Channelized Axial Attention – Considering Channel Relation within Spatial Attention for Semantic Segmentation

Ye Huang 1  Di Kang 2  Wenjing Jia 1 * Xiangjian He 1  Liu Liu 3

1 University of Technology Sydney  2 Tencent AI LAB  3 University of Sydney
edward.ye.huang@qq.com

Abstract

Spatial and channel attentions, modelling the semantic interdependencies in spatial and channel dimensions respectively, have recently been widely used for semantic segmentation. However, computing spatial and channel attentions separately sometimes causes errors, especially for those difficult cases. In this paper, we propose Channelized Axial Attention (CAA) to seamlessly integrate channel attention and spatial attention into a single operation with negligible computation overhead. Specifically, we break down the dot-product operation of the spatial attention into two parts and insert channel relation in between, allowing for independently optimized channel attention on each spatial location. We further develop grouped vectorization, which allows our model to run with very little memory consumption without slowing down the running speed. Comparative experiments conducted on multiple benchmark datasets, including Cityscapes, PASCAL Context, and COCO-Stuff, demonstrate that our CAA outperforms many state-of-the-art segmentation models (including dual attention) on all tested datasets.

1 Introduction

Semantic segmentation is a fundamental task in many computer vision applications, which assigns a class label to each pixel in the image. Most of the existing approaches (Yuan, Chen, and Wang 2020; Yang et al. 2018; Fu et al. 2019; Li et al. 2019) have adopted a pipeline similar to the one that is defined by Fully Convolutional Networks (FCNs) (Long, Shelhamer, and Darrell 2015) 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 (Zhao et al. 2017) and DeepLab (Chen et al. 2018), 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 attempts in this direction included Squeeze and Excitation Networks (SE Nets) (Hu, Shen, and Sun 2018) 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 (Wang et al. 2018; Vaswani et al. 2017) 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., (Fu et al. 2019)) 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. 1(a)). Although they have achieved improved 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. 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. 5, 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 (Wang et al. 2018), 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 paper include:

• 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 (Everingham et al. 2009), COCO-Stuff (Caesar, Uijlings, and Ferrari 2018) and Cityscapes (Marius et al. 2016), demonstrate the superiority of our approach over the state-of-the-art approaches.

2 Related Work

Spatial attention. Non-local networks (Wang et al. 2018) and Transformer (Vaswani et al. 2017) 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 (Chen et al. 2017; Zhao et al. 2017), 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 (Huang et al. 2020; Zhu et al. 2019) have been developed to optimize the GPU memory costs of spatial self-attention.

Channel Attention. Channel attention (Hu, Shen, and Sun 2018) examined the relationships between channels, and enhanced the important channels so as to improve performance. SENets (Hu, Shen, and Sun 2018) 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 (Fu et al. 2019), channel-wise relationships were modelled by self-attention, similar to the it is 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 (Fu et al. 2019), 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. 1(a)) can cause conflicting results, and thus weaken their performance when both results are summarized together. Similarly, in the cascaded model (see Fig. 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.

3 Exploring Conflicting Features

As we have analyzed earlier in Sect. 2, 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.

3.1 Visualizing Conflicts

For a parallel dual attention design such as DANet (Fu et al. 2019), since it has two auxiliary losses for each of spatial attention and channel attention, we directly use their 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 Examples of Conflicting Features

To visualize the impact of the feature conflicting issue in the existing dual attention designs (see Sect. 2), we present examples of the segmentation results obtained with the conflicting features in the parallel dual attention design (see Fig. 3) and the sequential dual attention design (see Fig. 4). As observed from Fig. 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. 4, the pixel representation obtained from the spatial attention ignores the correct representation obtained from the channel attention, and worsens the prediction.

