Rethink Dilated Convolution for Real-time Semantic Segmentation

Roland Gao

University of Toronto

roland.gao@mail.utoronto.ca

Abstract

Recent advances in semantic segmentation generally adapt an ImageNet pretrained backbone with a special context module after it to quickly increase the field-of-view. Although successful, the backbone, in which most of the computation lies, does not have a large enough field-of-view to make the best decisions. Some recent advances tackle this problem by rapidly downsampling the resolution in the backbone while also having one or more parallel branches with higher resolutions. We take a different approach by designing a ResNeXt inspired block structure that uses two parallel 3×3 convolutional layers with different dilation rates to increase the field-of-view while also preserving the local details. By repeating this block structure in the backbone, we do not need to append any special context module after it. 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 state-of-the-art 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 30 FPS, and 80.9 mIOU on CamVid test set at 70 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 [7] of panoptic segmentation, which, in addition to assigning a class to every pixel, separates instances of the same class.

Previous advances in semantic segmentation generally adapt ImageNet [10] pretrained backbones and add a context module with large average poolings like PPM [50] or large dilation rates like ASPP [4] to quickly increase the field-of-view. They take advantage of the ImageNet pretrained weights for faster convergence and for higher accuracy on smaller datasets like PASCAL VOC 2012 [12],
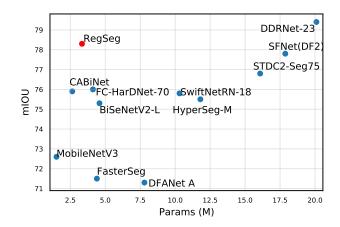

Figure 1. Params vs mIOU on Cityscapes test set. Our model is in red, while other models are in blue. We achieve SOTA params-accuracy trade-off.

where training from scratch may not be possible. Such approaches have two potential problems. ImageNet backbones usually have a large number of channels in the last few convolutional layers because they are meant to label images to one of the 1000 classes in ImageNet. For example, ResNet18 [16] ends with 512 channels, and ResNet50 with 2048 channels. The authors of Mobilenetv3 [19] find that halving the number of channels in the last convolutional layer does not reduce the accuracy when adapted for semantic segmentation, hinting at the channel-redundancy of ImageNet models. Second, ImageNet models are tuned to take input images with resolution around 224×224, but the images in semantic segmentation are much larger. For example, Cityscapes [8] has images with resolution 1024×2048, and CamVid [1] with 720×960. ImageNet models lack the field-of-view to encode such large images.

These two problems have inspired us to design a backbone specifically made for semantic segmentation. We increase the field-of-view directly in the backbone by introducing a novel dilated block structure called the D block, and we keep the number of channels in the backbone low. We take inspiration from the ResNeXt [45] block structure, which uses group convolution in the traditional ResNet

1

block to improve its accuracy while maintaining similar run time complexity. RegNet [36] takes the ResNeXt block and provides better baselines across a wide range of FLOP regimes. We adapt the fast RegNetY-600MF for semantic segmentation by swapping out the original Y block with our D block. In particular, when doing group conv, the D block uses one dilation rate for half of the groups and another dilation rate for the other half. By repeating the D block in our RegSeg's backbone, we can easily increase the field-of-view without losing the local details. RegSeg's backbone uses dilation rates as high as 14, and since it has enough field-of-view, we do not append any context modules such as ASPP or PPM.

Many recent works – such as Auto-DeepLab [30], dilated SpineNet [37], and DetectoRS [35] – are hesitant to include dilated convolution with large dilation rates in their architecture design space and still rely on context modules such as ASPP or PPM to increase the field-of-view. We attribute this to the fact that dilated conv leaves holes in between the weights. We solve this problem by starting with small dilation rates and always setting the dilation rate to 1 in one of the branches of the D block. We hope that this work can inspire future researchers to try larger dilation rates in their models.

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 [19] are not as effective. Our decoder is 1.0% better than LRASPP under the same training setting.

RegSeg runs in real-time. Using a T4 GPU with mixed precision, RegSeg runs at 30 FPS on Cityscapes and 70 FPS on CamVid. Many tasks require the model to run in real-time, such as autonomous driving or mobile deployment. Real-time models are more efficient than non-real-time models, and they have the potential to beat the state-of-the-art when scaled up to the same computational complexity. For example, EfficientNet [41] previously achieved state-of-the-art results on ImageNet by scaling up a low-compute model that they found using neural architecture search.

In summary, our contributions are:

- We propose a novel dilated block structure (D block) that can easily increase the field-of-view of the backbone while maintaining the local details. By repeating the D block in RegSeg's backbone, we can control the field-of-view without extra computation.

