Rethinking Dilated Convolution for Real-time Semantic Segmentation

Roland Gao
University of Toronto
roland.gao@mail.utoronto.ca

Abstract

The field-of-view is an important metric when designing a model for semantic segmentation. To obtain a large field-of-view, previous approaches generally choose to rapidly downsample the resolution, usually with average poolings or stride 2 convolutions. We take a different approach by using dilated convolutions with large dilation rates throughout the backbone, allowing the backbone to easily tune its field-of-view by adjusting its dilation rates, and show that it's competitive with existing approaches. To effectively use the dilated convolution, we show a simple upper bound on the dilation rate in order to not leave gaps in between the covolutional weights, and design an SE-ResNeXt inspired block structure that uses two parallel 3×3 convolutions with different dilation rates to preserve the local details. Manually tuning the dilation rates for every block can be difficult, so we also introduce a differentiable neural architecture search method that uses gradient descent to optimize the dilation rates. In addition, we propose a lightweight decoder that restores local information better than common alternatives. To demonstrate the effectiveness of our approach, our model RegSeg achieves competitive results on real-time Cityscapes and CamVid datasets. Using a T4 GPU with mixed precision, RegSeg achieves 78.3 mIOU on Cityscapes test set at 37 FPS, and 80.9 mIOU on CamVid test set at 112 FPS, both without ImageNet pretraining.

1. Introduction

Semantic segmentation is the task of assigning a class to every pixel in the input image. Applications of it include autonomous driving, natural scene understanding, and robotics. It is also the groundwork for the bottom-up approach [6] of panoptic segmentation, which, in addition to assigning a class to every pixel, separates instances of the same class.

The field-of-view is an important metric when designing a model for semantic segmentation. To quickly increase the field-of-view, previous advances in semantic segmentation generally adapt a backbone designed for ImageNet [10]

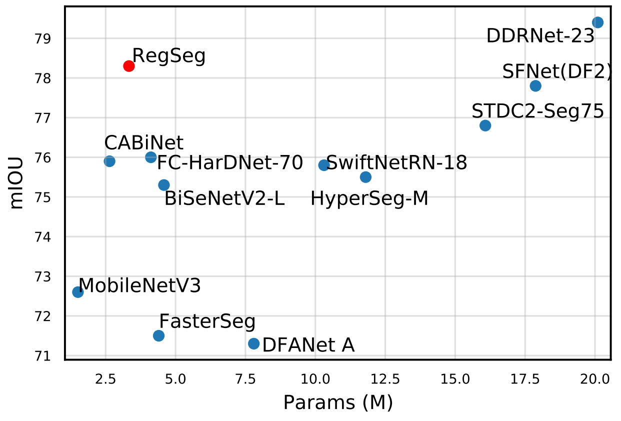

Figure 1. Params vs mIOU on Cityscapes test set. Our model is in red, while other models are in blue. We achieve SOTA params-accuracy trade-off.

and add a context module with large average poolings like PPM [43] or large dilation rates like ASPP [4]. However, the ImageNet backbone, which performs most of the computation, does not have the field-of-view required to encode high resolution images. Some recent advances such as DDRNet23 [16] and STDC [12] have already designed backbones specifically for semantic segmentation. Instead of applying rapid down sampling (DDRNet23) or using many more 3×3 convs (STDC) to increase the field-of-view, we keep the features at $1/16$ of the original resolution and use dilated convolutions with large dilation rates throughout the backbone, allowing its field-of-view to be precisely tuned by adjusting its dilation rates. To the best of our knowledge, we are the first to use large dilation rates throughout the backbone while achieving competitive results on standard semantic segmentation benchmarks.

To effectively use the dilated convolution, we have to solve many problems that previous researchers have struggled with. First, we show a simple upper bound on the dilation rate in order to not leave gaps in between the convolutional weights. Next, to preserve the local details, we modify the SE-ResNeXt [18, 38] block such that, when applying the 3×3 group conv, we use a small dilation rate

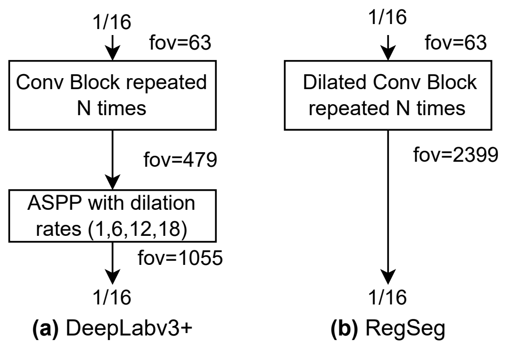
1/16
fov=63
Conv Block repeated N times
fov=479
ASPP with dilation rates (1,6,12,18)
fov=1055
1/16
(a) DeepLabv3+
1/16
fov=63
Dilated Conv Block repeated N times
fov=2399
1/16
(b) RegSeg

Figure 2. RegSeg vs DeepLabv3+ architecture design. RegSeg's simple design achieves larger field-of-view than DeepLabv3+ with RegNetY-600MF backbone (2399 vs 1055). DeepLabv3+ performs most of its computation in the conv blocks, where it has a low field-of-view. RegSeg performs most of its computation in the dilated blocks, where the field-of-view is much larger.

for half of the groups and another larger dilation rate for the other half. We adapt the fast RegNetY-600MF [11, 31] for semantic segmentation by swapping out the original SE-ResNeXt block, which they call the Y block, with our modified dilated block, which we call the D block. As shown in Fig. 2, RegSeg's simple design achieves higher field-of-view than the standard DeepLabv3+ (2399 vs 1055) without using ASPP or PPM. While DeepLabv3+ performs most of its computations in a low field-of-view ranging from 63 to 479, RegSeg performs most of its computations in a much larger field-of-view ranging from 63 to 2399.

