}$ as in Eq. (3.8), for each column and row, we obtain the channelized axial attention features, where the intermediate results $\boldsymbol{\alpha}_{i,j,m,n}$ and $\boldsymbol{\beta}_{i,j,n}$ are weighted by the spatially varying channel attention defined in Eqs. (3.9) and (3.10) as:

$$\mathbf{y}_{i,j} = \sum_{\forall n} C_{\text{row}}\left(A_{\text{row}}(\boldsymbol{x}_{i,j}, \boldsymbol{x}_{i,n})(\sum_{\forall m} C_{\text{col}}(\boldsymbol{\alpha}_{i,j,m,n}))\right). \tag{3.11}$$

Note that the spatially varying channel attention keeps a W dimension after averaging $H \times W$ during the channel attention (Fig. 3.5). Now each row has its own channel attention thanking the breaking down of spatial axial attention.

3.3.2.3 Going Deeper in Channel Attention.

Similar to the work in [20], we use two fully connected layers, followed by ReLU and sigmoid activations respectively, to first reduce the channel number and then increase it to the original channel number.

To further boost performance, we explore the design of more powerful channel attention modules for our channelization since our attention module keeps the spatial dimension, and thus contains more information than a regular SE module ($1 \times 1 \times C \times WorH$ vs $1 \times 1 \times C$, see Fig. 3.5).

We experimented with increased depth and/or width of hidden layers to enhance the capacity of spatial varying channel attention. We find that deeper hidden layers allow channel attention to find a better relationship between channels for our spatially varying channel attention. Moreover, increasing layer width is not as effective as adding layer depth (see Table 3.1).

Furthermore, in spatial domain, each channel of a pixel contains unique information that can lead to a unique semantic representation. We find that using Leaky ReLU [39] is more effective than ReLU in preventing the loss of information along deeper activations [46]. Apparently, this replacement only works in spatially varying channel attention.

3.4 Experiments

To demonstrate the effectiveness for accuracy of the proposed CAA, comprehensive experimental results are compared with the state-of-the-art methods on three benchmark datasets, *i.e.*, PASCAL Context [13], COCO-Stuff [1] and Cityscapes [38].

Using similar settings as in other existing works, we measure the segmentation accuracy using mean intersection over union (mIOU). Moreover, to demonstrate the efficiency of our CAA, we also compare the floating point operations per second (FLOPS) of different approaches. Experimental results show that our CAA outperforms the state-of-the-art methods on all tested datasets.

3.4.1 Implementation Details

3.4.1.1 Backbone

Our network is built on ResNet-101 [18] pre-trained on ImageNet. The original ResNet results in a feature map of 1/32 of the input size. Following other works [5, 27], we apply dilated convolution at the output stride = 16 for ablation experiments if not specified. We conduct experiments with the output stride = 8 to compare with the state-of-the-art methods.

3.4.1.2 Naive Upsampling

Unless otherwise specified, we directly bi-linearly upsampled the logits to the input size without refining using any low-level and high resolution features.

3.4.1.3 Training Settings

We employ stochastic gradient descent (SGD) for optimization, where the polynomial decay learning rate policy $(1 - \frac{iter}{maxiter})^{0.9}$ is applied with an initial learning rate = 0.01. We use synchronized batch normalization with batch size = 16 (8 for Cityscapes) during training. For data augmentation, following the practice of similar works [3], we only apply the most basic data augmentation strategies as in [5], including random flip, random scale and random crop.

3.4.2 Results on PASCAL Context

The PASCAL Context [41] dataset has 59 classes with 4,998 images for training and 5,105 images for testing. We train our CAA on the training set for 40k iterations. In the following, we first present a series of ablative experiments to show the effectiveness of our method. Then, quantitative and qualitative comparisons with other state-of-the-art methods are presented.

Note that, in ablation studies below, we report mean result with standard deviation (numbers in parentheses) calculated with 5 repeated experiments.

3.4.2.1 Effectiveness of the Proposed Channelization

We first report the impact of adding channelized axial attention and with different depth and width in Table 3.1, where '-' for the baseline result indicates no channelization is performed.

Layer Counts	# of Channels	mIOU (%)	FLOPs
-	-	50.27(±0.2)	68.7G
1	128	50.75(±0.2)	+0.00024G
3	128	50.85(±0.2)	+0.00027G
5	128	**51.06(±0.2)**	+0.00030G
7	128	50.40(±0.3)	+0.00043G
5	64	50.12(±0.2)	+0.00015G
5	256	50.35(±0.4)	+0.00098G

Table 3.1 : Results without using channelization (Row 1) and using channelization with different layer counts and channel numbers. Numbers in parentheses indicate standard deviations (see Sect. 3.4.2).

As can be seen from this table, our proposed channelization improves the mIOU over the baseline regardless of the layer counts and the number of channels used. In particular, the results shown in the table indicate that the best performance is achieved when the Layer Counts = 5 and the number of Channels = 128 (see the 3rd row in Table 3.1.

We also compare our model with the sequential design of "Axial Attention + SE", as shown in Table 3.2. We find the sequential design brings only marginal contributions to performance, showing that our proposed channelization method can combine the advantages of both spatial attention and channel attention more effectively. In Table 3.5, results obtained with *other backbones* are provided to demonstrate the effectiveness and robustness of CAA.

Axial Attention	+ SE	+ Our Channelization
50.27(±0.2)	50.37(±0.2)	51.06(±0.2)

Table 3.2 : Result comparison between axial attention, axial attention + SE and channelized axial attention.

Attention Base	Eval OS	Channelized	mIOU (%)
Axial Attention	16		50.27
	16	✓	51.06
Self Attention	16		50.42
	16	✓	51.09

Table 3.3 : Ablation study of applying our Channelized Attention on self-attention with ResNet-101. **Eval OS**: Output strides [5] during evaluation.

3.4.2.2 Channelized Self-Attention

In this section, we conduct experiments on the PASCAL Context by applying channelization to the original self-attention. We report its single-scale performance in Table 3.3 with ResNet-101. Our channelized method can also further improve the performance of self-attention by 0.67% (51.09% vs 50.42%).

3.4.2.3 Impact of the Testing Strategies

We compare the performance and computation cost of our proposed model against the baseline and DANet [14] with different testing strategies in Table 3.4. Using the same settings as those in other works [14], we add multi-scale, left-right flip and auxiliary loss during inference. The accuracies of CAA are further boosted with output stride = 8 since the channel attention can learn and optimize three times more pixels.

Methods	OS	MF	Aux	mIOU (%)	FLOPs
ResNet	16		-	-	59.85G
-101	8		-	-	190.70G
DANet	8				+101.25G
	8	✓	✓	52.60	-
Axial	16			50.27(±0.2)	+8.85G
Attention	16	✓		52.01(±0.2)	-
	8			51.24(±0.2)	+34.33G
	8	✓		52.51(±0.2)	-
Our	16			51.06(±0.2)	+8.85G
CAA	16	✓		53.09(±0.3)	-
	8			52.73(±0.1)	+34.33G
	8	✓		54.05(±0.1)	-
Our	16		✓	51.80(±0.2)	+8.85G
CAA	16	✓	✓	53.52(±0.2)	-
+	8		✓	53.48(±0.3)	+34.33G
Aux loss	8	✓	✓	54.65(±0.4)	-

Table 3.4 : Comparing results with different testing strategies. **OS:** Output stride in training and inference. **MF:** Apply multi-scale and left-right flipping during inference. **Aux:** Add auxiliary loss during training. "+" refers to the extra FLOPS over the baseline FLOPS of ResNet-101.

Backbone	OS	AA	C	mIOU (%)
ResNet-50	16	✓		49.73
[18]	16	✓	✓	**50.23**
Xception65	16	✓		52.42
[10]	16	✓	✓	**52.65**
EfficientNetB7	16	✓		57.24
[40]	16	✓	✓	**57.93**
	8	✓	✓	**58.40**

Table 3.5 : Ablation study of CAA with the backbones other than ResNet-101. All results are obtained in single scale without flipping. **OS**: Output strides during evaluation. **AA**: Axial Attention. **C**: Channelized.

3.4.2.4 Comparison with the State-of-the-art

Finally, in Table 3.6, we compare our approach with the state-of-the-art approaches. Like other similar works, we apply multi-scale and left-right flip during inference. For a fair comparison, we only compare with the methods that use ResNet-101 and naive upsampling in Table 3.6. More results using alternative backbones are included in Table 3.5.

As shown in this table, our proposed CAA outperforms all listed state-of-the-art models that are trained with an output stride = 8. Our CAA also performs better than EMANet and SPYGR that are trained with output stride = 16. Note that, in this and the following tables, we report the best results of our approach obtained in experiments.

In Fig. 3.6, we show some results obtained by our CAA model, FCN and Dual

Methods	Backbone	mIOU (%)
ENCNet [65]	ResNet-101	51.7
ANNet [72]	ResNet-101	52.8
EMANet [27]	ResNet-101	53.1
SPYGR [26]	ResNet-101	52.8
CPN [60]	ResNet-101	53.9
CFNet [67]	ResNet-101	54.0
DANet [14]	ResNet-101	52.6
Our CAA (OS = 16)	ResNet-101	53.7
Our CAA (OS = 8)	ResNet-101	**55.0**

Table 3.6 : Comparisons with other state-of-the-art approaches on the PASCAL Context test set. For a fair comparison, all compared methods use ResNet-101 and naive upsampling.

attention. Our model is able to handle previous failure cases better, especially when a class A covering a smaller region is surrounded by another class B covering a much larger region (see the boxed regions).

3.4.2.5 Stronger Backbone in PASCAL Context

As mentioned in above, our CAA outperforms the SOTA methods [67, 27] with the same settings (ResNet-101 + naive upsampling). Furthermore, we show that our proposed CAA is suitable for different backbones.

In this section, we report our CAA's performance with EfficientNet [40] in Table 3.7. Note that, this is not a fair comparison, since the listed methods were not trained under the same settings, or using the same backbone. The results show that our method can outperform the state-of-the-art Transformer [12] based hybrid

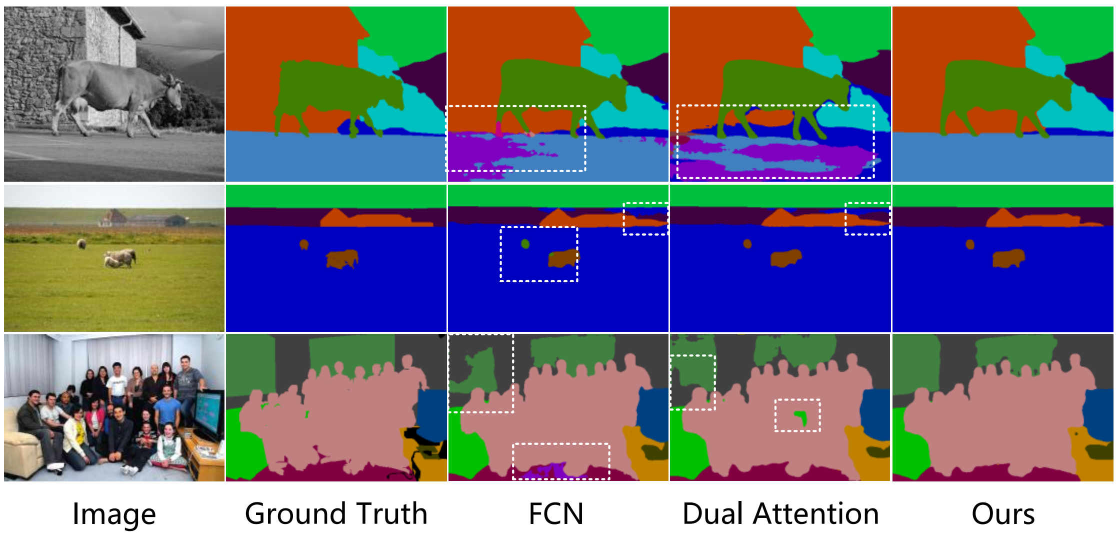


Figure 3.6 : Examples of the segmentation results obtained on the PASCAL Context dataset using FCN, DANet and CAA.

models such as SETR [48] and DPT [43] with the CNN backbone EfficientNet-B7. The *simple decoder* merges the low level features from output stride = 4, during the final upsampling (see [5] for details).

3.4.3 Results on COCO-Stuff 10K

Following the other works [14], we evaluate our CAA on COCO-Stuff 10K dataset [1], which contains 9,000 training images and 1,000 testing images with 172 classes. Our model is trained for 40k iterations. As shown in Table 3.8, our proposed CAA outperforms all other state-of-the-art approaches by a large margin of 1.3%, demonstrating that our model can better handle complex images with a large number of classes.

3.4.3.1 Stronger Backbone in COCOStuff-10K

We also report our CAA's results using Efficientnet-b7 [40] as the backbone in Table 3.9.

Methods	mIOU (%)
CTNet [29] + JPU	55.5
SETR-MLA [48]	55.83
HRNetV2 + OCR [11]	56.2
ResNeSt-269 [66] + DeepLab V3+	58.9
HRNetV2 + OCR + RMI	59.6
DPT-Hybrid [43]	60.46
Our CAA (EfficientNet-B7, w/o decoder)	60.12
Our CAA (EfficientNet-B7 + simple decoder)	**60.50**

Table 3.7 : Result comparison with the state-of-the-art approaches on the PASCAL Context test set for multi-scale prediction. Note that, the listed methods were not trained under the same settings, or using same backbone.

3.4.4 Results on Cityscapes

Following previous works [14], we use only the *fine* set with a crop size of 769×769 during training, and our training iteration is set to 80k. We report our results on the *test* set in Table 3.10. The results show our CAA is also working well on high-resolution images.