4 Methods

4.1 Preliminaries

Formulation of the Spatial Self-attention

Following Non Local (Wang et al. 2018) and Stand Alone Self Attention (Ramachandran et al. 2019), a 2D self-attention operation in spatial domain can be defined by:

$$y_{i,j} = \sum_{m,n} f(x_{i,j}, x_{m,n})g(x_{m,n}).$$

(1)

Here, a pairwise function $f$ computes the similarity between the pixel representation $x_{i,j}$ (query) at position $(i, j)$ and the pixel representation $x_{m,n}$ (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 (Wang et al. 2018) as $f$, i.e.,

$$f(x_{i,j}, x_{m,n}) = \text{softmax}_{m,n}(\theta(x_{i,j})^T \theta(x_{m,n})), \quad (2)$$

where $\theta$ is a $1 \times 1$ convolution layer transforming the feature maps $x$ to a new domain to calculate dot-product similarity (Wang et al. 2018) between every two pixels. Note that, following a common practice (Li et al. 2020), we use the same $1 \times 1$ convolution weights for both query and key. Then, these similarities are used as the weights (Eq. (1)) to aggregate features of all pixels, producing an enhanced pixel representation $y_{i,j}$ at position $(i, j)$.
Formulation of the Axial Attention. From the above equations, we can see the computational complexity of the self-attention module is \( O(H^4W^2) \), which requires large computation resources and prevents real-time applications, such as autopilot. Several subsequent works (Huang et al. 2020; Ho et al. 2019) 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. 2, we define axial similarity functions by:

\[
A_{\text{col}}(x_{i,j}, x_{m,j}) = \text{softmax}_m \left( \theta(x_{i,j})^T \theta(x_{m,j}) \right), \quad m \in [H] \quad (3) \\
A_{\text{row}}(x_{i,j}, x_{i,n}) = \text{softmax}_n \left( \phi(x_{i,j})^T \phi(x_{i,n}) \right), \quad n \in [W]. \quad (4)
\]

Note that we use different feature transformations (\( \theta, \phi \)) for the row and column attention calculations.

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

\[
y_{i,j} = \sum_{m,n} \left( A_{\text{row}}(x_{i,j}, x_{i,n}) \sum_{m} A_{\text{col}}(x_{i,j}, x_{m,j})g(x_{m,n}) \right) \quad (5)
\]

4.2 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.

Thus, we propose to compute channel attention within spatial attention. Specifically, we firstly break down spatial attention into more basic parts (Sect. 4.2). Then, spatially varying channel attention is generated with \( \alpha_{i,j,m,n} \) and \( \beta_{i,j,n} \). 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.

Breaking Down Axial Attention. For convenience, we firstly define two variables \( \alpha_{i,j,m,n} \) and \( \beta_{i,j,n} \) to represent the intermediate weighted features as follows:

\[
\alpha_{i,j,m,n} = A_{\text{col}}(x_{i,j}, x_{m,j})g(x_{m,n}) \quad (6) \\
\beta_{i,j,n} = A_{\text{row}}(x_{i,j}, x_{i,n}) \sum_{m} \alpha_{i,j,m,n}. \quad (7)
\]

Thus, Eq. (5) can be rewritten as:

\[
y_{i,j} = \sum_{m,n} \beta_{i,j,n} = \sum_{m,n} A_{\text{row}}(x_{i,j}, x_{i,n}) \left( \sum_{m} \alpha_{i,j,m,n} \right). \quad (8)
\]

Eqs. (6), (7) and (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. (6) and row attention as in Eq. (7), and 2) Summation: summing the elements along row and column axes in Eq. (8). Note that, this breakdown is also applicable to regular self-attention (see Table 3 and Appendix).

Spatially Varying Channel Attention. With the intermediate results \( \alpha_{i,j,m,n} \) and \( \beta_{i,j,n} \) in Eqs. (6) and (7), channel relation can be applied inside spatial attention, seamlessly.

---

Figure 5: The detailed architecture of the proposed CAA (Row Attention). We present the way to apply channel attention seamlessly in (b). We mark the independent spatial dimension in **bold** style. This allows channel attention to also consider spatial unique information. *Note that,* in our design, the “value” for row attention is obtained from the result of column attention. See Eq. 11 for details.
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 \( C_{\text{col}} \) and \( C_{\text{row}} \) are written as:

\[
C_{\text{col}}(\alpha_{i,j,m,n}) = \text{Sigmod} \left( \frac{\text{ReLU}(\sum_{v=m}^{H} \alpha_{i,j,v,m,n})}{H \times W} \omega_{1,c} \right) \alpha_{i,j,m,n},
\]