Manually tuning the dilation rates for every block can be difficult, so we also introduce a differentiable neural architecture search method that uses gradient descent to optimize the dilation rates. We accomplish this by viewing dilated conv as a special type of deformable conv [9] and using a trainable parameter r to generate the offsets required for deformable conv. Deformable conv's implementation allows back propagation on the offsets, so we use it to optimize the dilation rate r.

We also propose a lightweight decoder that effectively restores the local details lost in the backbone. Previous decoders such as the one in DeepLabv3+ [5] are too slow to run in real-time, and common lightweight alternatives such as LRASPP [17] are not as effective. Our decoder is 1.0% better than LRASPP under the same training setting.

RegSeg's simple design allows it to be extremely efficient, achieving state-of-the-art params-accuracy trade-off and FLOPS-accuracy trade-off on Cityscapes [7], as shown in Fig. 1 and Tab. 8. RegSeg achieves those results without pretraining on ImageNet [10], thus being data-efficient as well. RegSeg demonstrates competitive runtime performance as well when timed on Nvidia's T4 GPU and Mac's M1 GPU, as shown in Tab. 8.

2. Related works

2.1. Network design

The models found on ImageNet play an important role in general network design, and their improvements often transfer to other domains such as semantic segmentation. RegNet [11, 31] finds many improvements to the ResNeXt [38] architecture by using random search to run numerous experiments and analyzing trends to reduce the search space. They provide models across a wide range of flop regimes, and the models outperform EfficientNet [35] under comparable training settings. EfficientNetV2 [36] is the improved version of EfficientNet and trains faster by using regular convs instead of depthwise convs at the higher resolutions.

2.2. Semantic segmentation

Fully Convolutional Networks (FCNs) [26, 32] are shown to beat traditional approaches in the task of segmentation. DeepLabv3 [4] uses dilated conv in the ImageNet pretrained backbone to reduce the output stride to 16 or 8 instead of the usual 32, and increases the receptive field by proposing the Atrous Spatial Pyramid Pooling module (ASPP), which applies parallel branches of convolutional layers with different dilation rates. PSPNet [43] proposes the Pyramid Pooling Module (PPM), which applies parallel branches of convolutional layers with different input resolutions by first applying average poolings.

DeepLabv3+ [5] builds on top of DeepLabv3 by adding a simple decoder with two 3×3 convs at output stride 4 to improve the segmentation quality around boundaries. HRNetV2 [37] keeps parallel branches with different resolutions right in the backbone, with the finest one at output stride 4.

In our paper, we adapt RegNet for semantic segmentation by directly using dilated conv with large dilation rates in the backbone and show that it performs better than using DeepLabV3+'s approach of using attaching an ASPP.

2.3. Real-time semantic segmentation

MobilenetV3 uses the lightweight decoder LRASPP [17] to adapt the fast ImageNet model for semantic segmentation. BiSeNetV1 [40] and BiSeNetV2 [39] have two branches in the backbone (Spatial Path and Context Path) and merge them at the end to achieve good accuracy and performance without ImageNet pretraining. SFNet [24] proposes the Flow Alignment Module (FAM) to upsample low resolution features better than bilinear interpolation. STDC [12] rethinks the BiSeNet architecture by removing the Spatial Path and designing a better backbone. HarDNet [3] reduces GPU memory traffic consumption by using mostly 3×3 convs and barely any 1×1 convs. DDRNet-23 [16] uses two branches with multiple bilateral fusions

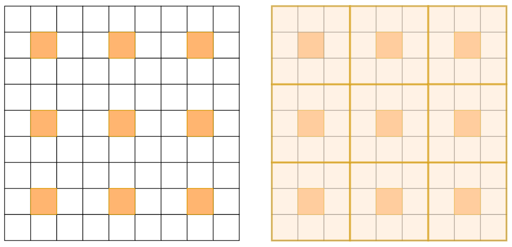

Figure 3. Upper bound $r \leq k/s$ in action. Left: a 3×3 dilated conv with dilation rate 3 leaves gaps in between the weights. Right: Using a 3×3 conv with stride 1 before the dilated conv results in no gaps because the upper bound is satisfied with $r = 3, k = 3, s = 1$.

between them and appends a new context module called the Deep Aggregation Pyramid Pooling Module (DAPPM) at the end of the backbone.

3. Methods

3.1. Field-of-view

We are interested in the field-of-view (FOV), also known as the receptive field, of our model gained through convolutions. For example, a composition of two 3×3 convs is equal in kernel size and stride to a 5×5 conv, and we simply say that the field-of-view is 5. More generally, the field-of-view of a composition of convs can be calculated iteratively as described in FCN [26]. Suppose the composition of convs up to the current point is equal in kernel size and stride to one $k \times k$ conv with stride s, and we compose it with a $k' \times k'$ conv with stride s'. We update k and s by

$$k \leftarrow k + (k' - 1) * s \tag{1}$$

$$s \leftarrow s * s' \tag{2}$$

The field-of-view is the final value of k.