3.4.5 Results on COCOStuff-164k

The recent method Segformer [57] used COCOStuff-164k (164,000 images), *i.e.*, the full set of COCOStuff-10k to validate its performance for the first time. Since Segformer is a strong backbone, in this section, we also use EfficientNet-B5 + CAA to verify the robustness of our CAA on COCOStuff-164k. Table 3.11 shows that our method outperforms the recent strong baselines Segformer and SETR [48] by a

Methods	Backbone	mIOU (%)
DSSPN [30]	ResNet-101	38.9
SVCNet [11]	ResNet-101	39.6
EMANet [27]	ResNet-101	39.9
SPYGR [26]	ResNet-101	39.9
OCR [61]	ResNet-101	39.5
DANet [14]	ResNet-101	39.7
Our CAA	ResNet-101	**41.2**

Table 3.8 : Comparisons with other state-of-the-art approaches on the COCO-Stuff 10K test set. For a fair comparison, all compared methods use ResNet-101 and naive upsampling.

large margin, indicating our CAA keeps the superior performance with large training data.

3.5 Extra Visualizations

3.5.1 COCOStuff-10k

Fig. 3.7 shows some examples of the segmentation results obtained on the COCO-Stuff 10K with our proposed CAA in comparison to the results of FCNs [37], DANet [14], and the ground truth (output stride = 8, ResNet-101). As it can be seen, our CAA can segment common objects such as building, human, or sea very well.

Methods	mIOU (%)
HRNetV2 + OCR [11]	40.5
DRAN	41.2
HRNetV2 + OCR + RMI	45.2
Our CAA (EfficientNet-B7)	**45.4**

Table 3.9 : Result comparison with the state-of-the-art approaches on the COCO-Stuff-10K test set for multi-scale prediction. Note that, the listed methods were not trained under the same settings, or using same backbone.

3.5.2 PASCAL Context

In this section, we show more examples of the segmentation results obtained on the PASCAL Context in Fig. 3.8. The results show that the failure cases in FCN and DANet are segmented much better by our CAA, especially hard cases (see the 2nd row).

3.5.2.1 Cityscapes

In Fig. 3.9, we compare the segmentation results obtained on Cityscapes validation set with DANet and our CAA. Key areas of difference are highlighted with white boxes. The results show that many errors produced by DANet no longer exist in our CAA results.

3.6 Pseudo Code of Group Vectorization

Algorithm 2 presents the pseudo code of implementing the proposed grouped vectorization.

As shown in the Algorithm, the attention map A is splitted into G groups along

Methods	Backbone	mIOU (%)
CFNet [67]	ResNet-101	79.6
ANNet [72]	ResNet-101	81.3
CCNet [23]	ResNet-101	81.4
CPN [60]	ResNet-101	81.3
SPYGR [26]	ResNet-101	81.6
OCR [61]	ResNet-101	81.8
DANet [14]	ResNet-101	81.5
Our CAA	ResNet-101	**82.6**

Table 3.10 : Comparisons with other state-of-the-art approaches on the Cityscapes test set. For a fair comparison, all compared methods use ResNet-101 and naive upsampling.

the second dimension. Padding is applied to ensure the equal size between groups. After the channelization is applied on each groups, the output feature map is recovered by the concatenation of all the groups.

3.7 Limitation

Since CAA breaks the regular dot-product operation of self-attention, and inserts channel attention inside it, though it has a very small computational overhead as shown in Tab 3.1, its actual speed may be much slower than the standard self-attention, because the standard self-attention has been well optimized by CUDNN, etc.

Methods	mIOU (%)
ResNet-50 + DeepLabV3+ [5]	38.4
HRNetV2 + OCR	42.3
SETR [48]	45.8
Segformer-B5 [57]	46.7
Our CAA (EfficientNet-B5)	**47.30**

Table 3.11 : Result comparison with the state-of-the-art approaches on the COCO-Stuff-164K test set for multi-scale prediction. Note that, the listed methods were not trained under same settings, or using same backbone. *Methods* other than CAA and Segformer are reproduced in Segformer paper.

3.8 Summary

In this chapter, a novel and effective Channelized Axial Attention has been proposed, and it has effectively combined the advantages of the popular spatial-attention and channel attention. Specifically, we have first broken down the spatial attention into two steps and inserted channel attention in between, enabling different spatial positions to have their own channel attentions. Experiments on the three popular benchmark datasets have demonstrated the effectiveness of our proposed channelized axial attention.

Algorithm 2 Our proposed grouped vectorization algorithm

```
Require: $G$: Group Number, $A$: Attention Map $[N, H, H, W]$, $X$: Feature Map $[N, C, H, W]$
 1: $padding \leftarrow H \,\%\, G$
 2: $A \leftarrow$ Transpose $A$ into $[H, N, H, W]$
 3: $H^+ \leftarrow H + padding$
 4: $A \leftarrow$ padding zero to $A$ into $[H^+, N, H, W]$
 5: $A \leftarrow$ Reshape $A$ into $[G, H^+ \,//\, G, N, H, W]$
 6: for $g \in G$ do
 7:     $Y_g \leftarrow$ Channelization $(X, A_g)$, $Y_g \in [H^+ \,//\, G, N, C, W]$
 8: end for
 9: $Y \leftarrow$ Concat$(Y_{0,1,\ldots G})$, $Y \in [G, H^+ \,//\, G, N, C, W]$
10: $Y \leftarrow$ Reshape $Y$ into $[H^+, N, C, W]$
11: $Y \leftarrow$ Remove padding from $Y$ into $[H, N, C, W]$
12: $Y \leftarrow$ Transpose $Y$ into $[N, C, H, W]$
        return Y
```

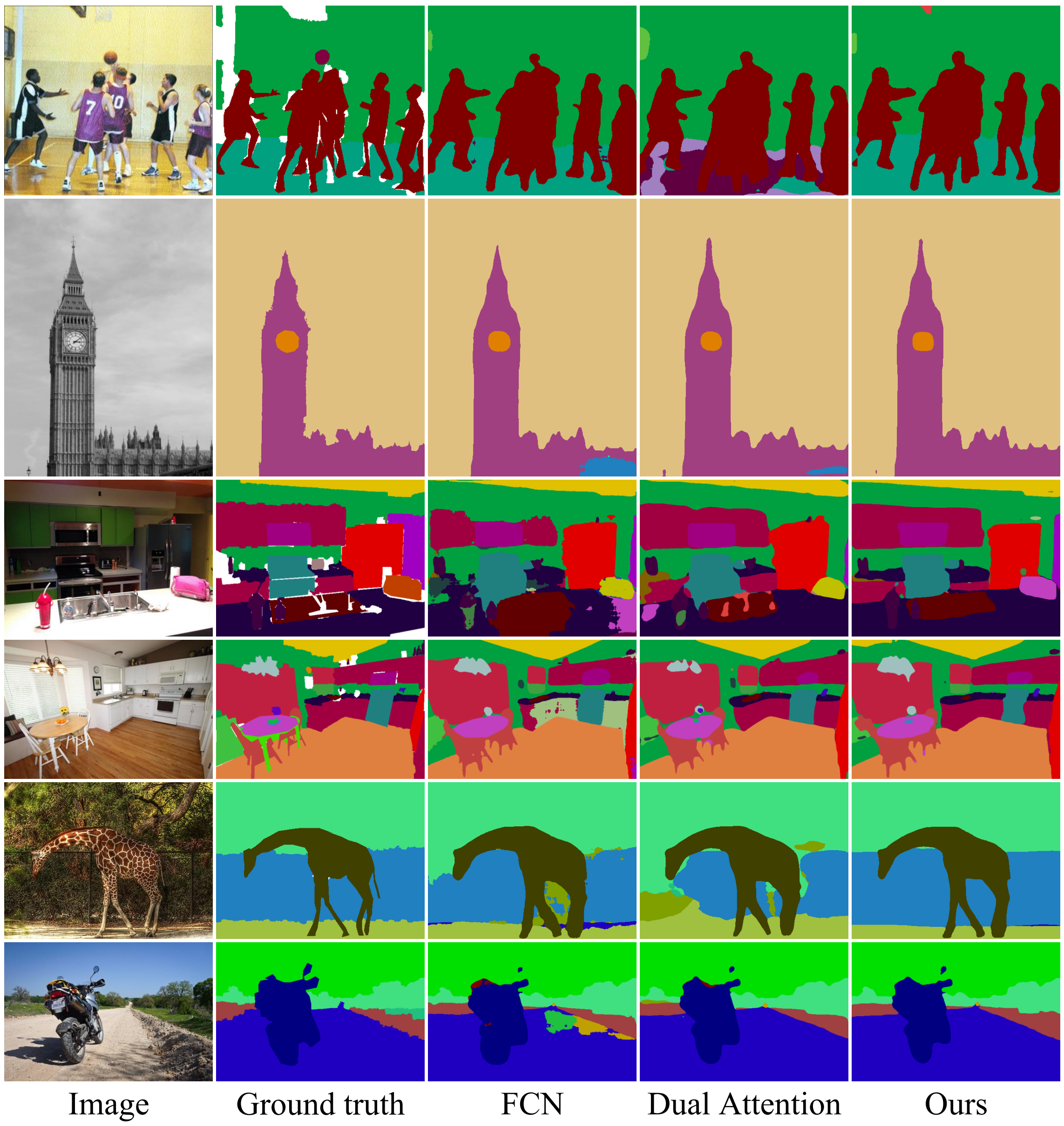


Figure 3.7 : Examples of the results obtained on the COCO-Stuff 10K dataset with our proposed CAA in comparison to the results obtained with FCN, DANet and the ground truth.

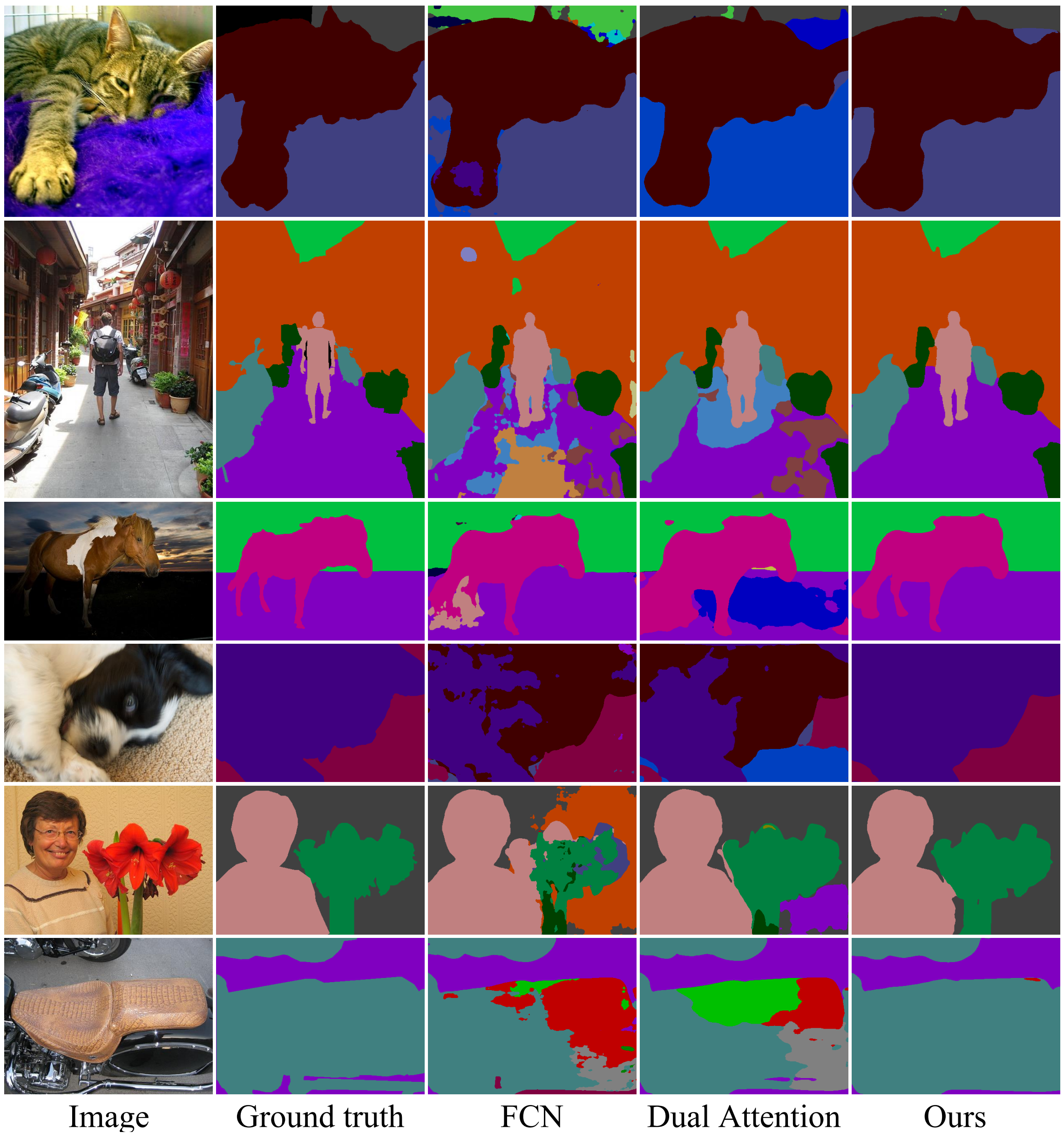


Figure 3.8 : Examples of the results obtained on the PASCAL Context dataset with our proposed CAA in comparison to the results obtained with FCN, DANet and the ground truth.