- We introduce a lightweight decoder that performs better than common alternatives.

- We conduct extensive experiments to show the effectiveness of our approach. RegSeg achieves 78.3 mIOU

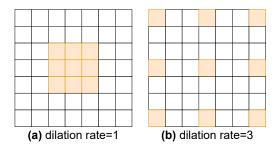**(a)** dilation rate=1 **(b)** dilation rate=3

Figure 2. Dilated Convolution.

on Cityscapes test set at 30 FPS, and 80.9 mIOU on CamVid test set at 70 FPS, both without ImageNet pre-training. RegSeg outperforms the best peer-reviewed results of SFNet(DF2) [26] by 0.5% on Cityscapes test set. And RegSeg outperforms the concurrent, non-peer-reviewed work of DDRNet-23 [18] by 0.5% on Cityscapes val set under the same training setting.

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, 36] finds many improvements to the ResNeXt [45] 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 [41] under comparable training settings. EfficientNetV2 [42] is the improved version of EfficientNet and trains faster by using regular convs instead of depthwise convs at the higher resolutions. In our paper, we take inspiration from RegNet by adapting their block structure for semantic segmentation.

2.2. Semantic segmentation

Fully Convolutional Networks (FCNs) [31, 38] 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. Fig. 2 shows an example of dilated conv. PSPNet [50] proposes the Pyramid Pooling Module (PPM), which applies parallel branches of convolutional layers with different input resolutions by first applying average poolings. In our paper, we propose the dilated block (D block) with a similar structure to ASPP and make it the building block of our backbone, instead of attaching one at the end.

2

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. HR-NetV2 [44] keeps parallel branches with different resolutions right in the backbone, with the finest one at output stride 4.

2.3. Real-time semantic segmentation

MobilenetV3 uses the lightweight decoder LRASPP [19] to adapt the fast ImageNet model for semantic segmentation. BiSeNetV1 [48] and BiSeNetV2 [47] 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 [26] proposes the Flow Alignment Module (FAM) to upsample low resolution features better than bilinear interpolation. STDC [13] rethinks the BiSeNet architecture by removing the Spatial Path and designing a better backbone. HarD-Net [3] reduces GPU memory traffic consumption by using mostly 3×3 convs and barely any 1×1 convs. DDRNet-23 [18] uses two branches with multiple bilateral fusions between them and appends a new context module called the Deep Aggregation Pyramid Pooling Module (DAPPM) at the end of the backbone. DDRNet-23 is a concurrent work that has not been peer-reviewed. DDRNet-23 is currently the state of the art on real-time Cityscapes semantic segmentation, and we show that RegSeg outperforms DDRNet-23 when trained under the same training setting.

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 [31]. 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

$$k \leftarrow k + (k' - 1) * s \qquad (1)$$
$$s \leftarrow s * s' \qquad (2)$$

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

There are two main ways to efficiently increase the field-of-view. One is to downsample early on with stride 2 convs or average poolings. The other is to use dilated conv. A 3×3 conv with dilation rate r is equal in field-of-view to a conv with kernel size $2r + 1$. However, to not leave any holes in between the weights, we need $k/s \geq r$, where k and s are
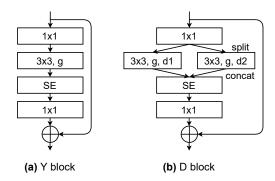

(a) Y block **(b)** D block

Figure 3. Y block and D block. When $d1 = d2 = 1$, the D block is the same as the Y block.
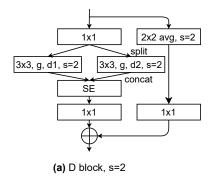

(a) D block, s=2

Figure 4. D block when stride $s = 2$.

calculated using the composition of convs up to the current point, as described in the previous paragraph. This serves as an upper bound on r, and, in practice, we choose dilation rates much lower than the upper bounds.

The relationship between the field-of-view and the input image size highly influences the accuracy of the model. For example, if we use the ResNet [16] 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.

3.2. Dilated block

Our dilated block (D block) takes inspiration from the Y block of RegNet [36], also known as the SE-ResNeXt block [20]. 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

Block structure	ms
Y block	1.0
D block(1,1)	1.1
D block(1,4)	1.2
D block(1,10)	1.2

Table 1. Block Latency. The manual split and concatenation brings a 0.1 ms delay and the use of dilated conv brings another 0.1ms delay. For timing purposes, we use $w = 256$ and $g = 16$ for all blocks, and $1 \times 256 \times 64 \times 128$ as input. Sec. 4.7 explains the timing setup in more detail.