A 3×3 conv with dilation rate r is equal in field-of-view to a conv with kernel size $2r + 1$. However, this field-of-view is only valid if there are no gaps in between the convolution weights, which happens when the following inequality is satisfied.

$$r \leq k/s \tag{3}$$

Fig. 3 gives a simple example of this upper bound in action. Using a 3×3 conv with dilation rate 3 as the first operation leaves gaps in between weights, as shown in Fig. 3a. If we first use a 3×3 conv with stride 1 before the dilated conv, the inequality is satisfied, so there will not be any gaps, as shown in Fig. 3b.

If the composition of convs before the current point is equal in kernel size and stride to a $k \times k$ conv with stride s, then each weight location of the dilated conv corresponds to a $k \times k$ region in the input image. Moving from one weight location to its neighbouring weight location requires shifting r pixels in one direction, which corresponds to shifting the $k \times k$ region in the input image by sr pixels. To not leave any gaps in between the two regions, we need $sr \leq k$, or $r \leq k/s$, as desired. In practice, we choose dilation rates much lower than the upper bounds.

Intuitively, the model should have a large enough field-of-view so that each pixel in the output can see the entire image. For example, if we use the ResNet [14] architecture on ImageNet with a testing crop size of 224×224 and look at the feature maps right before the global average pooling, the model needs a field-of-view of at least $224 * 2 - 1 = 447$ for the top-left pixel to see the entire image. Similarly, on Cityscapes with image size 1024×2048, the model needs a field-of-view of 2047 for the top-left pixel of the output to see the bottom-left pixel of the input image, and a field-of-view of 4095 to see the bottom-right pixel of the input image.

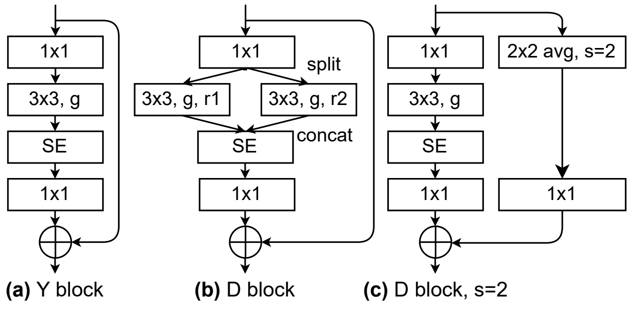

Figure 4. Y block and D block. When $r1 = r2 = 1$, the D block is the same as the Y block.

3.2. Dilated block

Our dilated block (D block) takes inspiration from the Y block of RegNet [31], also known as the SE-ResNeXt block [18]. The Y block and our new D block utilize group convolutions. Suppose input channels = output channels = w, which is always true for the 3×3 convs in the Y block and the D block. A group conv has an attribute called the group width g, and g must divide w. During the forward pass, the input with w channels are split into w/g groups with g channels each, and a regular conv is applied to each group, and the outputs are concatenated together to form the w channels again.

Since there is a conv for each group, we can apply different dilation rates to different groups to extract multi-scale features. For example, we can apply dilation rate 1 to half of the groups, and dilation rate 10 to the other half. This is the key to our D block. Fig. 4a shows the Y block. Fig. 4b shows our D block. When $r1 = r2 = 1$, the D block is

Operator	Stride	#Channels	#Repeat
3x3 conv	2	32	1
D block	2	48	1
D block	2	128	3
D block	2	256	13
D block	1	320	1

Table 1. Backbone. #Channels is the number of output channels, and the number of input channels is inferred from the previous block. When stride $= 2$ and #repeat > 1, the first block has stride 2 and the rest have stride 1.

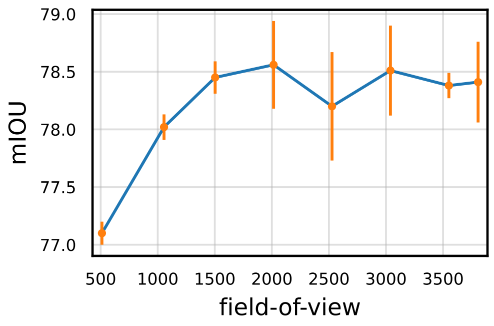

Figure 5. mIOU vs field-of-view on Cityscapes with error bars. The accuracy increases as the field-of-view increases and it stabilizes when the field-of-view reaches around 2000.

equivalent to the Y block. In Sec. 4.4, we experiment with some D blocks that have 4 branches of different dilation rates, but find that they are no better than D blocks that have 2 branches. Fig. 4c shows the D block when stride $= 2$. Similar to the ResNet D-variant [15], we apply a 2×2 average pooling on the shortcut branch when the block's stride $= 2$. BatchNorm [21] and ReLU immediately follow each conv, except that the ReLUs right before the summation are replaced with one after the summation. We use an SE [18] reduction ratio of $1/4$.

Other multi-path block structures with attention mechanisms have been proposed before, such as SKNet [23], ResNeSt [41], DetectoRS [30]. Unlike the previous approaches, our main focus is to bring dilated convolution with large dilation rates into the backbone to efficiently increase its field-of-view. For example, ResNeSt does not use dilated convolution at all; SKNet and detectoRS use only small dilation rates that are at most 3. Moreover, the D block is a direct replacement of the Y block in the original RegNet model, with exactly the same number of parameters and FLOPs; other methods use more parameters and FLOPs than the original model to implement the multi-path block structure.