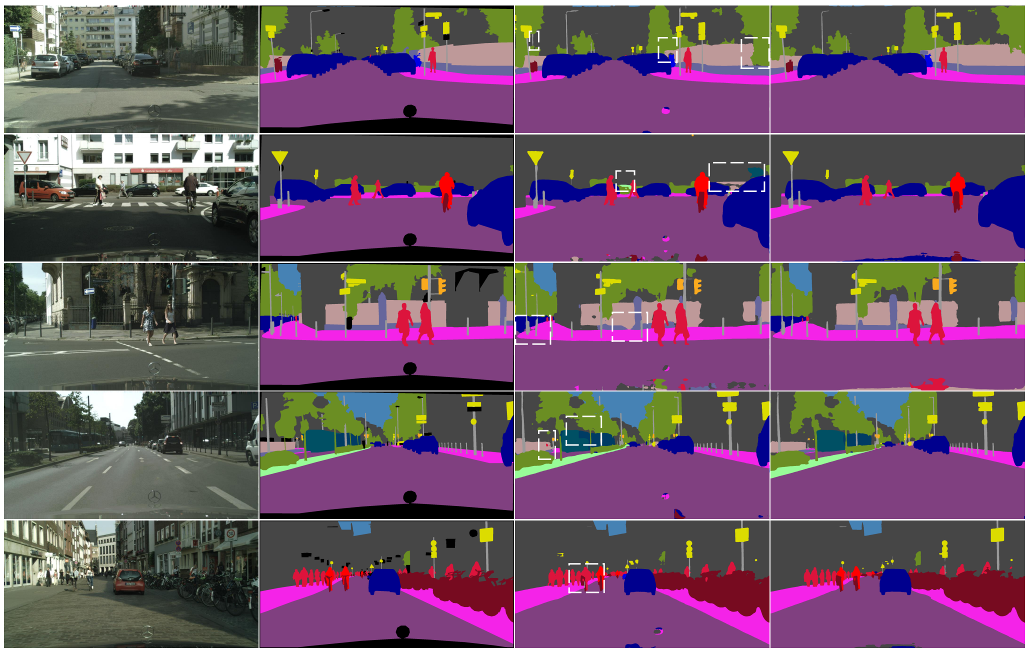


Figure 3.9 : Extra examples of the segmentation results obtained on the Cityscapes validation set [38] with our proposed CAA in comparison to the results obtained with DANet [14] and the ground truth.

Chapter 4

CAR: Class-aware Regularization

4.1 Introduction

After the early work FCNs [37], which used fully convolutional networks to make the dense per-pixel segmentation task more efficient, many works [69, 3] have been proposed which have greatly advanced the segmentation accuracy on various benchmark datasets. Among these methods, many of them have focused on better fusing spatial domain context information to obtain more powerful feature representations (termed *pixel features* in this work) for the final per-pixel classification. For example, VGG [47] utilized large square context information by successfully training a very deep network, and DeepLab [3] and PSPNet [69] utilized multi-scale features with the ASPP and PPM modules.

Recently, methods based on dot-product self-attention (SA) [53, 14, 62, 67, 27, 72, 12, 43, 48] have become very popular since they can easily capture the long-range relationship between pixels. SA aggregates information dynamically (by different attention maps for different inputs) and selectively (using weighted averaging spatial features according to their similarity scores). Using multi-scale and self-attention techniques during spatial information aggregation has worked very well (*e.g.*, 80% mIOU on Cityscapes [38] (single-scale w/o flipping)).

As complements to the above methods, many recent works have proposed various modules to utilize class-level contextual information. The class-level information is often represented by the class center/context prior, which contains the mean features of each class in the images. OCR [61] and ACFNet [64] extract "soft" class centers according to the predicted coarse segmentation mask by using the weighted sum. CPNet [60] proposed a context prior map/affinity map, which indicates if two spatial locations belong to the same class, and used this predicted context prior map for feature aggregation. However, they [61, 64, 60] simply concatenated these class-level features with the original pixel features for the final classification.

In this chapter, we also focus on utilizing class level information. Instead of focusing on how to better extract class-level features like the existing methods [61, 64, 60], we use the simple, but accurate, average feature according to the GT mask, and focus on maximizing the inter-class distance during feature learning. This is because it mirrors how humans can robustly recognize an object by itself no matter what other objects it appears with.

Learning more separable features makes the features of a class less dependent upon other classes, resulting in improved generalization ability, especially when the training set contains only limited and biased class combinations (*e.g.*, cows and grass, boats and beach). Fig. 4.1 illustrates an example of such a problem, where the classification of dog and sheep depends on the classification of grass class, and has been mis-classified as cow. In comparison, networks trained with our proposed CAR successfully generalize to these unseen class combinations.

To better achieve this goal, we propose CAR, a class-aware regularizations module, that optimizes the class center (intra-class) and inter-class dependencies during feature learning. Three loss functions are devised. The first loss encourages more compact class representations within each class, and the other two directly maximize the distance between different classes. Specifically, an intra-class center-to-pixel loss (termed as "intra-c2p", Eq. (4.3)) is first devised to produce more compact representation within a class by minimizing the distance between all pixels and their class center. In our work, a class center is calculated as the averaged feature of

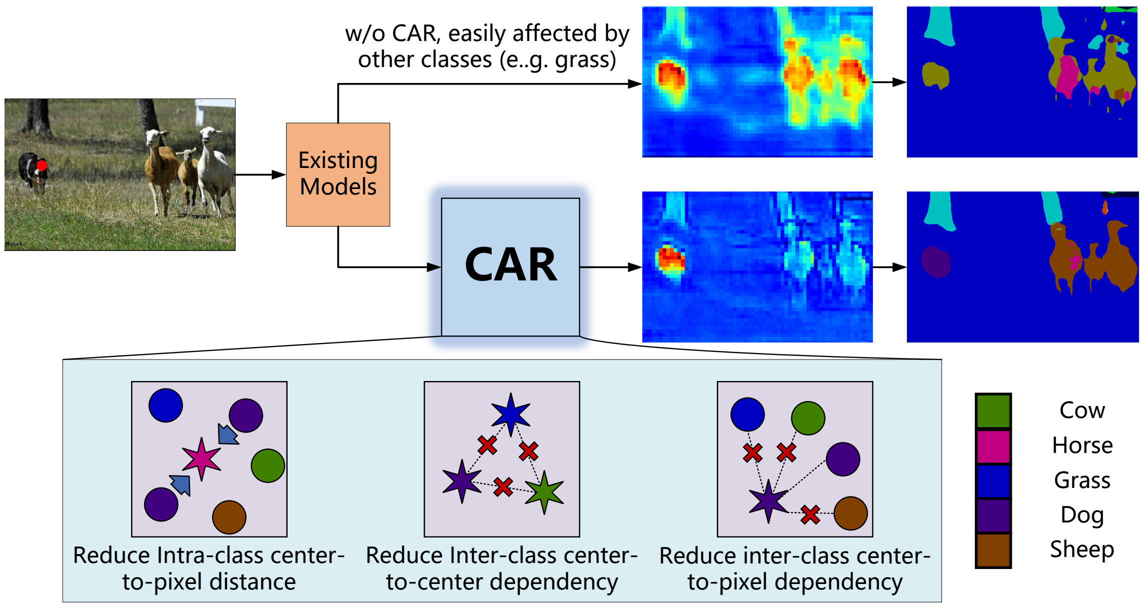

w/o CAR, easily affected by other classes (e..g. grass)
Existing Models
CAR
Reduce Intra-class center-to-pixel distance
Reduce Inter-class center-to-center dependency
Reduce inter-class center-to-pixel dependency
Cow
Horse
Grass
Dog
Sheep

Figure 4.1 : The concept of the proposed CAR. Our CAR optimizes existing models with three regularization targets: 1) reducing pixels' intra-class distance, 2) reducing inter-class center-to-center dependency, and 3) reducing pixels' inter-class dependency. As highlighted in this example (indicated with a red dot in the image), with our CAR, the grass class does not affect the classification of dog/sheep as much as before, and hence successfully avoids previous (w/o CAR) mis-classification.

all pixels belonging to the same class according to the GT mask. More compact intra-class representations leave a relatively large margin between classes, thus contributing to more separable representations. Then, an inter-class center-to-center loss ("inter-c2c", Eq. (4.6)) is devised to maximize the distance between any two different class centers. This inter-class center-to-center loss alone does not necessarily produce separable representations for every individual pixels. Therefore, a third inter-class center-to-pixel loss ("inter-c2p", Eq. (4.13)) is proposed to enlarge the distance between every class center and all pixels that do not belong to the class.

In summary, our contributions in this work are as follows.

1. We propose a universal class-aware regularization module that can be inte-

grated into various segmentation models to largely improve the accuracy.

2. We devise three novel regularization terms to achieve more separable and less class-dependent feature representations by minimizing the intra-class variance and maximizing the inter-class distance.
3. We calculate the class centers directly from ground truth during training, thus avoiding the error accumulation issue of the existing methods and introducing no computational overhead during inference.
4. We provide image-level feature-similarity heatmaps to visualize the learned inter-class features with our CAR are indeed less related to each other.

We conduct extensive experiments on many baselines and demonstrate that our CAR can improve all SOTA methods substantially, including CNN and Transformer based models.

4.2 Related Work

4.2.1 Self-Attention

Dot-product self-attention proposed in [53, 50] has been widely used in semantic segmentation [14, 62, 67, 72]. Specifically, self-attention determines the similarity between a pixel with every other pixel in the feature map by calculating their dot product, followed by softmax normalization. With this attention map, the feature representation of a given pixel is enhanced by aggregating features from the whole feature map weighted by the aforementioned attention value, thus easily taking long-range relationship into consideration and yielding boosted performance. In self-attention, in order to achieve correct pixel classification, the representation of pixels belonging to the same class should be similar to gain greater weights in the final representation augmentation.

4.2.2 Class Center

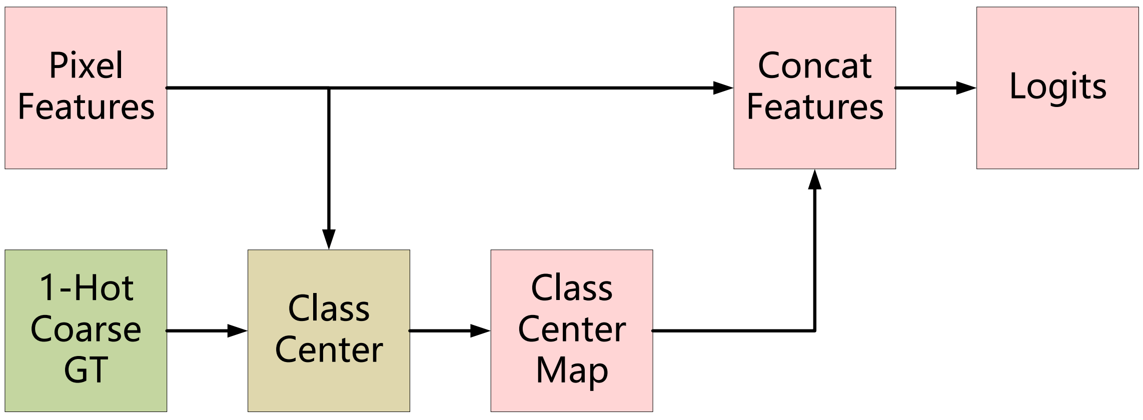
In 2019 [64, 61], the concept of *class center* was introduced to describe the overall representation of each class from the categorical context perspective. In these approaches, the center representation of each class was determined by calculating the dot product of the feature map and the coarse prediction (*i.e.*, weighted average) from an auxiliary task branch, supervised by the ground truth [69]. After that, those intra-class centers are assigned to the corresponding pixels on feature map. Furthermore, in 2020 [60], a learnable kernel and one-hot ground truth were used to separate the intra-class center from inter-class center, and then concatenated with the original feature representation.

All of these works [61, 64, 60] have focused on extracting the intra (inter) class centers, but they then simply concatenated the resultant class centers with the original pixel representations to perform the final logits. We argue that the categorical context information can be utilized in a more effective way so as to reduce the inter-class dependency.

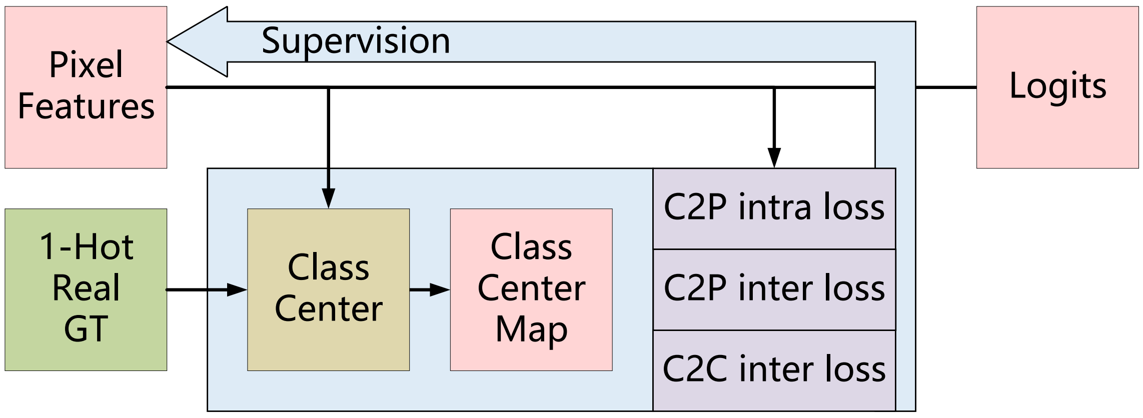
To this end, we propose a CAR approach, where the extracted class center is used to directly regularize the feature extraction process so as to boost the differentiability of the learned feature representations (see Fig. 4.1) and reduce their dependency on other classes. Fig. 4.2 contrasts the two different designs. More details of the proposed CAR are provided in Sect. 4.3.