Operator	d1, d2	Stride	#Channels	#Repeat
3x3 conv	-	2	32	1
D block	1, 1	2	48	1
D block	1, 1	2	128	3
D block	1, 1	2	256	2
D block	1, 2	1	256	1
D block	1, 4	1	256	4
D block	1, 14	1	256	6
D block	1, 14	1	320	1

Table 2. 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.
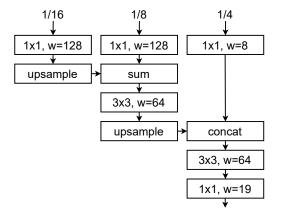

Figure 5. Decoder. w shows the number of output channels. All convs except the final one are followed by BatchNorm [23] and ReLU.

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. 3a shows the Y block. Fig. 3b shows our D block. When $d1 = d2 = 1$, the D block is 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. 4 shows the D block when stride = 2. Similar to the ResNet D-variant [17], we apply a 2×2 average pooling on the shortcut branch when the block's stride = 2. BatchNorm [23] 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 reduction ratio of $1/4$.

Modern deep learning frameworks support group conv where each group applies the same dilation rate. Since the D block uses different dilation rates for different groups, we have to manually split the input, apply conv, and then concatenate. We use $(d1, d2)$ to denote the dilation rates in a D block. In Tab. 1, we show that the manual split and concatenation and the use of dilated conv impact the speed. As a result, the Y block is slightly faster than the D block even though they have the same FLOP complexity and parameter count. When we use D block$(1, 1)$ in practice, we do not have to manually split and concatenate since both branches use the same dilation rate.

3.3. Backbone

Our backbone is built by repeating the D block, in a style similar to RegNet. 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 $1/32$. We increase the dilation rates for the thirteen stride 1 blocks at $1/16$: one $(1, 1)$, one $(1, 2)$, four $(1, 4)$, and seven $(1, 14)$. As a shorthand, we denote the dilation rates as $(1, 1) + (1, 2) + 4 * (1, 4) + 7 * (1, 14)$. We use dilation rates $(1, 1)$ for all the other blocks, making them equivalent to the Y block, except for the 2×2 avg pooling when stride = 2. In a format similar to EfficientNetV2 [42], we display the backbone of RegSeg in Tab. 2. Choosing the right dilation rates for the last thirteen blocks is nontrivial, and we experiment with the dilation rates in Sec. 4.4.

3.4. 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. 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 [23] and ReLU. The decoder is shown in Fig. 5. This simple decoder performs better than many existing decoders that have similar latencies. We experiment with different decoder designs in Sec. 4.5.

4. Experiments

4.1. Datasets

Cityscapes [8] 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 [24, 33, 48], 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 $= 0.05$, weight decay $= 0.0001$, but we do not decay BatchNorm parameters. We use the poly learning rate scheduler that sets the current learning rate to the initial learning rate multiplied by $(1 - \frac{cur\_iter}{total\_iter})^{0.9}$. We also apply a linear warmup [15] from $0.1lr$ to lr for the first 3000 iterations. During training, we apply random horizontal flipping, random scaling of $[400, 1600]$, and random cropping of 768×768. We use a reduced set of RandAug [9] 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 [51] with class uniform percent $= 0.5$. We use cross-entropy loss and batch size $= 8$. RegSeg is trained from randomly initialized weights using PyTorch's [34] default initialization. We train for 1000 epochs on a single T4 GPU. To speed up training without any loss in accuracy, we use mixed precision training. When submitting to the test server, we train on the trainval set and additionally use 1024×1024 crop size and Online Hard Example Mining loss [39] (OHEM). OHEM loss, also known as bootstrapped loss, averages the pixel losses that are over 0.3, or averages the top $1/16$ pixel losses if the original proportion is less than $1/16$.

When doing ablation studies, we train for only 500 epochs. To halve the number of CPUs required when doing ablation studies, we store the images at half resolution on

truck	bus	train	mIoU	mIOU^R
68.11	73.55	35.77	72.71	75.26
77.35	79.05	53.18	74.63	75.52
73.17	77.78	58.77	74.65	75.54
71.31	81.73	74.27	75.46	75.41

Table 3. Reproducibility. The classes truck, bus, and train all vary a lot across runs while the other non-shown classes do not vary as much. By removing these three classes from the metric, we can achieve lower variation.

disk and resize them back to full resolution when loading unless specified otherwise. We store and load the images at full resolution when comparing against DDRNet-23 [18] and when we submit to the test server.

On CamVid, the training setting is similar to that in Cityscapes. Because we do not have ImageNet pretraining and CamVid is small, we use Cityscapes pretrained models. We use random horizontal flipping, random scaling of $[288, 1152]$, and random cropping of 720×960 with batch size 12. We do not use RandAug or class uniform sampling. We train for 200 epochs.