3.3. Backbone

This backbone is directly inspired by RegNetY-600MF [31] and has similar params, FLOPs, and runtime. The backbone starts with one 32-channel 3×3 conv with stride 2. Then it has one 48-channel D block at $1/4$ resolution, three 128-channel D block at $1/8$, thirteen 256-channel D block at $1/16$, ending with one 320-channel D block at $1/16$. Group width $g = 16$ for all D blocks. We do not downsample to $1/32$. In a format similar to EfficientNetV2 [36], we display the backbone of RegSeg in Tab. 1.

We tune the dilation rates for the last 13 stride 1 blocks while fixing the dilation rate to 1 for all the earlier blocks. We can easily adjust the field-of-view of the backbone by specifying the dilation rates. For example, we can perform an mIOU vs field-of-view analysis by training models of field-of-view from 500 to 3500 in steps of 500, as shown in Fig. 5. In order to understand what dilation rates would be good for the model, we develop a differentiable neural architecture search method, as discussed in the next section.

3.4. Gradient descent on the dilation rates

Manually tuning the dilation rates for every block can be time-consuming, so we introduce a differentiable neural architecture search method that uses gradient descent to optimize the dilation rates. First, we note that dilated conv is a special case of deformable conv [9], which uses offsets, generally produced by an auxiliary conv, to guide the positions of the weights of the main conv. Even though deformable conv is theoretically more powerful than the dilated conv, it is significantly slower. Besides, its ability to gain the field-of-view is equal to that of the dilated conv because they are both bounded above by $r \leq k/s$, as discussed in Sec. 3.1.

For each dilation rate that we want to tune, we introduce a trainable parameter r that represents the dilation rate. We relax the supposedly integer variable to a floating-point and use it to generate the offsets of the deformable conv, such that when r is an integer, the deformable conv does exactly what the dilated conv with dilation rate r does. Fig. 7 shows examples of when $r = 1.5$ and $r = 2.5$. After training the model with deformable conv, we have to convert the model's fractional dilation rates to integers and train the converted model again. For each block with dilation rates $r1, r2$, we convert the dilation rates to floor$(\min(r1, r2))$, ceil$(\max(r1, r2))$ so that the converted model maintains the same field-of-view as the original one and preserves the local details.

3.5. Decoder

The decoder's job is to restore the local details lost in the backbone. Similar to DeepLabv3+ [5], we use $[k \times k, c]$ to denote a $k \times k$ conv with c output channels. We take the backbone's last $1/4$, $1/8$, and $1/16$ feature maps as inputs.

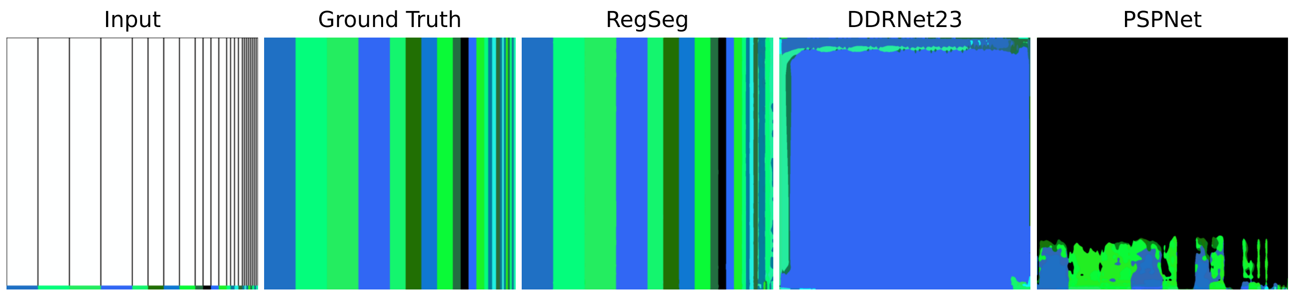

Figure 6. Comparison between RegSeg and DDRNet23 on the toy dataset.

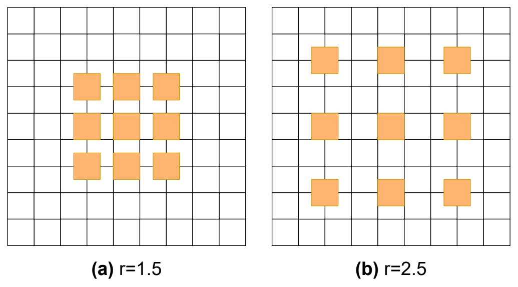

Figure 7. Trainable dilation rates. In order to perform gradient descent on the dilation rates, we extend the type to floating-point. Left: $r = 1.5$, right: $r = 2.5$.

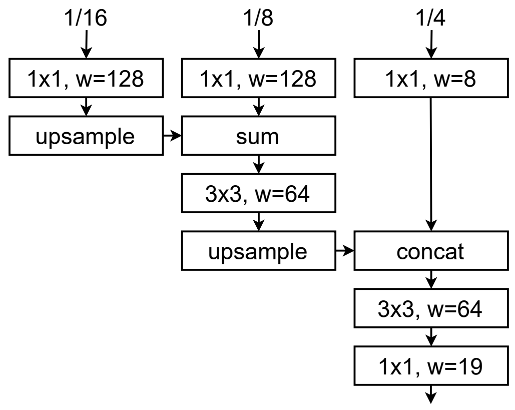

Figure 8. Decoder. w shows the number of output channels. All convs except the final one are followed by BatchNorm [21] and ReLU.