4.2.3 Inter-Class Reasoning

Recently, [9, 34] studied the class dependency as a dataset prior and demonstrated that inter-class reasoning could improve the classification performance. For example, a car usually does not appear in the sky, and therefore the classification of sky can help reduce the chance of mis-classifying an object in the sky as a car. However, due to the limited training data, such class-dependency prior may also contain



(a) Design of OCR, ACFNet and CPNet



(b) Our CAR

Figure 4.2 : The difference between the proposed CAR and previous methods that use class-level information. Previous models focus on extracting class center while using simple concatenation of the original pixel feature and the class/context feature for later classification. In contrast, our CAR uses direct supervision related to class center as regularization during training, resulting in small intra-class variance and low inter-class dependency. See Fig. 4.1 and Sect. 4.3 for details.

bias, especially when the desired class relation rarely appears in the training set.

Fig. 4.1 shows such an example. In the training set, cow and grass are dependent on each other. However, as shown in this example, when there is a dog or sheep standing on the grass, the class dependency learned from the limited training data may result in errors and predict the target into a class that appears more often in the training data, *i.e.*, cow in this case. In our CAR, we design inter-class and intra-class loss functions to reduce such inter-class dependency and achieve more robust segmentation results.

4.3 Methodology

4.3.1 Extracting Class Centers from Ground Truth

Denote a feature map and its corresponding resized one-hot encoded ground-truth mask as $\mathbf{X} \in \mathbb{R}^{H \times W \times C*}$ and $\mathbf{Y} \in \mathbb{R}^{H \times W \times N_{\text{class}}}$, respectively. We first get the spatially flattened class mask $\mathbf{Y}_{\text{flat}} \in \mathbb{R}^{HW \times N_{\text{class}}}$ and flattened feature map $\mathbf{X}_{\text{flat}} \in \mathbb{R}^{HW \times C}$. Then, the class center[†], which is the average features of all pixel features of a class, can be calculated by:

$$\boldsymbol{\mu}_{image} = \frac{\mathbf{Y}_{\text{flat}}^{T} \cdot \mathbf{X}_{\text{flat}}}{\mathbf{N}_{\text{non-zero}}} \in \mathbb{R}^{N_{\text{class}} \times C}, \tag{4.1}$$

where $\mathbf{N}_{\text{non-zero}}$ denotes the number of non-zero values in the corresponding map of the ground-truth mask $\mathbf{Y}$. In our experiments, to alleviate the negative impact of noisy images, we calculate the class centers using all the training images in a batch, and denote them as $\boldsymbol{\mu}_{\text{batch}}$[‡]

4.3.2 Reducing Intra-Class Feature Variance

4.3.2.1 Motivation.

More compact intra-class representation can lead to a relatively larger margin between classes, and therefore result in more separable features. In order to reduce

[*] H, W and C denote images' height and width, and number of channels, respectively.

[†] It is termed as *class center* in [64] and *object region representations* in [61].

[‡] We use $\boldsymbol{\mu}$ and omit the subscript *batch* for clarify.

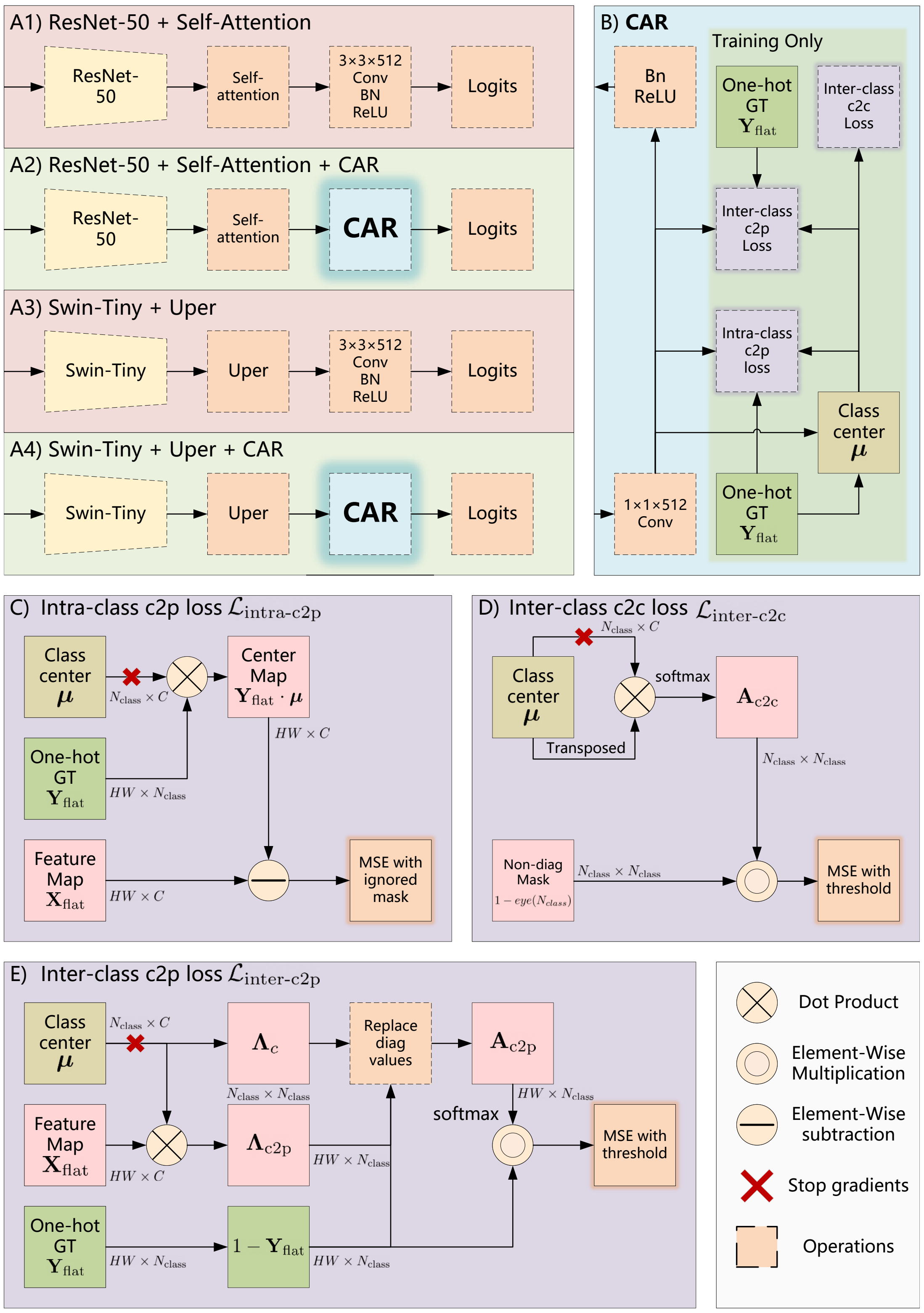

A1) ResNet-50 + Self-Attention
ResNet-50
Self-attention
3×3×512 Conv BN ReLU
Logits
A2) ResNet-50 + Self-Attention + CAR
CAR
A3) Swin-Tiny + Uper
Swin-Tiny
Uper
A4) Swin-Tiny + Uper + CAR
B) CAR
Training Only
Bn ReLU
One-hot GT $\mathbf{Y}_{\text{flat}}$
Inter-class c2c Loss
Inter-class c2p Loss
Intra-class c2p loss
Class center $\boldsymbol{\mu}$
1×1×512 Conv
C) Intra-class c2p loss $\mathcal{L}_{\text{intra-c2p}}$
Center Map $\mathbf{Y}_{\text{flat}} \cdot \boldsymbol{\mu}$
$N_{\text{class}} \times C$
$HW \times C$
$HW \times N_{\text{class}}$
Feature Map $\mathbf{X}_{\text{flat}}$
MSE with ignored mask
D) Inter-class c2c loss $\mathcal{L}_{\text{inter-c2c}}$
softmax
$\mathbf{A}_{\text{c2c}}$
Transposed
$N_{\text{class}} \times N_{\text{class}}$
Non-diag Mask $1 - eye(N_{class})$
MSE with threshold
E) Inter-class c2p loss $\mathcal{L}_{\text{inter-c2p}}$
$\boldsymbol{\Lambda}_c$
Replace diag values
$\mathbf{A}_{\text{c2p}}$
$\boldsymbol{\Lambda}_{\text{c2p}}$
$HW \times N_{\text{class}}$
$1 - \mathbf{Y}_{\text{flat}}$
Dot Product
Element-Wise Multiplication
Element-Wise subtraction
Stop gradients
Operations

Figure 4.3 : **The proposed CAR approach.** CAR can be inserted into various segmentation models, right before the logit prediction module (A1-A4). CAR contains three regularization terms, including (C) intra-class center-to-center loss $\mathcal{L}_{\text{intra-c2p}}$ (Sect. 4.3.2.2), (D) inter-class center-to-center loss $\mathcal{L}_{\text{inter-c2c}}$ (Sect. 4.3.3.2), and (E) inter-class center-to-pixel loss $\mathcal{L}_{\text{inter-c2p}}$ (Sect. 4.3.3.3).

the intra-class feature variance, existing works [53, 14, 72, 60, 27, 62] usually use self-attention to calculate the dot-product similarity in spatial space to encourage similar pixels to have a compact distance implicitly. For example, the self-attention in [53] implicitly pushed the feature representation of pixels belonging to the same class to be more similar to each other than those of pixels belonging to other classes. In our work, we devise a simple *intra-class center-to-pixel loss* to guide the training, which can achieve this goal very effectively and produce improved accuracy.

4.3.2.2 Intra-class Center-to-pixel Loss.

We define a simple but effective intra-class center-to-pixel loss to suppress the intra-class feature variance by penalizing large distance between a pixel feature and its class center. The Intra-class Center-to-pixel Loss $\mathcal{L}_{\text{intra-c2p}}$ is defined by:

$$\mathcal{L}_{\text{intra-c2p}} = f_{\text{mse}}(\mathcal{D}_{\text{intra-c2p}}), \tag{4.2}$$

where

$$\mathcal{D}_{\text{intra-c2p}} = (1 - \sigma)|\mathbf{Y}_{\text{flat}} \cdot \boldsymbol{\mu} - \mathbf{X}_{\text{flat}}|. \tag{4.3}$$

In Eq. (4.3), σ is a spatial mask indicating pixels being ignored (*i.e.*, ignore label), $\mathbf{Y}_{\text{flat}} \cdot \boldsymbol{\mu}$ distributes the class centers $\boldsymbol{\mu}$ to the corresponding positions in each image. Thus, our intra-class loss $\mathcal{L}_{\text{intra-c2p}}$ will push the pixel representations to their corresponding class center, using mean squared error (MSE) in Eq. (4.3).

4.3.3 Maximizing Inter-class Separation

4.3.3.1 Motivation

Humans can robustly recognize an object by itself regardless which other objects it appears with. Conversely, if a classifier *heavily* relies on the information from other classes to determine the classification result, it will easily produce wrong classification results when a rather rare class combination appears during inference.

Maximizing inter-class separation, or in another words, reducing the inter-class dependency, can therefore help the network generalize better, especially when the training set is small or is biased.

As shown in Fig. 4.1, the dog and sheep are mis-classified as the cow because cow and grass appear together more often in the training set. To improve the robustness of the model, we propose to reduce this inter-class dependency. To this end, the following two loss functions are defined.

4.3.3.2 Inter-class Center-to-center Loss

The first loss function is to maximize the distance between any two different class centers. Inspired by the center loss used in face recognition [54], we propose to reduce the similarity between class centers $\boldsymbol{\mu}$, which are the averaged features of each class calculated according to the GT mask. The *inter-class relation* is defined by the dot-product similarity [50] between any two classes as:

$$\mathbf{A}_{\text{c2c}} = \text{softmax}(\frac{\boldsymbol{\mu}^T \cdot \boldsymbol{\mu}}{\sqrt{C}}), \quad \mathbf{A}_{\text{c2c}} \in \mathbb{R}^{N_{class} \times N_{class}}. \tag{4.4}$$

Moreover, since we only need to constrain the inter-class distance, only the non-diagonal elements are retained for the later loss calculation as:

$$\mathbf{D}_{\text{inter-c2c}} = \Big(1 - eye(N_{class})\Big)\mathbf{A}_{\text{c2c}}. \tag{4.5}$$

We only penalize larger similarity values between any two different classes than a pre-defined threshold $\frac{\epsilon_0}{N_{class}-1}$, *i.e.*,

$$\mathcal{D}_{\text{inter-c2c}} = f_{\text{sum}}\Big(\max(\mathbf{D}_{\text{inter-c2c}} - \frac{\epsilon_0}{N_{class}-1}, 0)\Big). \tag{4.6}$$

Thus, the Inter-class Center-to-center Loss $\mathcal{L}_{\text{inter-c2c}}$ is defined by:

$$\mathcal{L}_{\text{inter-c2c}} = f_{\text{mse}}(\mathcal{D}_{\text{inter-c2c}}). \tag{4.7}$$

Here, a small margin is used in consideration of the feature space size and the mislabeled ground truth.