\[
C_{\text{row}}(\beta_{i,j,n}) = \text{Sigmod} \left( \frac{\text{ReLU}(\sum_{v=n}^{W} \beta_{i,j,v,n})}{H \times W} \omega_{2,c} \right) \beta_{i,j,n},
\]

where \( \text{Sigmod}(\cdot) \) is the learned channel attention, and \( \omega_{1,c} \), \( \omega_{2,c} \) are the learned relationships between different channels according to \( \alpha_{i,j,m,n} \) and \( \beta_{i,j,n} \).

Thus, instead of directly using \( \alpha_{i,j,m,n} \) and \( \beta_{i,j,n} \) as in Eq. (8), for each column and row, we obtain the channelized axial attention features, where the intermediate results \( \alpha_{i,j,m,n} \) and \( \beta_{i,j,n} \) are weighted by the spatially varying channel attention defined in Eqs. (9) and (10) as:

\[
y_{i,j} = \sum_{n} C_{\text{row}}(\beta_{i,j,n}) \sum_{m} C_{\text{col}}(\alpha_{i,j,m,n}) .
\]

Note that the spatially varying channel attention keeps a \( W \) dimension after averaging \( H \times W \) during the channel attention (Fig. 5). Now each row has its own channel attention thanks to the breaking down of spatial axial attention.

### 5 Experiments

#### 5.1 Implementation Details

**Architecture** Our network is built on ResNet-101 (He et al. 2016) pre-trained on ImageNet, and then followed by our CAA module (or axial + SE for ablation study), as well as the image-level block and auxiliary head in (Zhang et al. 2019; Fu et al. 2019). The original ResNet results in a feature map of \( 1/32 \) of the input size. Following other works (Chen et al. 2018; Li et al. 2019), 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.

**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.

**Training Settings** We employ stochastic gradient descent (SGD) for optimization, where the polynomial decay learning rate policy \((1 - \frac{\text{iter}}{\text{max iter}})^{0.9}\) 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, we only apply the most basic data augmentation strategies as in (Chen et al. 2018), including random flip, random scale and random crop.

#### 5.2 Results on PASCAL Context

The PASCAL Context (Mottaghi et al. 2014) dataset has 59 classes with 4,989 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.
### Table 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. 5.2).

<table>
<thead>
<tr>
<th>Layer Counts</th>
<th># of Channels</th>
<th>mIOU (%)</th>
<th>FLOPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>50.27±0.2</td>
<td>68.7G</td>
</tr>
<tr>
<td>1</td>
<td>128</td>
<td>50.75±0.2</td>
<td>+0.00024G</td>
</tr>
<tr>
<td>3</td>
<td>128</td>
<td>50.85±0.2</td>
<td>+0.00027G</td>
</tr>
<tr>
<td>5</td>
<td>128</td>
<td>51.06±0.2</td>
<td>+0.00030G</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>50.40±0.3</td>
<td>+0.00043G</td>
</tr>
<tr>
<td>5</td>
<td>64</td>
<td>50.12±0.2</td>
<td>+0.00015G</td>
</tr>
<tr>
<td>5</td>
<td>256</td>
<td>50.35±0.4</td>
<td>+0.00098G</td>
</tr>
</tbody>
</table>

Table 2: Result comparison between axial attention, axial attention + SE and channelized axial attention.

<table>
<thead>
<tr>
<th>Axial Attention</th>
<th>+ SE</th>
<th>+ Our Channelization</th>
</tr>
</thead>
<tbody>
<tr>
<td>50.27±0.2</td>
<td>50.37±0.2</td>
<td>51.06±0.2</td>
</tr>
</tbody>
</table>

### Effectiveness of the Proposed Channelization

We first report the impact of adding channelized axial attention and with different depth and width in Table 1, where ‘-’ for the baseline result indicates no channelization is performed.

As can be seen from Table 2, our proposed channelization improves the mIOU over the baseline regardless of the layer counts and the number of channels used. In particular, the best performance is achieved when the Layer Counts = 5 and the number of Channels = 128.