We apply a $[1 \times 1, 128]$ conv to $1/16$, a $[1 \times 1, 128]$ conv to $1/8$ and a $[1 \times 1, 8]$ conv to $1/4$. We upsample the $1/16$, sum it with the $1/8$, and apply a $[3 \times 3, 64]$ conv. We upsample again, concatenate with the $1/4$, and apply a $[3 \times 3, 64]$ conv, before the final $[1 \times 1, 19]$ conv. All convs except the final one are followed by BatchNorm [21] and ReLU. The decoder is shown in Fig. 8. This simple decoder performs better than many existing decoders that have similar latencies, as shown in Sec. 4.5.

3.6. Dilated conv versus downsampling

While dilated conv and downsampling are both effective ways to gain field-of-view, they both have advantages and disadvantages. Downsampling is usually computationally cheaper than dilated conv because it operates at a coarser resolution. On the other hand, dilated conv can handle thin and long objects that downsampling methods usually fail at because dilated conv operates at a high resolution and has a large field-of-view while downsampling method's coarse resolution loses representations for thin objects. Since downsampling methods have to spend more computation to upsample and retrieve the local details than dilated conv methods, they are not necessarily computationally cheaper overall. RegSeg's simple and intuitive design allows it to run in real-time while achieving competitive results.

We create a toy dataset to demonstrate RegSeg's ability to handle thin and long objects. The synthetic images are 1024×1024 and contain rectangles that have the full height of 1024 but a small width. The ground truth labels are at the bottom of the rectangles, and the model needs to propagate the labels to the top. As shown in Fig. 6, RegSeg excels at the task, only failing on extra thin rectangles of width 8 since RegSeg operates at $1/16$ of the input resolution. Downsampling methods such as DDRNet23 [16] and PSPNet [43] completely fail at the task.

4. Experiments

4.1. Datasets

Cityscapes [7] is a large-scale dataset focused on street scene parsing. It contains 2975 images for training, 500 for validation, and 1525 for testing. We do not use the 20000 coarsely labeled images. There are 19 classes and ignore label $= 255$. The image size is 1024×2048.

CamVid [1] is another street scene dataset similar to Cityscapes. It contains 367 images for training, 101 for validation, and 233 for testing. Following previous works

[22, 28, 40], we use only 11 classes and set all other classes to the ignore label $= 255$. We train on the trainval set and evaluate on the test set. The image size is 720×960.

4.2. Train setting

On Cityscapes, we use SGD with momentum $= 0.9$, initial learning rate $= 0.05$, batch size $= 8$, epochs $= 1000$, and weight decay $= 0.0001$, but we do not decay BatchNorm parameters. We use Online Hard Example Mining loss [33] (OHEM), which averages pixel losses that are over 0.3 or averages the top $1/16$ pixel losses if the original proportion is less than $1/16$. We use the poly learning rate scheduler and a linear warmup [13] from $0.1lr$ to lr for the first 3000 iterations. We apply random horizontal flipping, random scaling of $[400, 1600]$ for the shorter side while preserving the aspect ratio, and random cropping of 1024×1024. We use a reduced set of RandAug [8] operations (auto contrast, equalize, rotate, color, contrast, brightness, sharpness). For each image, we apply 2 random operations of magnitude 0.2 (out of 1). We also use class uniform sampling [44] with class uniform percent $= 0.5$. The weights are initialized using PyTorch's [29] default initialization. We use a single T4 GPU with mixed precision training. When submitting to the test server, we train on the trainval set. We also experiment with ImageNet [10] pretraining. In this case, we first train RegSeg for 100 epochs on ImageNet with train crop size 320×320 before training 500 epochs on Cityscapes, where the training recipe on Cityscapes is the same except that we also use exponential moving average (EMA) with alpha 0.0031 and update every 32 iterations.

On CamVid, the training setting is similar to that in Cityscapes. Since RegSeg is not pretrained on ImageNet and CamVid is small, it is initialized using Cityscapes pretrained weights. We use batch size $= 12$ and epochs $= 200$. We apply random horizontal flipping, random scaling of $[288, 1152]$, and random cropping of 720×960. We do not apply RandAug or class uniform sampling.

4.3. Reproducibility

To make our ablation studies possible, we need the results to be reproducible. We sort the training images by their filenames to prevent different orders caused by different file systems. Before randomly initializing the model weights, we set the random seed to 0. At the start of each epoch, we set the random seed to the current epoch. By doing so, we eliminate the problem of being in different states of the random number generator during training, caused by initializing different models or by resuming the model training after an incomplete training session. Furthermore, because random shuffling of the filenames happens at the start of each epoch, we can guarantee the same order of images even under different data augmentations. We run each experiment three times to get the mean and standard deviation.