4.3.3.3 Inter-class Center-to-pixel Loss.

Maximizing only the distances between class centers does not necessarily result in separable representation for every individual pixels. We further maximize the distance between a class center and any pixel that does not belong to this class. More concretely, we first compute the center-to-pixel dot product as:

$$\mathbf{\Lambda}_{\text{c2p}} = \boldsymbol{\mu}^T \cdot \mathbf{X}_{\text{flat}}, \quad \mathbf{\Lambda}_{\text{c2p}} \in \mathbb{R}^{HW \times N_{\text{class}}}. \tag{4.8}$$

Ideally, with the previous loss $\mathcal{L}_{\text{inter-c2c}}$, the features of all pixels belonging to the same class should be equal to that of the class center. Therefore, we replace the intra-class dot product with its ideal value, namely using the class center $\boldsymbol{\mu}$ for calculating the intra-class dot product as:

$$\mathbf{\Lambda}_c = diag(\boldsymbol{\mu}^T \cdot \boldsymbol{\mu}), \tag{4.9}$$

and the replacement effect is achieved by using masks as:

$$\mathbf{\Lambda}' = \mathbf{\Lambda}_{\text{c2p}}(1 - \mathbf{Y}_{\text{flat}}) + \mathbf{\Lambda}_c. \tag{4.10}$$

This updated dot product $\mathbf{\Lambda}'$ is then used to calculate similarity across class axis with a softmax as:

$$\mathbf{A}_{\text{c2p}} = \text{softmax}(\mathbf{\Lambda}'), \quad \mathbf{A}_{\text{c2p}} \in \mathbb{R}^{HW \times N_{\text{class}}}. \tag{4.11}$$

Similar to the calculation of $\mathcal{L}_{\text{inter-c2c}}$ in the previous subsection, we have

$$\mathbf{D}_{\text{inter-c2p}} = (1 - \mathbf{Y}_{\text{flat}})\mathbf{A}_{\text{c2p}}, \tag{4.12}$$

and

$$\mathcal{D}_{\text{inter-c2p}} = f_{\text{sum}}\Big(\max(\mathbf{D}_{\text{inter-c2p}} - \frac{\epsilon_1}{N_{\text{class}} - 1}, 0)\Big). \tag{4.13}$$

Thus, the Inter-class Center-to-pixel Loss $\mathcal{L}_{\text{inter-c2p}}$ is defined by:

$$\mathcal{L}_{\text{inter-c2p}} = f_{\text{mse}}(\mathcal{D}_{\text{inter-c2p}}). \tag{4.14}$$

4.3.4 Differences with OCR, ACFNet and CPNet

Methods that are most closely related to ours are OCR [61], ACFNet [64] and CPNet [60], which all focus on better utilizing class-level features and differ on how to extract the class centers and context features. However, they all use a **simple concatenation** to fuse the original pixel feature and the complementary context feature. For example, OCR and ACFNet first produce a coarse segmentation, which is supervised by the GT mask with a categorical cross-entropy loss, and then use this predicted coarse mask to generate the (soft) class centers by weighted summing all the pixel features.

OCR then aggregates these class centers according to their similarity to the original pixel feature termed as "pixel-region relation", resulting in a "contextual feature". Slightly differently from OCR, ACFNet directly uses the probability (from the predicted coarse mask) to aggregate class center, obtaining a similar context feature[§] termed as "attentional class feature".

CPNet defines an affinity map, which is a binary map indicating if two spatial locations belong to the same class. Then, they use a sub-network to predict their ideal affinity map and use the soft version affinity map termed as "Context Prior Map" for feature aggregation, obtaining a class feature (center) and a context feature. Note that CPNet concatenates class feature, which is the updated pixel feature, and the context feature.

We also propose to utilize class-level contextual features. Instead of extracting and fusing pixel features with sub-networks, we propose three loss functions to directly regularize training and encourage the learned features to maintain certain desired properties. The approach is simple but more effective, thanking the direct supervision (validated in Tab. 4.5). Moreover, our class center estimate is more

[§]Some nonlinear transformation layers used to increase regression capability are omitted here.

accurate because we use the GT mask. This strategy largely reduces the complexity of the network and introduces no computational overhead during inference. Furthermore, it is compatible with all existing methods, including OCR, ACFNet and CPNet, demonstrating great generalization capability.

4.4 Experiments

4.4.1 Implementation

Training Settings. For both CAR and baselines, we apply the settings common to most works [65, 67, 27, 23, 72], including SyncBatchNorm, batch size = 16, weight decay (0.001), 0.01 initial LR, and poly learning decay with SGD during training. In addition, for the CNN backbones (*e.g.*, ResNet), we set *output stride* = 8 (see [4]). Training iteration is set to 30k iterations unless otherwise specified. For the thresholds in Eq. 4.6 and Eq. 4.13, we set $\epsilon_0 = 0.5$ and $\epsilon_1 = 0.25$.

Determinism and Reproducibility. Our implementations are based on the latest NVIDIA deterministic framework (2022), so that exactly the same results can be always reproduced with the same hardware (*e.g.*, same GPU types) and same training settings (including random seed). To demonstrate the effectiveness of our CAR with equal comparisons, we reproduced all the baselines that we compare, all conducted with exactly the same settings unless otherwise specified.

4.4.2 Experiments on Pascal Context

The Pascal Context [41] dataset is split into 4,998/5,105 for training/test set. We use its 59 semantic classes following the common practice [61, 67]. Unless otherwise specified, both baselines and CAR are trained on the training set with 30k iterations. The ablation studies are presented as below.

	Methods	$\mathcal{L}_{\text{intra-c2p}}$	$\mathcal{L}_{\text{inter-c2c}}$	$\mathcal{L}_{\text{inter-c2p}}$	A	mIOU (%)
R1	ResNet-50 + Self-Attention [53]	-	-			48.32
R2					✓	48.56
R3	+ CAR	✓				49.17
R4		✓	✓			49.79
R5		✓	✓	✓		50.01
R6		✓			✓	49.62
R7		✓	✓		✓	50.00
R8		✓	✓	✓	✓	**50.50**
S1	Swin-Tiny + UperNet [56]	-	-			49.62
S2					✓	49.82
S3	+ CAR	✓				49.01
S4		✓	✓			50.63
S5		✓	✓	✓		50.26
S6		✓			✓	49.62
S7		✓	✓		✓	50.58
S8		✓	✓	✓	✓	**50.78**

Table 4.1 : Ablation studies of adding CAR to different methods on Pascal Context dataset. All results are obtained with single scale test without flipping. “A” means replacing the 3×3 conv with 1×1 conv. CAR improves the performance of different types of backbones (CNN & Transformer) and head blocks (SA & Uper), showing that the proposed CAR generalizes well on different network architectures.

4.4.2.1 CAR on ResNet-50 + Self-Attention.

We firstly test the CAR with “ResNet-50 + Self-Attention” (w/o image-level block in [67]) to verify the effectiveness of the proposed loss functions, *i.e.*, $\mathcal{L}_{\text{intra-c2p}}$,

$\mathcal{L}_{\text{inter-c2c}}$, and $\mathcal{L}_{\text{inter-c2p}}$.

As shown in Tab. 4.1, using $\mathcal{L}_{\text{intra-c2p}}$ directly improves 1.30 mIOU (48.32 vs 49.62). Introducing $\mathcal{L}_{\text{inter-c2c}}$ and $\mathcal{L}_{\text{inter-c2p}}$ further improves 0.38 mIOU and 0.50 mIOU. Finally, with all three loss functions, the proposed CAR improves 2.18 mIOU from the regular ResNet-50 + Self-attention (48.32 vs 50.50).

4.4.2.2 CAR on Swin-Tiny + Uper.

"Swin-Tiny + Uper" is a totally different architecture from "ResNet-50 + Self-Attention [53]". Swin [35] is a recent Transformer-based backbone network. Uper [56] is based on the pyramid pooling modules (PPM) [69] and FPN [32], focusing on extracting multi-scale context information. Similarly, as shown in Tab. 4.1, after adding CAR, the performance of Swin-Tiny + Uper also increases by 1.16, showing that our CAR can generalize to different architectures well.

4.4.2.3 The Devil is In the Architecture's Detail.

We find it important to replace the leading 3×3 conv (in the original method) with 1×1 conv (Fig. 4.3B). For example, $\mathcal{L}_{\text{intra-c2p}}$ and $\mathcal{L}_{\text{inter-c2p}}$ did not improve the performance in Swin-Tiny + Uper (Row S3 vs S1, and S5 vs S4 in Tab. 4.1). A possible reason is that the network is trained to maximize the separation between different classes. However, if the two pixels lie on different sides of the segmentation boundary, a 3×3 conv will merge the pixel representations from different classes, making the proposed losses harder to optimize.

To keep simplicity and maximize generalization, we use the same network configurations for all the baseline methods. However, performance may be further improved with some minor dedicated modifications for each baseline when deploying our CAR. For example, decreasing the filter number to 256 for the last conv layer of ResNet-50 + Self-Attention + CAR results in a further improvement to 51.00

Methods	CAR	CAR (Moving Average)		
		0.8	0.9	0.99
ResNet-50 + Self-Attention	50.50	49.80(-0.70)	50.26(-0.24)	49.96(-0.54)
Swin-Tiny + UperNet	50.78	49.56(-1.22)	50.03(-0.75)	48.93(-1.85)

Table 4.2 : Ablation studies of adding moving average to CAR on Pascal Context. The decay rate stands for the effect of old class center.

mIOU (from 50.50). Replacing the conv layer after PPM (inside the Uper block, Fig. 4.3A3) from 3×3 to 1×1 in Swin-Tiny + UperNet boosts Swin (tiny & large) + UperNet + CAR by an extra 0.5-1.0 mIOU. We did not try to exhaustively search these variants since they did not generalize.

CAR using Moving Average. We also implemented a moving average version of CAR which tracks the class center $\boldsymbol{\mu}$ with moving average similar to BatchNorm. As shown in Tab. 4.2, we find this moving average version of CAR negatively impacts both ResNet-50 + Self-Attention and Swin-Tiny + Uper.

Methods	Baseline	CAR	
		Image Class Center	Batch Class Center
ResNet-50 + Self-Attention	48.32	49.78	50.50
Swin-Tiny + UperNet	49.62	49.45	50.78

Table 4.3 : Comparison of mIOUs (%) obtained when using the batch class center vs using the image class center in CAR.

4.4.2.4 Ablation studies on batch class center

In our experiments, we calculated the class centers using a *batch* to alleviate the negative impact of noisy images. Here, we investigate the impact of using the class center of each individual image for class-aware regularizations.

	$\mathcal{L}_{\text{intra-c2p}}$	$\mathcal{L}_{\text{inter-c2c}}$	$\mathcal{L}_{\text{inter-c2p}}$	A	Training	Inference
R-50	-	-			167.41	167.21
+ SA				✓	159.16	158.96
+ CAR	✓				167.65	167.21
	✓	✓			167.66	167.21
	✓	✓	✓		167.78	167.21
	✓			✓	159.40	158.96
	✓	✓		✓	159.41	158.96
	✓	✓	✓	✓	159.53	158.96

Table 4.4 : The computational cost (in GFLOPs) of the proposed CAR on a 513×513 image with an output stride of 8.

4.4.2.5 CAR on Different Baselines (Pascal Context).

After we have verified the effectiveness of each part of the proposed CAR, we then tested CAR on multiple well-known baselines. All of the baselines were reproduced under similar conditions (see Sect. 4.4.1). Experimental results shown in Tab. 4.5 demonstrate the generalizability of our CAR on different backbones and methods.

4.4.2.6 Computational Cost of CAR

The proposed CAR only affects training since it is a set of regularizations (see Fig. 4.3). Similar to other losses/regularizations, we do not need to calculate the

CAR terms during inference. Table 4.4 compares the FLOPs calculated with the TensorFlow analyzer using ResNet-50 as an example. It can be seen that our CAR is very lightweight and introduces little extra training computations since we only impose center-to-center and center-to-pixel regularizations.

4.4.2.7 Visualization of Class Dependency Maps.

In Fig. 4.5, we present the class dependency maps calculated on the complete Pascal Context *test* set, where every pixel stores the dot-product similarities between every two class centers. The maps indicate the inter-class dependency obtained with the standard ResNet-50 + Self-Attention and Swin-Tiny + UperNet, and the effect of applying our CAR. A hotter color means that the class has higher dependency on the corresponding class, and vice versa. Usually, each class should has high dependency on itself due to the properties of the dot-product similarity used in Eq 4.11. However, without CAR, it may also dependent on inter-class as well. According to Fig. 4.5 a1-a2, we can easily observe that the inter-class dependency has been significantly reduced with CAR on ResNet50 + Self-Attention. Fig. 4.5 b1-b2 show a similar trend when tested with different backbones and head blocks. This partially explains the reason why baselines with CAR generalize better on rarely seen class combinations (Figs. 4.1 and 4.4).

4.4.2.8 Visualization of Pixel-relation Maps.

In Fig. 4.4, we visualize the pixel-to-pixel relation energy map, based on the dot-product similarity between a red-dot marked pixel and other pixels, as well as the predicted results for different methods, for comparison. Examples are from Pascal Context test set. As we can see, with CAR supervision, the existing models focus better on objects themselves rather than other objects. Therefore, this reduces the possibility of the classification errors because of the class-dependency bias.