We also compare our model with the sequential design of “Axial Attention + SE”, as shown in Table 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 5, results obtained with other backbones are provided to demonstrate the effectiveness and robustness of CAA.

<table>
<thead>
<tr>
<th>Layer Counts</th>
<th># of Channels</th>
<th>mIOU (%)</th>
<th>FLOPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>128</td>
<td>50.75±0.2</td>
<td>+0.00024G</td>
</tr>
<tr>
<td>3</td>
<td>128</td>
<td>50.85±0.2</td>
<td>+0.00027G</td>
</tr>
<tr>
<td>5</td>
<td>128</td>
<td>51.06±0.2</td>
<td>+0.00030G</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>50.40±0.3</td>
<td>+0.00043G</td>
</tr>
</tbody>
</table>

### 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 with ResNet-101. Our channelized method can also further improve the performance of self-attention by 0.67% (51.09% vs 50.42%).

### Impact of the Testing Strategies

We compare the performance and computation cost of our proposed model against the baseline and DANet (Fu et al. 2019) with different testing strategies in Table 4. Using the same settings as in other works (Fu et al. 2019), 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.

### Comparison with the State-of-the-art

Finally, in Table 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 the main paper. More results using alternative backbones are included in Table 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. 6, we show some results obtained by our CAA model, FCN, and Dual attention. Our model is able to handle previous failure cases better, especially when some class A covering only a smaller region is surrounded by another class B covering a much larger region.

### 5.3 Results on the COCO-Stuff 10K

Following the other works (Fu et al. 2019), we evaluate our CAA on COCO-Stuff 10K dataset (Caesar, Uijlings, and Ferrari 2018), which contains 9,000 training images and 1,000 testing images with 171 classes. Our model is trained for 40k iterations. As shown in Table 7, our proposed CAA outperforms all other state-of-the-art approaches by a large margin of 1.3%, demonstrating that our model can better
Table 5: Ablation study of other backbones. All results are obtained in single scale without flipping. OS: Output strides during evaluation. AA: Axial Attention. C: Channelized.

<table>
<thead>
<tr>
<th>Backbone</th>
<th>OS</th>
<th>AA</th>
<th>C</th>
<th>mIOU (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-50</td>
<td>16</td>
<td>✓</td>
<td>✓</td>
<td>49.73</td>
</tr>
<tr>
<td>(He et al. 2016)</td>
<td>16</td>
<td>✓</td>
<td>✓</td>
<td><strong>50.23</strong></td>
</tr>
<tr>
<td>Xception65</td>
<td>16</td>
<td>✓</td>
<td>✓</td>
<td>52.42</td>
</tr>
<tr>
<td>(Chollet 2017)</td>
<td>16</td>
<td>✓</td>
<td>✓</td>
<td>52.65</td>
</tr>
<tr>
<td>EfficientNetB7</td>
<td>16</td>
<td>✓</td>
<td>✓</td>
<td>57.24</td>
</tr>
<tr>
<td>(Tan and Le 2019)</td>
<td>16</td>
<td>✓</td>
<td>✓</td>
<td>57.93</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>✓</td>
<td>✓</td>
<td><strong>58.40</strong></td>
</tr>
</tbody>
</table>

Table 6: Comparisons with other state-of-the-art approaches on the PASCAL Context test set. For a fair comparison, all compared methods used ResNet-101 and naive upsampling.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Backbone</th>
<th>mIOU (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENCNet (Zhang et al. 2018)</td>
<td>ResNet-101</td>
<td>51.7</td>
</tr>
<tr>
<td>ANNet (Zhu et al. 2019)</td>
<td>ResNet-101</td>
<td>52.8</td>
</tr>
<tr>
<td>EMANet (Li et al. 2019)</td>
<td>ResNet-101</td>
<td>53.1</td>
</tr>
<tr>
<td>SPYGR (Li et al. 2020)</td>
<td>ResNet-101</td>
<td>52.8</td>
</tr>
<tr>
<td>CPN (Yu et al. 2020)</td>
<td>ResNet-101</td>
<td>53.9</td>
</tr>
<tr>
<td>CFNet (Zhang et al. 2019)</td>
<td>ResNet-101</td>
<td>54.0</td>
</tr>
<tr>
<td>DANet (Fu et al. 2019)</td>
<td>ResNet-101</td>
<td>52.6</td>
</tr>
<tr>
<td>Our CAA (OS = 16)</td>
<td>ResNet-101</td>
<td>53.7</td>
</tr>
<tr>
<td>Our CAA (OS = 8)</td>
<td>ResNet-101</td>
<td>55.0</td>
</tr>
</tbody>
</table>