Dilation rates	Field-of-view	mIOU
(1,1)+(1,2)+(1,2)+(1,3) +(2,3)+(2,7)+(2,3)+(2,6) +(2,5)+(2,9)+(2,11)+(4,7)+(5,14)	2399	78.50 ± 0.25

Table 2. The dilation rates found using differentiable neural architecture search. This method achieves comparable accuracy to the manually searched results in Fig. 5

backbone	context module	mIOU	FPS
RegNetY-600MF	None	75.66 ± 0.53	31
RegNetY-600MF	ASPP	77.29 ± 0.18	28
RegNetY-600MF	PPM	77.47 ± 0.29	30
RegSeg	None	$\mathbf{78.65} \pm 0.28$	30

Table 3. mIOU vs backbone on Cityscapes. We fix the decoder while ablating the backbone. When using a backbone with a small field-of-view such as RegNetY-600MF, adding a context module such as PPM or ASPP increases performance. However, incorporating dilated convolution into the backbone is much more effective than adding a context module, as shown by the superiority of RegSeg.

number of branches	mIOU
2	78.57 ± 0.39
4	78.47 ± 0.56

Table 4. mIOU vs number of branches in the D block on Cityscapes.

model	mIOU	FPS
fractional dilation rates	78.57 ± 0.39	18
integral dilation rates	78.65 ± 0.28	**30**

Table 5. Fractional dilation rates vs integral dilation rates. Rounding the dilation rates to integers gives no loss in accuracy and increases the inference speed.

4.4. Backbone ablation studies

We perform many ablation studies on the backbone design. First, we experimentally confirm our claim that field-of-view is essential for the accuracy. As shown in Fig. 5, accuracy increases along with the field-of-view until it stabilizes when field-of-view hits around 2000. Second, we find that using 4 branches in the D block is not better than using 2 branches, as shown in Tab. 4. Third, we show that the model with the rounded dilation rates perform just as well as the differentiable neural searched model with the fractional dilation rates and significantly increases the inference speed, as shown in Tab. 5.

Decoder	mIOU
LRASPP [17]	77.75 ± 0.19
BiSeNetDecoder [40]	77.87 ± 0.18
SFNetDecoder [24]	78.12 ± 0.38
Sec. 3.5 decoder	$\mathbf{78.65} \pm 0.28$

Table 6. Decoder Comparison on Cityscapes. Our best decoder performs better than common alternatives.

The main motivation for the development of RegSeg is that using the dilated conv throughout the backbone should perform better than using dilated conv solely in the last layer. As shown in Tab. 3, although adding a context module such as ASPP [4] or PPM [43] increases performance over the small-field-of-view RegNetY-600MF, RegSeg performs more than 1% better than RegNetY+ASPP and RegNetY+PPM by using dilated conv throughout the backbone. Fig. 9 shows a qualitative comparison between RegSeg and PSPNet (RegNetY600MF+PPM).

We show the dilation rates found by the differentiable neural architecture search method in Tab. 2. There are a few patterns that we found. The dilation rate in one branch is small to preserve the local details while the other branch uses a larger dilation rate to increase the field-of-view. The dilation rate increases over the blocks because the later blocks can use large dilation rates to the capitalize on the already increased field-of-view and because their upper bounds on the dilation rates are higher according to Sec. 3.1. Interestingly, the DNAS model does not always keep the smaller dilation rate in a D block to 1 as our manual models do; instead, the small dilation rate increases to 2, 4, and 5 the closer it is to the output.

4.5. Decoder comparison

In Tab. 6, we experiment with the decoder design while fixing the backbone architecture. Existing decoders [17, 24, 40] perform much worse than our best decoder, potentially because they are designed for backbones that do not have a large field-of-view. Notably, RegSeg performs around 1% better than DeepLab's LRASPP.

4.6. Timing

We time RegSeg using Nvidia's T4 GPU and Mac's M1 GPU, both with mixed precision. We use PyTorch 1.12.1 [29] with CUDA 11.3. The input size is $1 \times 3 \times 1024 \times 2048$ on Cityscapes, and $4 \times 3 \times 720 \times 960$ on CamVid. After 10 iterations of warm up, we average the model's time over the next 100 iterations. We set torch.backends.cudnn.benchmark=True and use torch.cuda.synchronize().

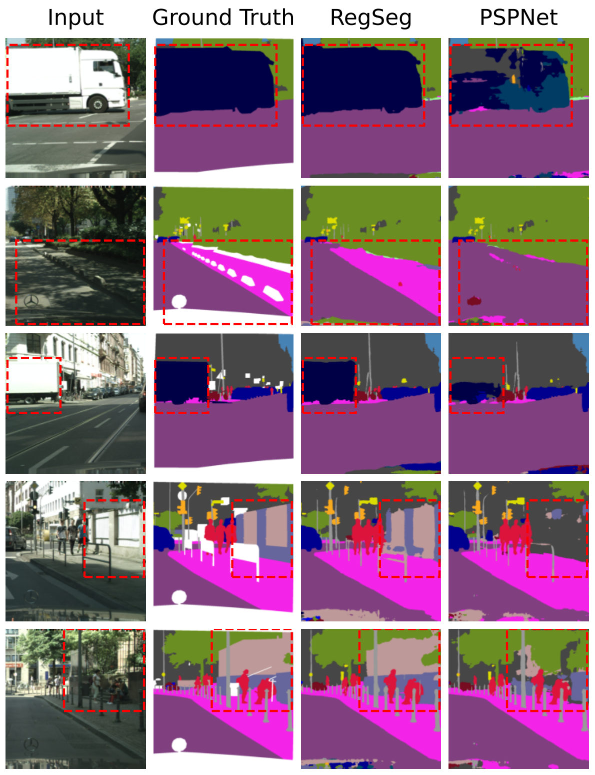

Figure 9. Comparison between RegSeg and baseline PSPNet (RegNetY600MF+PPM) on Cityscapes. RegSeg performs better than PSPNet on large objects.