Methods	Backbone	mIOU(%)	
		Pascal Context	COCO-Stuff10K
FCN [37]	ResNet-50 [18]	47.72	34.10
FCN + CAR	ResNet-50 [18]	48.40(+0.68)	34.91(+0.81)§
FCN [37]	ResNet-101 [18]	50.93	35.93
FCN + CAR	ResNet-101 [18]	51.39(+0.49)	36.88(+0.95)§
DeepLabV3 [4]	ResNet-50 [18]	48.59	34.96
DeepLabV3 + CAR	ResNet-50 [18]	49.53(+0.94)	35.13(+0.17)
DeepLabV3 [4]	ResNet-101 [18]	51.69	36.92
DeepLabV3 + CAR	ResNet-101 [18]	52.58(+0.89)	37.39(+0.47)
Self-Attention [53]	ResNet-50 [18]	48.32	34.35
Self-Attention + CAR	ResNet-50 [18]	50.50(+2.18)	36.58(+2.23)§
Self-Attention [53]	ResNet-101 [18]	51.59	36.53
Self-Attention + CAR	ResNet-101 [18]	52.49(+0.9)	38.15(+1.62)
CCNet [23]	ResNet-50 [18]	49.15	35.10
CCNet + CAR	ResNet-50 [18]	49.56(+0.41)	36.39(+1.29)
CCNet [23]	ResNet-101 [18]	51.41	36.88
CCNet + CAR	ResNet-101 [18]	51.97(+0.56)	37.56(+0.68)
DANet [14]	ResNet-101 [18]	51.45	35.80
DANet + CAR	ResNet-101 [18]	52.57(+1.12)	37.47(+1.67)
CPNet [60]	ResNet-101 [18]	51.29	36.92
CPNet + CAR	ResNet-101 [18]	51.98(+0.69)	37.12(+0.20)§
OCR [61]	HRNet-W48 [52]	54.37	38.22
OCR + CAR	HRNet-W48 [52]	54.99(+0.62)	39.53(+1.31)
UperNet [56]	Swin-Tiny [35]	49.62	36.07
UperNet + CAR	Swin-Tiny [35]	50.78(+1.16)	36.63(+0.56)§
UperNet [56]	Swin-Large [35]	57.48	44.25
UperNet + CAR	Swin-Large [35]	58.97(+1.49)	44.88(+0.63)
CAA [22]	EfficientNet-B5 [40]	57.79	43.40
CAA + CAR	EfficientNet-B5 [40]	58.96(+1.17)	43.93(+0.53)

Table 4.5 : Ablation studies of adding CAR to different baselines on Pascal Context [41] and COCOStuff-10K [1]. We deterministically reproduced all the baselines with the same settings. All results are obtained with single-scale testing without flipping. CAR works very well in most existing methods. § means reducing the class-level threshold ϵ_0 from 0.5 to 0.25. We found it is sensitive for some model variants to handle a large number of class. Affinity loss [60] and Auxiliary loss [69] are applied on CPNet and OCR, respectively, since they highly rely on those losses.

4.4.2.9 Exceeding state-of-the-art (SOTA) in Pascal Context

The main motivation of our CAR is to utilize class-level information as regularizations during training to boost the performance of all existing methods. However, following the convention and also for readers who are interested, we compare with state-of-the-art methods in Tab. 4.6 regardless their architectures are related to ours or not. Since Swin [35] is not compatible with dilation, we use JPU [55] as the substitution to obtain features with output stride = 8. Uper contains an FPN [32] module that can obtain features with output stride = 4.

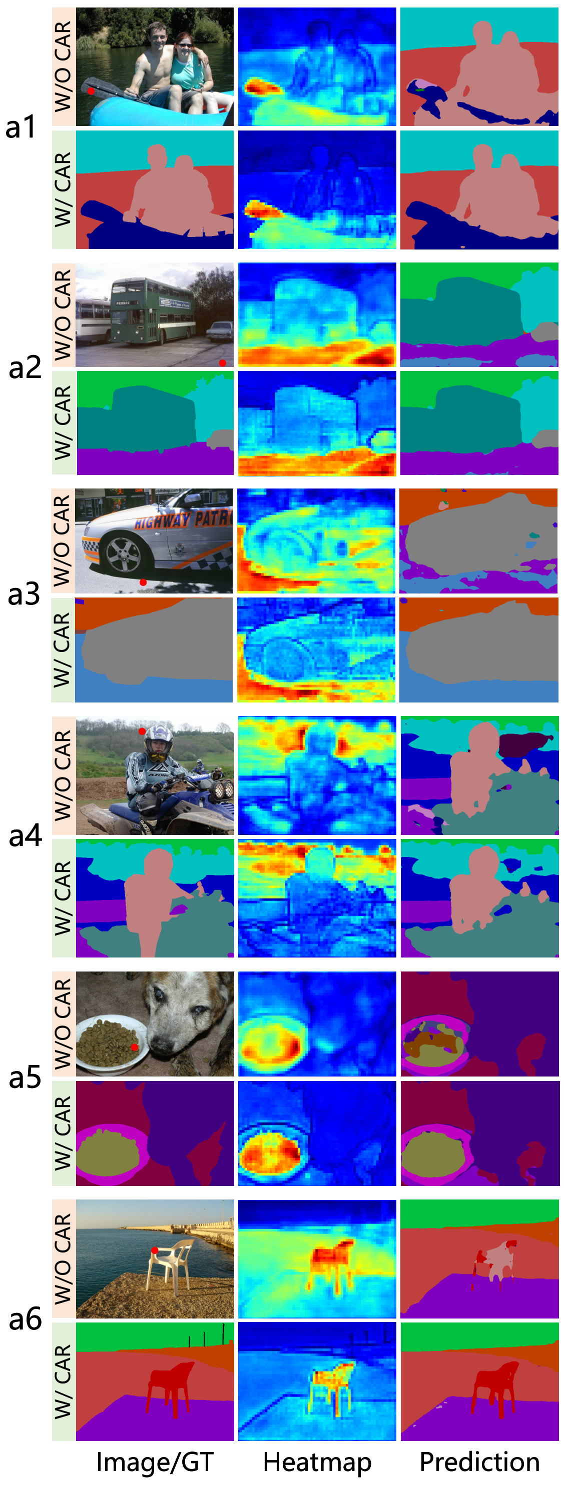
Boosted by our CAR, the strong model ConvNext-Large [36] + CAA [22] achieved the performance of 62.70% mIOU under single-scale testing, and 63.91% under multi-scale testing. Furthermore, increasing the training iterations from the default 30K to 40K when using Adam optimizer can further increase performance in Pascal Context dataset. Thus, the SOTA single model performance has now been boosted to 62.97% under single-scale testing, and 64.12% under multi-scale testing. This has outperformed the previous SOTA single model, *i.e.*, EfficientNetB7 + CAA, by a large margin.

4.4.3 Experiments on COCOStuff-10K

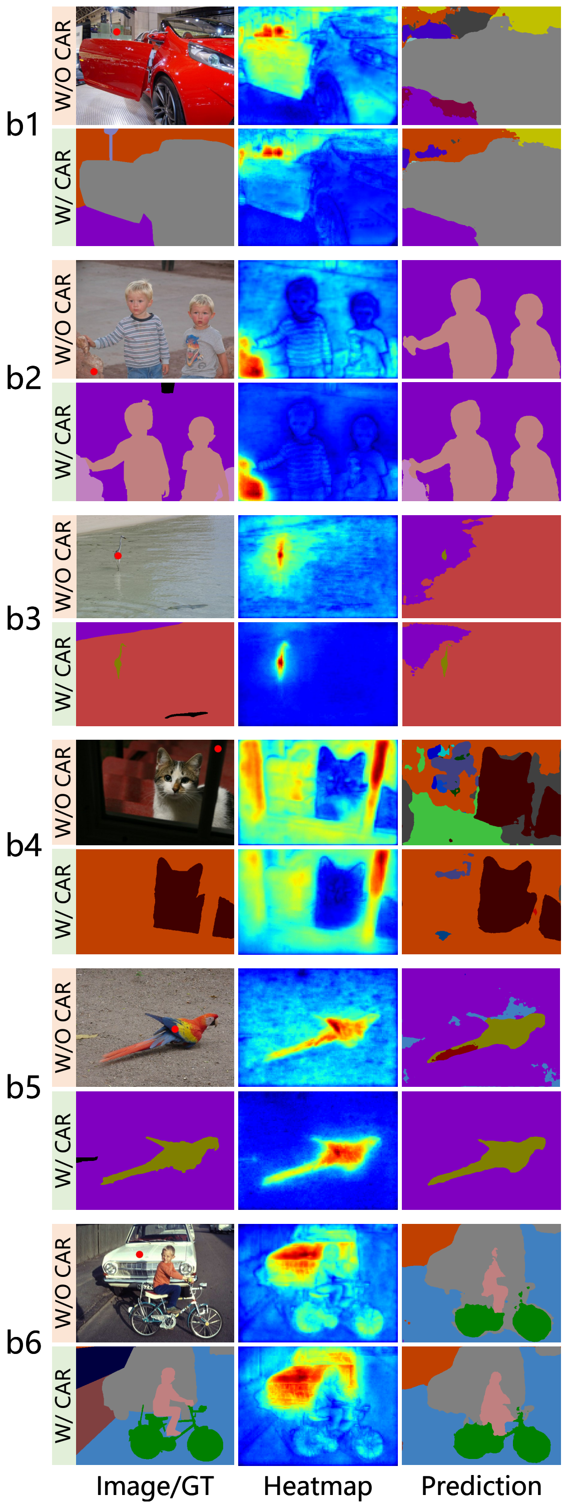
COCOStuff-10K dataset [1] is widely used for evaluating the robustness of semantic segmentation models [27, 61]. The COCOStuff-10k dataset is a very challenging dataset containing 171 labeled classes and 9000/1000 images for training/test.

4.4.3.1 CAR on Different Baselines (COCOStuff-10K).

As shown in Tab. 4.5, all of the tested baselines gain performance boost ranging from 0.17% to 2.23% with our proposed CAR. This demonstrates the generalization ability of our CAR when handling a large number of classes.



(a) ResNet50 + Self-Attention



(b) Swin-Tiny + UperNet

Figure 4.4 : Visualization of the feature similarity between a given pixel (marked with a red dot in the image) and all pixels, as well as the segmentation results on Pascal Context test set. A hotter color denotes larger similarity value. Apparently, our CAR reduces the inter-class dependency and exhibits better generalization ability, where energies are better restrained in the intra-class pixels.

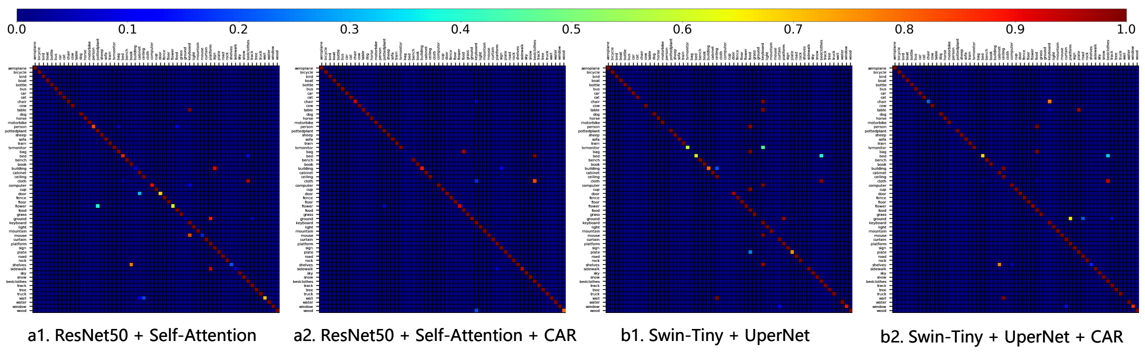


Figure 4.5 : Class dependency maps generated on Pascal Context test set. One may zoom in to see class names. A hotter color means that the class has higher dependency to the corresponding class, and vice versa. It is obvious that our CAR reduces the inter-class dependency, thus providing better generalizability (see Figs. 4.1 and 4.4).

4.4.3.2 Exceeding SOTA performance in COCOStuff-10K

Similar to Sect. 4.4.2.9, in Tab. 4.7, boosted by our CAR, the strong model ConvNext-Large [36] + CAA achieved the performance of 49.03% mIOU under single-scale testing, and 50.01% under multi-scale testing. This has also outperformed the previous SOTA single model, *i.e.*, EfficientNetB7 + CAA, by a large margin.

4.5 Extra Visualizations

4.5.1 Visualization of OCRNet on Pascal Context

Similar to the Sect. 4.4.2.8, in Fig. 4.6, we visualize the pixel-to-pixel relation energy maps obtained with HRNetW48 [61] + OCR [61]. This figure shows that our CAR can further improve the robustness of class center based models by making better use of the class center. Interestingly, as shown in C12 of Fig. 4.6 and Fig. 4.4 shown in Sect. 4.4.2.8 what is predicted by ResNet-50 + Self-Attention, we find that

Table 4.6 : Experiments on boosting the SOTA single-model performance on Pascal Context by our CAR. See Sect. 4.4.2.9 for the details. *§*: We report previous SOTA scores as reference. *SS*: Single scale without flipping. *MF*: Multi-scale with flipping. JPU is used to get features with output stride = 8. *Aux*: Apply auxiliary loss during training (see [69]). *Iterations*: training iterations.

Methods	Backbone	Aux	Optimizer	Iterations	SS mIOU(%)	MF mIOU(%)
CAA§	EfficientNet-B7-D8	✓	SGD	30K	-	60.30
UperNet	Swin-Large		SGD	30K	57.48	59.45
UperNet + CAR	Swin-Large		SGD	30K	58.97	60.76
CAA	Swin-Large + JPU		SGD	30K	58.31	59.75
CAA + CAR	Swin-Large + JPU		SGD	30K	59.84	61.46
CAA + CAR	Swin-Large + JPU		Adam	30K	60.68	62.21
CAA	ConvNeXt-Large + JPU		SGD	30K	60.48	61.80
CAA + CAR	ConvNeXt-Large + JPU		SGD	30K	61.40	62.69
CAA + CAR	ConvNeXt-Large + JPU		Adam	30K	62.65	63.77
CAA + CAR	ConvNeXt-Large + JPU	✓	Adam	30K	62.70	63.91
CAA + CAR	ConvNeXt-Large + JPU	✓	Adam	40K	**62.97**	**64.12**

cow/sheep/dog misclassification is a common issue in many semantic segmentation models, especially when *i.e.* grass and cow co-exist frequently during training. This issue is better addressed by our CAR due to its reduced inter-class dependency.