5.4 Results on the Cityscapes

The Cityscapes dataset (Marius et al. 2016) has 19 classes. Following previous works (Fu et al. 2019), we train 80k iteration on the fine set with a crop size of 769×769. We report our results on the test set in Table 8. Results show our CAA is also working well on high-resolution images.

In Fig. 8, we compare the results on Cityscapes validation set predicted by DANet and our CAA. Key areas of difference are highlighted with white boxes. Results show that many errors produced by DANet no longer exist in our CAA.

5.5 Effectiveness of Our Grouped Vectorization

In sec. 4, we introduced the grouped vectorization to split tensors into multiple groups so as to reduce the GPU memory usage when performing channel attention inside spatial attention. As we use more groups in group vectorization, the proportionally less GPU memory is needed for the computation. However, longer running time is required. In this section, we conduct experiments to show the variation of the inference time (seconds/image) when different numbers of groups are used.

5.6 Stronger Backbone on the PASCAL Context

In this section, we report the performance of our CAA with EfficientNet-B7 (Tan and Le 2019) in Table 9. The “simple decoder” used in our method merges the low level features from output stride = 4, during the final upsampling (see (Chen et al. 2018) for details). Note that, the listed methods were not trained under the same settings, or using the same backbone. Our method outperforms the state-of-the-art transformer-based hybrid models, including SETR (Sixiao et al. 2021) and DPT (Ranftl, Bochkovskiy, and Koltun 2021).

5.7 Results on the COCOStuff-164k

Segformer (Xie et al. 2021) used COCOStuff-164k (164,000 images), i.e., the full set of COCOStuff-10k to validate...
Figure 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.

Table 8: 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.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Backbone</th>
<th>mIOU (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFNet (Zhang et al. 2019)</td>
<td>ResNet-101</td>
<td>79.6</td>
</tr>
<tr>
<td>ANNNet (Zhu et al. 2019)</td>
<td>ResNet-101</td>
<td>81.3</td>
</tr>
<tr>
<td>CCNet (Huang et al. 2020)</td>
<td>ResNet-101</td>
<td>81.4</td>
</tr>
<tr>
<td>CPN (Yu et al. 2020)</td>
<td>ResNet-101</td>
<td>81.3</td>
</tr>
<tr>
<td>SPYGR (Li et al. 2020)</td>
<td>ResNet-101</td>
<td>81.6</td>
</tr>
<tr>
<td>OCR (Yuan, Chen, and Wang 2020)</td>
<td>ResNet-101</td>
<td>81.8</td>
</tr>
<tr>
<td>DANet (Fu et al. 2019)</td>
<td>ResNet-101</td>
<td>81.5</td>
</tr>
<tr>
<td><strong>Our CAA</strong></td>
<td>ResNet-101</td>
<td><strong>82.6</strong></td>
</tr>
</tbody>
</table>

Table 9: 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.

<table>
<thead>
<tr>
<th>Methods</th>
<th>mIOU (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-50 + DeepLabV3+ (Chen et al. 2018)</td>
<td>38.4</td>
</tr>
<tr>
<td>HRNetV2 + OCR</td>
<td>42.3</td>
</tr>
<tr>
<td>SETR (Sixiao et al. 2021)</td>
<td>45.8</td>
</tr>
<tr>
<td>Segformer-B5 (Xie et al. 2021)</td>
<td>46.7</td>
</tr>
<tr>
<td><strong>Our CAA (EfficientNet-B5)</strong></td>
<td><strong>47.30</strong></td>
</tr>
</tbody>
</table>

Table 10: Result comparison with the state-of-the-art approaches on the COCO-Stuff-164K test set.
References