Model	Extra data	mIOU	T4 FPS
STDC2-Seg [12]	IM	73.9	152.2
GAS [25]	-	72.8	153.1
CAS [42]	-	71.2	169
SFNet(DF2) [24]	IM	70.4	134
SFNet(ResNet-18) [24]	IM	73.8	36
MSFNet [34]	IM	75.4	91
HyperSeg-S [27]	IM	78.4	38.0
TD4-PSP18 [19]	IM	72.6	25.0
VideoGCRF [2]	C	75.2	-
BiSeNetV2 [39]	C	76.7	124
BiSeNetV2-L [39]	C	78.5	33
CCNet3D [20]	C	79.1	-
DDRNet-23 [16]	C	80.1±0.4	98*
RegSeg	C	$\mathbf{80.9} \pm 0.07$	112*

Table 7. Accuracy and speed comparison on CamVid, IM: ImageNet, C: Cityscapes. DDRNet-23 and RegSeg are timed using the same environment.

Model	Extra data	val mIOU	test mIOU	T4 FPS	M1 FPS	Resolution	Params (M)	GFLOPs
MobileNetV3 [17]	None	72.36	72.6	56	4.7	1024x2048	1.51	9.74
BiSeNetV2-L [39]	None	75.8	75.3	83	28.1	512x1024	4.59	139
HyperSeg-M [27]	IM	76.2	75.8	31	5.9	512x1024	10.3	8.4
FC-HarDNet-70 [3]	None	77.7	76.0	40	14.0	1024x2048	4.12	35.6
STDC2-Seg75 [12]	IM	77.0	76.8	40	17.8	768x1536	16.1	54.9
SFNet(DF2) [24]	IM	-	77.8	53	11.7	1024x2048	17.9	80.4
DDRNet-23 [16]	IM	**79.1**	**79.4**	33	9.3	1024x2048	20.1	143.1
RegSeg	None	78.50	78.3	37	11.3	1024x2048	3.34	39.1
RegSeg	IM	**79.42**	**79.1**	37	11.3	1024x2048	3.34	39.1
RegSeg	IM	78.06	77.5	46	18.3	768x1536	3.34	22.0

Table 8. Accuracy and speed comparison on Cityscapes. IM: ImageNet. All FPS numbers are measured under our timing environment.

Model	T4 FPS	M1 FPS	val mIOU	Params (M)	GFLOPs
DDRNet-23	33	9.3	78.55 ± 0.35	20.1	143.1
RegSeg	**37**	**11.3**	78.50 ± 0.25	**3.34**	**39.1**

Table 9. Comparison against DDRNet-23 under the exact same training setting. FPS of both models are calculated using Sec. 4.6. RegSeg achieves higher FPS and better parameters and flops efficiency while maintaining similar accuracy.

4.7. Comparison on CamVid

As shown in Tab. 7, RegSeg achieves 80.9 mIOU on CamVid test set at 112 FPS. It outperforms the previous SOTA DDRNet-23 by 0.8%, and BiSeNetV2-L by 2.4%. The results show that RegSeg may generalize better than DDRNet-23. RegSeg demonstrates better data efficiency as CamVid is a small dataset.

4.8. Comparison on Cityscapes

We compare against other real-time models on Cityscapes. As shown in Fig. 1, RegSeg achieves the best parameter-accuracy trade-offs. In Tab. 8, we show the accuracy and speed comparison on Cityscapes. We measure the FPS of all models in our timing environment. RegSeg outperforms HarDNet [3], which is the previous SOTA model without extra data, by 1.5%, and outperforms SFNet(DF2) [24] by 0.5%. RegSeg also outperforms the popular BiSeNetV2-L [39] by 3.0%, and MobileNetV3+LRASPP [17] by 5.7%. With ImageNet pretraining, RegSeg achieves 79.4 mIOU on the Cityscapes val set. By using a smaller resolution, 768×1536, RegSeg achieves 78.06 on the Cityscapes val set with 46 FPS on T4 and 18.3 FPS on M1, outperforming STDC2-Seg75 [12]. As shown in Tab. 9, RegSeg achieves similar results with DDRNet-23 when trained using the exact same training settings, while having better parameters, flops, and runtime efficiency. Both models are trained 5 times under our training setting described in Sec. 4.2.

5. Limitations

RegSeg's backbone might not suitable for small images such as ImageNet's usual 224×224 train crop size because most weights of the 3×3 dilated conv with large dilation rates will hit the zero padding, and the dilated conv will reduce to a normal 1×1 conv.

6. Conclusion

In this paper, we are interested in increasing the field-of-view of the backbone by using dilated convolution with large dilation rates, while aiming for real-time performance. We demonstrate how to effectively use the dilated convolution, by having an upper bound on the dilation rate to not leave gaps in between convolutional weights and introducing the novel D block, which can increase the field-of-view in one path while preserving the local details in the other. We introduce a differentiable neural architecture search method to optimize the dilation rates by gradient descent and compare the results with the manual search metho. We also propose a lightweight decoder that performs better than common alternatives. Together, RegSeg is a radically novel approach on real-time semantic segmentation that is competitve with the state of the art on Cityscapes and CamVid datasets.