4.5.2 Visualization of DeepLab on Pascal Context

We also visualize the pixel-to-pixel relation energy map of ResNet-50 [18] + DeepLabV3 [4] in Fig. 4.7. These visualizations clearly show that the reduced inter-class dependency helps to correct the classification.

Table 4.7 : Experiments on boosting SOTA on COCOStuff10k, levering the previous single model SOTA and boosted by our CAR. See Sect. 4.4.3.2 for details. *§*: We report the original SOTA scores. *SS*: Single scale without flipping. *MF*: Multi-scale with flipping. *Aux* Apply auxiliary loss during training, see [69].

Methods	Backbone	Aux	Optimizer	SS mIOU(%)	MF mIOU(%)
CAA§	EfficientNet-B7-D8	✓	SGD	-	45.40
UperNet	Swin-Large		SGD	44.25	46.10
UperNet + CAR	Swin-Large		SGD	44.88	46.64
CAA	Swin-Large + JPU		SGD	44.22	45.31
CAA + CAR	Swin-Large + JPU		SGD	45.48	46.99
CAA	ConvNeXt-Large + JPU		SGD	46.49	47.23
CAA + CAR	ConvNeXt-Large + JPU		SGD	46.70	47.77
CAA + CAR	ConvNeXt-Large + JPU		Adam	48.20	48.83
CAA + CAR	ConvNeXt-Large + JPU	✓	Adam	**49.03**	**50.01**

4.6 Extra Technical Details

4.6.1 Deterministic

Control variables are very important for all scientific research. In computer vision, we always use the same backbones and the same datasets when verifying the difference between two methods.

Without using "deterministic" technology, all operations in neural networks contain some randomness. Nowadays, with the latest deterministic technology and fixed seeds, experiments can be conducted in a fully-controlled environment. This means that the performance difference between different settings (*i.e.*, w/ and w/o CAR) is not affected by this randomness any more but faithfully reflects the effectiveness of different methods.

In Tab. 4.8, we report the performance of our proposed CAR (ResNet-50 + Self-attention and Swin-Tiny + UperNet) with different seeds for readers who are interested in how our CAR performs when trained with different random seeds. As it is shown, our CAR consistently improves the mIOU over its baseline using different random seeds, demonstrating the effectiveness of the proposed CAR.

Methods	Seed (mIOU%)		
	0 (default)	1	2
ResNet-50 + Self-Attention	48.32	47.54	47.69
ResNet-50 + Self-Attention + CAR	50.50(+2.18)	50.20(+2.66)	50.59(+2.90)
Swin-Tiny + UperNet	49.62	49.24	49.54
Swin-Tiny + UperNet + CAR	50.78(+1.16)	50.57(+1.33)	50.75(+1.21)

Table 4.8 : Ablation studies of our proposed CAR using different random seeds on the Pascal Context dataset.

4.7 Limitation

CAR requires real ground-truth of semantic segmentation during training. Therefore, it cannot be directly applied on downstream tasks (*e.g.*, object detection, depth estimation), because those downstream tasks usually do not have real ground-truth of semantic segmentation. There are two workarounds to address this limitation:

- Pre-train the model of the downstream task on a segmentation dataset first, and then fine-tune the model on the downstream task.

- Predict a coarse prediction like OCR [61], and then use the coarse prediction result as the ground-truth for CAR to regularize the subsequent modules (*e.g.*, the downstream task).

However, both workarounds may have some negative impact on the performance. We leave this as one of future works to explore.

4.8 Summary and Future Work

In this chapter, we have aimed to make a better use of class level context information. We have proposed a universal class-aware regularizations (CAR) approach to regularize the training process and boost the differentiability of the learned pixel representations. To this end, we have proposed to minimize the intra-class feature variance and maximize the inter-class separation simultaneously. Experiments conducted on benchmark datasets with extensive ablation studies have validated the effectiveness of the proposed CAR approach, which has boosted the existing models' performance by up to 2.18% mIOU on Pascal Context and 2.23% on COCOStuff-10k with no extra inference overhead.

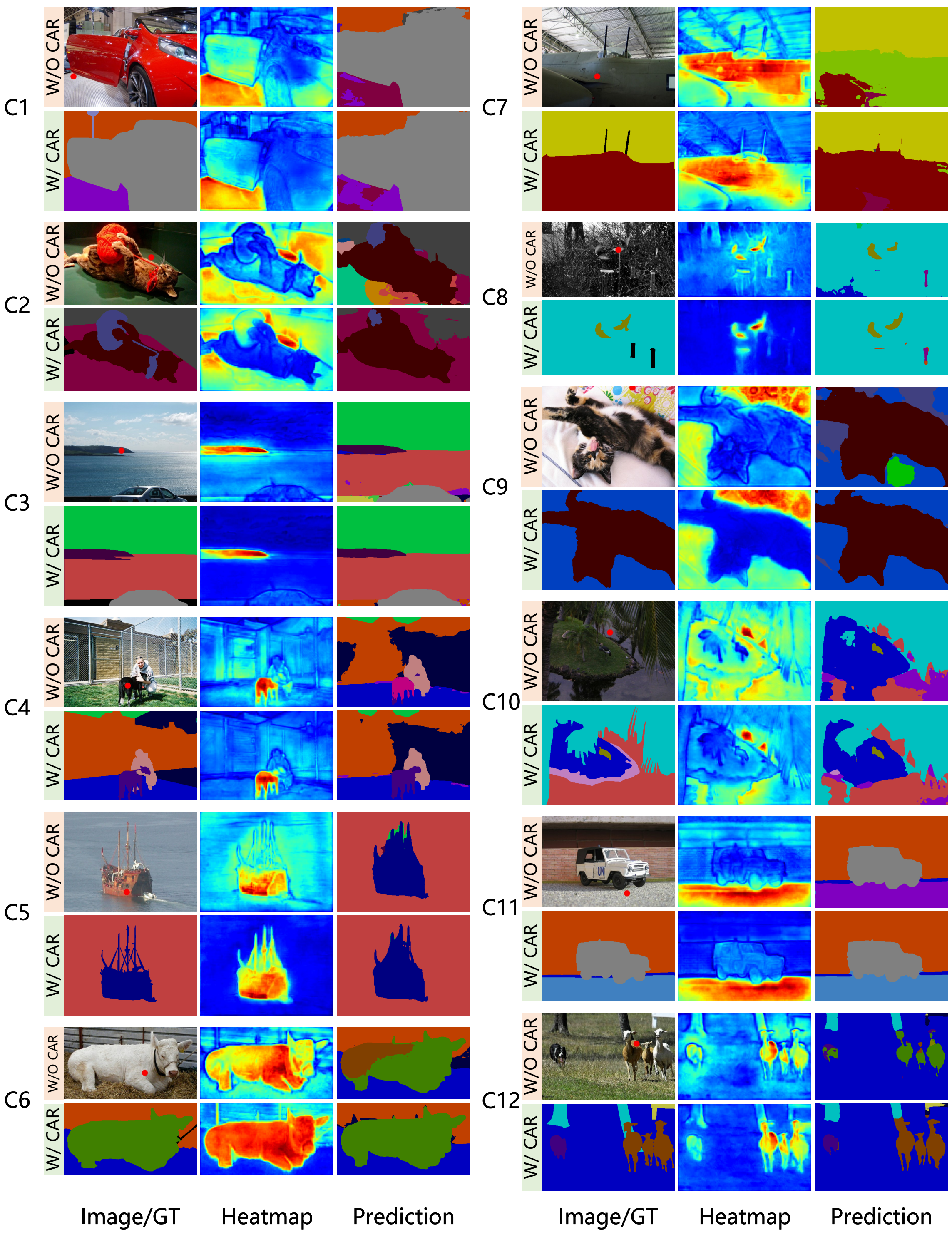


Figure 4.6 : Visualization of the feature similarity between a given pixel (marked with a red dot in the image) and all other pixels, as well as the segmentation results of **HRNetW48** [52] + **OCR** [61] on Pascal Context test set. A hotter color denotes a greater similarity value.

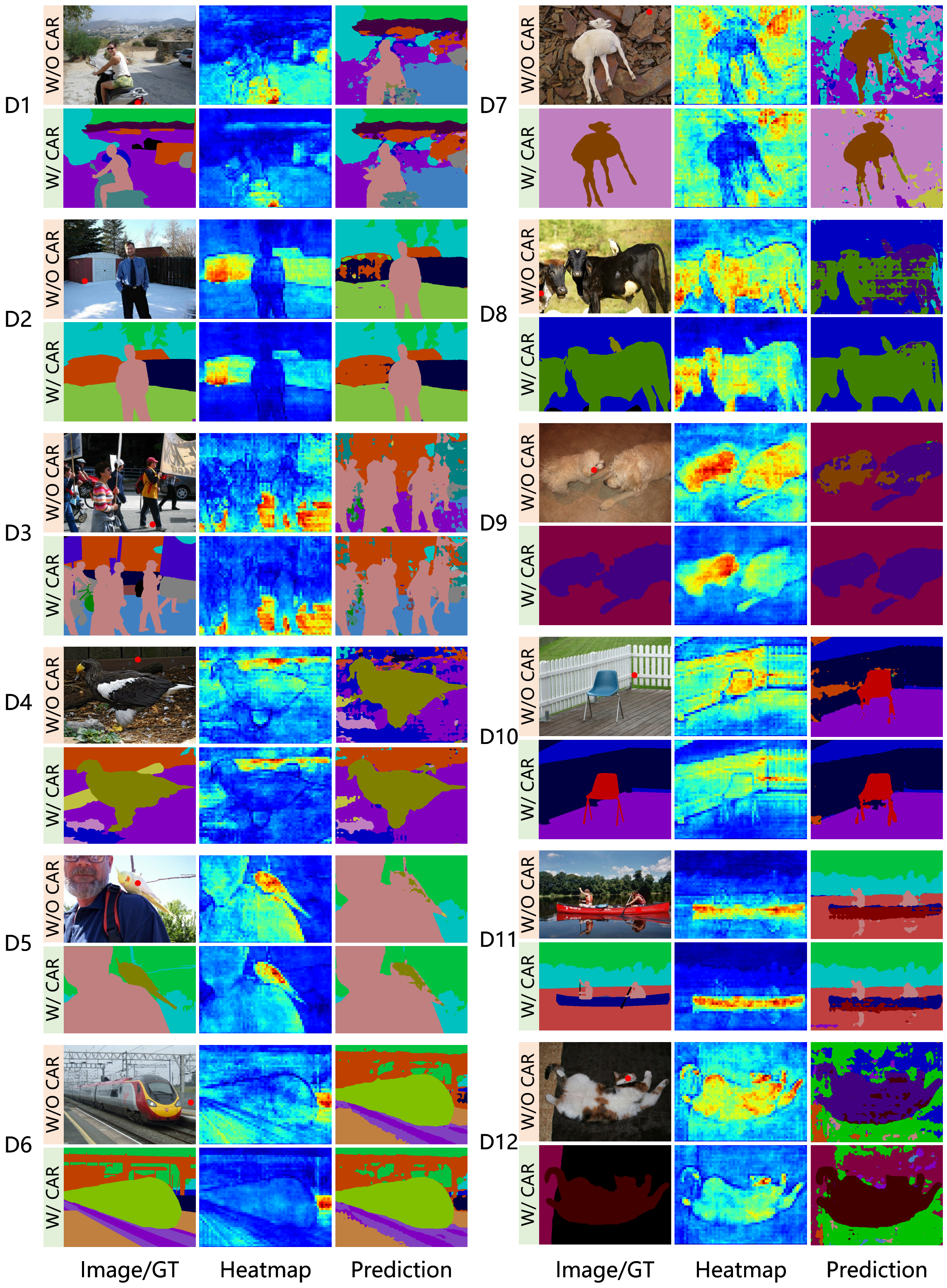


Figure 4.7 : Visualization of the feature similarity between a given pixel (marked with a red dot in the image) and all pixels, as well as the segmentation results of **ResNet-50** [18] + **DeepLab** [4] on Pascal Context test set. A hotter color denotes a greater similarity value.

Chapter 5

Summary and Discussion

5.1 Conclusion

In this thesis, we have focused on addressing the challenges in context information aggregation of image semantic segmentation. By considering the context information during the aggregation process to collect useful and effective information, new approaches have been proposed to enhance the pixel encoding and ensure the correctness of classification for semantic segmentation.

Specifically, the earlier multi-scale approaches, such as DeepLab, used multiple dilated convolutions with different dilation rates to aggregate multi-range information into the center pixel, so can cause diverse feature encoding for the same object, simply because it appears with a different scale. To address this scale variability problem, a Kernel-Sharing Atrous Convolution (KSAC) approach has been developed in this thesis. By sharing the same convolution kernel for different dilation rates, the same objects are always captured with a unified encoding, regardless of the scale it appears in the image. Moreover, the computational cost and GPU memory usage have also been substantially reduced since the convolution operation is only computed once.

As the improvement from multi-scale approaches to context aggregation, the attention mechanism enables a pixel to aggregate context information from all pixels in the spatial domain, weight it by its spatial relationships with all other pixels. It can also be applied to the channel domain, which models the channel-wise relation to enhance the encoding of a channel in the overall feature map. However, when both channel and spatial attention are performed together, feature conflicts can occur because features of different channels focus on different aspects of the images. To tackle the issue, a Channelized Axial Attention (CAA) has been proposed, and it considers channel relation within spatial attention. In other words, they can work seamlessly. Therefore, the conflicts are alleviated since both attentions are no longer computed separately. The newly proposed CAA stands in the 1st position in the Pascal Context dataset and COCOStuff-10K dataset for nearly a year.

Besides only modeling the encoding relations of independent images, class centers directly bring the unified encoding of each class across the whole dataset. However, existing methods have only focused on the extraction of the class center from the feature map and have simply merged or concatenated it with the original feature map to perform the final prediction, so their effectiveness has bee dramatically decreased since the extracted class center may be overridden by the original feature. Moreover, the class center in those works is extracted based on the coarse feature map and coarse predictions, so the error accumulation problem can occur. In our Class-Aware Regularization (CAR) approach, the class center is computed from real ground truth during training to get as accurate class encoding as possible. Instead of focusing on the class-center extraction process, the proposed CAR uses class-center to achieve three targets, *i.e.*, 1) reducing pixels' intra-class distance, 2) reducing inter-class center-to-center dependency, and 3) reducing pixels' inter-class dependency. By using CAR to regularize the existing neural networks and including the networks w/ or w/o class center designs, they have minimized the intra-class feature variance and maximized the inter-class feature separation automatically. With the help of CAR, the CAA again stands in the 1st position in the Pascal Context at the time when this thesis is submitted.

All three works completed during this PhD project have resolved the critical issues associated with the existing context aggregation works, increased the accuracy of the context encoding, and boosted the performance of semantic segmentation to a new level with reduced or negligible computational overhead.

5.2 Future Work

In this section, a few directions that may further improve the robustness of context-aware image semantic segmentation are suggested.

5.2.1 Mining Inter-class Relations with Inter-class Dependent Encodings

In CAR, the importance of keeping the inter-class features separated so as to reduce the dependency between inter-class objects in both CNN and Transformer encoder has already been verified (see the Experiment sections of Chapter 4 for details). Based on our observation, pixel feature extraction should be strictly inter-class independent so as to provide zero-dependency pixel encoding.

However, it is hard to deny that inter-class relations may help improve the accuracy of semantic segmentation, especially for imbalanced datasets or when there are strong reasonable connections between some classes. Another reason that CAR can achieve success after removing the inter-class reasoning is because of the weak inter-class and intra-class modeling of the existing networks. In other words, without the regularization from CAR, the existing networks can easily rely on the inter-class encoding before fully mining intra-class encoding, whereas the intra-class encoding may become hard to learn due to the dataset imbalance but should be carefully considered in pixel encoding.

After the encoding of a class-independent pixel is extracted (the intra-class context is fully mined), it is now safe to perform inter-class modeling to enhance the class encoding. One of the possible ways to model the inter-class relations is to use the knowledge graph model to predict the low confidence pixels.

5.2.2 Decouple Upsampling Encoding

Because of the hardware limitation, major neural networks have to process a downsampled feature map, and it is required to upsample to the original input size. Bilinear interpolation has been used for feature map upsampling in semantic segmentation for many years since it has replaced the transposed convolution because of the "checkerboard effects". Bilinear interpolation works very well in many scenarios. Many works have also been proposed to improve the interpolation, they are usually called "refinement", and can be categorized into "low-level feature fusion" (e.g., UNet [45], FPN [32]) or "boundary supervision" [63].

However, there are some details that many works are not aware. A plain bilinear interpolation without "boundary supervision" can already bring many "good-looking" boundaries in up-sampled prediction, even in some complex images, though it still has many failure cases, leading to the motivation of boundary supervision. This means, by considering the algorithm of bilinear interpolation, the pixel encoding of the low-resolution feature map has a significant effect on assisting the interpolation process.

It also means that, given a limited encoding space (limited size of the channel domain), especially in the boundary pixels, it is hard to reach to class center (*e.g.*, regularized by CAR) because many of channels are taken to consider interpolation. It may even become an issue because a boundary pixel may need to consider 64 missing pixels (output stride = 8) in order to achieve the class optimization target.

Obviously, disentangling class encoding and interpolation encoding from pixel encoding, and allowing them to be optimized by different targets is necessary to avoid this issue. However, it is very hard to perform disentanglement since lacking the theory and methodology to understand the effects of each channel in hidden layers.

Bibliography

[1] H. Caesar, J. Uijlings, and V. Ferrari, "Coco-stuff: Thing and stuff classes in context," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[2] N. Carion, F. Massa, G. Synnaeve, N. Usunier, and S. Z. Alexander Kirillov, "End-to-end object detection with transformers," in *European Conference on Computer Vision*, 2020.

[3] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[4] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 2017.

[5] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *European Conference on Computer Vision*, 2018.

[6] B. Cheng, M. D. Collins, Y. Zhu, T. Liu, T. S. Huang, H. Adam, and L.-C. Chen, "Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[7] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar, "Masked-

attention mask transformer for universal image segmentation," 2022.

[8] B. Cheng, A. G. Schwing, and A. Kirillov, "Per-pixel classification is not all you need for semantic segmentation," 2021.

[9] S. Choi, J. T. Kim, and J. Choo, "Cars can't fly up in the sky: Improving urban-scene segmentation via height-driven attention networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[10] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[11] H. Ding, X. Jiang, B. Shuai, A. Q. Liu, and G. Wang, "Semantic correlation promoted shape-variant context for segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[12] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[13] M. Everingham, L. V. Gool, C. K.l.Wiliams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, 2009.

[14] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu, "Dual attention network for scene segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[15] J. Fu, J. Liu, Y. Wang, Y. Li, Y. Bao, J. Tang, and H. Lu, "Adaptive context network for scene parsing," in *International Conference on Computer Vision*, 2019.

[16] J. Fu, J. Liu, Y. Wang, J. Zhou, C. Wang, and H. Lu, "Stacked deconvolutional network for semantic segmentation," *IEEE Transaction on Image Processing*, pp. 1–1, 2019.

[17] B. Hariharan, P. Arbelaez, L. Bourdev, S. Maji, and J. Malik, "Semantic contours from inverse detectors," in *International Conference on Computer Vision*, 2011.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[19] J. Ho, N. Kalchbrenner, D. Weissenborn, and T. Salimans, "Axial attention in multidimensional transformers," 2019.

[20] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[21] Y. Huang, D. Kang, L. Chen, X. Zhe, W. Jia, L. Bao, and X. He, "Car: Class-aware regularizations for semantic segmentation," in *European Conference on Computer Vision*, 2022.

[22] Y. Huang, D. Kang, W. Jia, X. He, and L. liu, "Channelized axial attention - considering channel relation within spatial attention for semantic segmentation," in *AAAI*, 2022.

[23] Z. Huang, X. Wang, Y. Wei, L. Huang, H. Shi, W. Liu, and T. S. Huang, "Ccnet: Criss-cross attention for semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[24] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollar, "Panoptic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Conference on Neural Information Processing Systems*, 2012.

[26] X. Li, Y. Yang, Q. Zhao, T. Shen, Z. Lin, and H. Liu, “Spatial pyramid based graph reasoning for semantic segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[27] X. Li, Z. Zhong, J. Wu, Y. Yang, Z. Lin, and H. Liu, “Expectation-maximization attention networks for semantic segmentation,” in *International Conference on Computer Vision*, 2019.

[28] X. Li, X. Li, L. Zhang, C. Guangliang, J. Shi, Z. Lin, Y. Tong, and S. Tan, “Improving semantic segmentation via decoupled body and edge supervision,” in *European Conference on Computer Vision*, 2020.

[29] Z. Li, Y. Sun, and J. Tang, “Ctnet: Context-based tandem network for semantic segmentation,” *arXiv preprint arXiv:2104.09805*, 2021.

[30] X. Liang, H. Zhou, and E. Xing, “Dynamic-structured semantic propagation network,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[31] G. Lin, A. Milan, C. Shen, and I. Reid, “Refinenet: Multi-path refinement networks for high-resolution semantic segmentation.” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[32] T.-Y. Lin, P. Dollá, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[33] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. Yuille, and L. Fei-Fei,

"Auto-deeplab: hierarchical neural architecture search for semantic image segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[34] M. Liu, D. Schonfeld, and W. Tang, "Exploit visual dependency relations for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[35] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *ICCV*, 2021.

[36] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2022.

[37] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[38] C. Marius, O. Mohamed, R. Sebastian, R. Timo, E. Markus, B. Rodrigo, F. Uwe, S. Roth, and S. Bernt, "The cityscapes dataset for semantic urban scene understanding," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[39] A. L. Mass, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *International Conference on Machine Learning*, 2013.

[40] T. Mingxing and L. Quoc, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*, 2019.

[41] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, “The role of context for object detection and semantic segmentation in the wild,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[42] P. Ramachandran, N. Parmar, A. Vaswani, I. Bello, A. Levskaya, and J. Shlens, “Stand-alone self-attention in vision models,” in *Conference on Neural Information Processing Systems*, 2019.

[43] R. Ranftl, A. Bochkovskiy, and V. Koltun, “Vision transformers for dense prediction,” in *ICCV*, 2021.

[44] S. Robin, G. Ricardo, L. Ivan, and S. Cordelia, “Segmenter: Transformer for semantic segmentation,” in *ICCV*, 2021.

[45] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *MICCAI*, 2015.

[46] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[47] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *International Conference on Learning Representations*, 2015.

[48] Z. Sixiao, L. Jiachen, Z. Hengshuang, Z. Xiatian, L. Zekun, W. Yabiao, F. Yanwei, F. Jianfeng, X. Tao, T. P. H.S., and Z. Li, “Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[49] T. Takikawa, D. Acuna, V. Jampani, and S. Fidler, “Gated-scnn: Gated shape

cnns for semantic segmentation," in *International Conference on Computer Vision*, 2019.

[50] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Łukasz Kaiser, and I. Polosukhin, "Attention is all you need," in *Conference on Neural Information Processing Systems*, 2017.

[51] H. Wang, Y. Zhu, H. Adam, A. Yuille, and L.-C. Chen, "Max-deeplab: End-to-end panoptic segmentation with mask transformers," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[52] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao, "Deep high-resolution representation learning for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[53] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[54] Y. Wen1, K. Zhang, Z. Li, and Y. Qiao, "Discriminative feature learning approach for deep face recognition," in *European Conference on Computer Vision*, 2016.

[55] H. Wu, J. Zhang, K. Huang, K. Liang, and Y. Yizhou, "Fastfcn: Rethinking dilated convolution in the backbone for semantic segmentation," 2019.

[56] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun, "Unified perceptual parsing for scene understanding," in *European Conference on Computer Vision*, 2018.

[57] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "Segformer: Simple and efficient design for semantic segmentation with transformers," *arXiv preprint arXiv:2105.15203*, 2021.

[58] M. Yang, K. Yu, C. Zhang, Z. Li, and K. Yang, “Denseaspp for semantic segmentation in street scenes,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[59] T.-J. Yang, M. D, Collins, Y. Zhu, J.-J. Hwang, X. Z. Ting Liu, V. Sze, G. Papandreou, and L.-C. Chen, “Deeperlab: Single-shot image parser,” in *arXiv preprint arXiv:1902.05093, 2019*, 2019.

[60] C. Yu, J. Wang, C. Gao, G. Yu, C. Shen, and N. Sang, “Context prior for scene segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[61] Y. Yuan, X. Chen, and J. Wang, “Object-contextual representations for semantic segmentation,” in *European Conference on Computer Vision*, 2020.

[62] Y. Yuan, L. Huang, J. Guo, C. Zhang, X. Chen, and J. Wang, “Ocnet: Object context network for scene parsing,” *International Journal of Computer Vision*, 2021.

[63] Y. Yuan, J. Xie, X. Chen, and J. Wang, “Segfix: Model-agnostic boundary refinement for segmentation,” in *European Conference on Computer Vision*, 2020.

[64] F. Zhang, Y. Chen, Z. Li, Z. Hong, J. Liu, F. Ma, J. Han, and E. Ding, “Acfnet: Attentional class feature network for semantic segmentation,” in *International Conference on Computer Vision*, 2019.

[65] H. Zhang, K. Dana, J. Shi, Z. Zhang, X. Wang, A. Tyagi, and A. Agrawal, “Context encoding for semantic segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[66] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, Z. Zhang, H. Lin, Y. Sun, T. He, J. Muller,

R. Manmatha, M. Li, and A. Smola, “Resnest: Split-attention networks,” *arXiv preprint arXiv:2004.08955*, 2020.

[67] H. Zhang, H. Zhan, C. Wang, and J. Xie, “Semantic correlation promoted shape-variant context for segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[68] Z. Zhang, X. Zhang, C. Peng, X. Xue, and J. Sun, “Exfuse: enhancing feature fusion for semantic segmentation,” in *European Conference on Computer Vision*, 2018.

[69] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, “Pyramid scene parsing network,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[70] Z. Zhong, Z. Q. Lin, R. Bidart, X. Hu, I. B. Daya, and Z. Li, “Squeeze-and-attention networks for semantic segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[71] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, “Scene parsing through ade20k dataset.” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[72] Z. Zhu, M. Xu, S. Bai, T. Huang, and X. Bai, “Asymmetric non-local neural networks for semantic segmentation,” in *International Conference on Computer Vision*, 2019.