References

[1] Gabriel J Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. Segmentation and recognition using structure from motion point clouds. In *ECCV*, pages 44–57. Springer, 2008. 5

[2] Siddhartha Chandra, Camille Couprie, and Iasonas Kokkinos. Deep spatio-temporal random fields for efficient video segmentation. In *CVPR*, pages 8915–8924, 2018. 7

[3] Ping Chao, Chao-Yang Kao, Yu-Shan Ruan, Chien-Hsiang Huang, and Youn-Long Lin. Hardnet: A low memory traffic network. In *ICCV*, pages 3552–3561, 2019. 2, 8

[4] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for seman-

tic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 1, 2, 7

[5] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. 2, 4

[6] Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *CVPR*, 2020. 1

[7] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, pages 3213–3223, 2016. 2, 5

[8] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPR Workshops*, pages 702–703, 2020. 6

[9] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017. 2, 4

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. Ieee, 2009. 1, 2, 6

[11] Piotr Dollár, Mannat Singh, and Ross Girshick. Fast and accurate model scaling. In *CVPR*, 2021. 2

[12] Mingyuan Fan, Shenqi Lai, Junshi Huang, Xiaoming Wei, Zhenhua Chai, Junfeng Luo, and Xiaolin Wei. Rethinking bisenet for real-time semantic segmentation. In *CVPR*, pages 9716–9725, June 2021. 1, 2, 7, 8

[13] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 6

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 3

[15] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *CVPR*, pages 558–567, 2019. 4

[16] Yuanduo Hong, Huihui Pan, Weichao Sun, and Yisong Jia. Deep dual-resolution networks for real-time and accurate semantic segmentation of road scenes. *arXiv preprint arXiv:2101.06085*, 2021. 1, 2, 5, 7, 8

[17] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. In *ICCV*, 2019. 2, 7, 8

[18] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, pages 7132–7141, 2018. 1, 3, 4

[19] Ping Hu, Fabian Caba, Oliver Wang, Zhe Lin, Stan Sclaroff, and Federico Perazzi. Temporally distributed networks for fast video semantic segmentation. In *CVPR*, pages 8818–8827, 2020. 7

[20] Zilong Huang, Xinggang Wang, Yunchao Wei, Lichao Huang, Humphrey Shi, Wenyu Liu, and Thomas S Huang. Ccnet: Criss-cross attention for semantic segmentation. *TPAMI*, 2020. 7

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456. PMLR, 2015. 4, 5

[22] Hanchao Li, Pengfei Xiong, Haoqiang Fan, and Jian Sun. Dfanet: Deep feature aggregation for real-time semantic segmentation. In *CVPR*, June 2019. 6

[23] Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 510–519, 2019. 4

[24] Xiangtai Li, Ansheng You, Zhen Zhu, Houlong Zhao, Maoke Yang, Kuiyuan Yang, Shaohua Tan, and Yunhai Tong. Semantic flow for fast and accurate scene parsing. In *ECCV*, pages 775–793. Springer, 2020. 2, 7, 8

[25] Peiwen Lin, Peng Sun, Guangliang Cheng, Sirui Xie, Xi Li, and Jianping Shi. Graph-guided architecture search for real-time semantic segmentation. In *CVPR*, pages 4203–4212, 2020. 7

[26] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015. 2, 3

[27] Yuval Nirkin, Lior Wolf, and Tal Hassner. Hyperseg: Patchwise hypernetwork for real-time semantic segmentation. In *CVPR*, pages 4061–4070, 2021. 7, 8

[28] Marin Orsic, Ivan Kreso, Petra Bevandic, and Sinisa Segvic. In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images. In *CVPR*, June 2019. 6

[29] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS*, 32:8026–8037, 2019. 6, 7

[30] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. In *CVPR*, pages 10213–10224, 2021. 4

[31] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *CVPR*, pages 10428–10436, 2020. 2, 3, 4

[32] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014. 2

[33] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *CVPR*, pages 761–769, 2016. 6

[34] Haiyang Si, Zhiqiang Zhang, Feifan Lv, Gang Yu, and Feng Lu. Real-time semantic segmentation via multiply spatial fusion network. *arXiv preprint arXiv:1911.07217*, 2019. 7

[35] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, pages 6105–6114. PMLR, 2019. 2

[36] Mingxing Tan and Quoc V Le. Efficientnetv2: Smaller models and faster training. In *ICML*, 2021. 2, 4

[37] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition. *TPAMI*, 2019. 2

[38] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, pages 1492–1500, 2017. 1, 2

[39] Changqian Yu, Changxin Gao, Jingbo Wang, Gang Yu, Chunhua Shen, and Nong Sang. Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation. *IJCV*, pages 1–18, 2021. 2, 7, 8

[40] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *ECCV*, pages 325–341, 2018. 2, 6, 7

[41] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Haibin Lin, Zhi Zhang, Yue Sun, Tong He, Jonas Mueller, R Manmatha, et al. Resnest: Split-attention networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2736–2746, 2022. 4

[42] Yiheng Zhang, Zhaofan Qiu, Jingen Liu, Ting Yao, Dong Liu, and Tao Mei. Customizable architecture search for semantic segmentation. In *CVPR*, pages 11641–11650, 2019. 7

[43] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, pages 2881–2890, 2017. 1, 2, 5, 7

[44] Yi Zhu, Karan Sapra, Fitsum A Reda, Kevin J Shih, Shawn Newsam, Andrew Tao, and Bryan Catanzaro. Improving semantic segmentation via video propagation and label relaxation. In *CVPR*, pages 8856–8865, 2019. 6