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.

Even after this careful reproducibility procedure, the variation on Cityscapes is still very larger, as shown Tab. 3, where we train the same model 4 times and measure the standard deviation. This is likely because of class imbalance and an out-of-distribution validation set. Class imbalance exists because some classes show up less and some classes have smaller instances. The validation set is out-of-distribution because the images in the training set and validation set are taken from different cities. These problems could potentially be alleviated by pretraining on a larger dataset, but we focus on training from scratch. Unsatisfied by the large variation, we inspect the IOU of individual classes and find that a few classes (truck, bus, and train) vary a lot more than the other classes. We produce a new metric called reduced mIOU (mIOU^R), where we remove these

Row	Dilation rates	Field-of-view	mIOU^R
1	(1,1)+(1,2)+4*(1,4)+7*(1,14)	3807	75.85
2	(1,1)+(1,2)+(1,4)+(1,6)+(1,8)+(1,10)+7*(1,12)	3743	75.75
3	(1,1)+(1,2)+(1,4)+(1,6)+(1,8)+(1,10)+7*(1,3,6,12)	3743	75.69
4	(1,1)+(1,2)+(1,4)+(1,6)+(1,8)+8*(1,10)	3295	75.58
5	(1,1)+(1,2)+6*(1,4)+5*(1,6,12,18)	3807	75.54
6	(1,1)+(1,2)+(1,4)+10*(1,6)	2207	75.53
7	(1,1)+(1,2)+(1,4)+(1,6)+(1,8)+(1,10)+(1,12)+6*(1,14)	4127	75.45
8	5*(1,4)+8*(1,10)	3263	75.44
9	(1,1)+(1,2)+(1,4)+(1,6)+(1,8)+(1,10)+7*(1,4,8,12)	3743	75.17
10	Dilated RegNetY-600MF 8*(1,1)+3*(2,2)	607	73.25

Table 4. Backbone Ablation Studies. Our best backbone has small dilation rates early on and large dilation rates later.

Decoder	mIOU^R
Sec. 3.4 decoder	75.84
sum+3x3 conv	75.75
concat+Y block	75.70
concat+3x3 conv	75.62
sum+1x1 conv	74.93
LRASPP [19]	74.85
SFNetDecoder [26]	74.80
BiSeNetDecoder [48]	74.68

Table 5. Decoder Ablation Studies. Our best decoder performs better than common alternatives.

Random Resize	Random Crop	mIOU^R
[400,1600]	768x1536	76.25
[400,1600]	1024x1024	76.15
[400,1600]	512x1024	76.10
[400,1600]	768x768	75.82
[512,2048]	1024x1024	75.78
[512,2048]	768x768	75.17
[512,2048]	512x1024	75.03

Table 6. Random Resize and Random Crop. [400, 1600] random resize performs better than [512, 2048], and 1024x1024 better than other common crop settings.

three classes before taking the mean. As shown Tab. 3, its variation is much smaller, allowing our ablation studies to hold some weight.

4.4. Backbone ablation studies

In Tab. 4, we experiment with the dilation rates in the last 13 blocks and show their field-of-view while fixing everything else. We always keep $d1 = 1$ to preserve the local details. We see that (a) having 4 branches (row 3, 5, and 9) is not necessarily better than 2 branches, (b) dilation rates should be small early on (row 4 vs row 8), and (c) the best field-of-view is around 3800 (row 1 and 2 vs row 4, 6, and 7). Our best backbone is 2.6% better than the dilated RegNetY-600MF [36] with output stride = 16.

4.5. Decoder ablation studies

In Tab. 5, we experiment with the decoder design while fixing the backbone architecture. We take the backbone's last 1/4, 1/8, and 1/16 feature maps as inputs. We always apply a $[1 \times 1, 128]$ conv to 1/16 and bilinearly upsample it by a factor of 2. We apply a $[1 \times 1, 128]$ conv to 1/8 if it will be summed with 1/16, or a $[1 \times 1, 32]$ conv if it will be concatenated with 1/16. After 1/8 and 1/16 are either summed or concatenated, we can use a 1×1 conv, a 3×3 conv, or a Y block, all with output channels = 128, to further decode the features, before the final 1×1 conv to 19 channels. All convs are immediately followed by Batch-Norm [23] and ReLU, except the final conv to 19 channels. All these decoders have output stride = 8 except for our best decoder. We see that 3×3 conv is 0.8% better than 1×1 conv when using summation. Summation and concatenation are similar, and 3×3 conv and the Y block are similar. The best decoder is the one described in Sec. 3.4, which additionally uses 1/4 features. Existing decoders [19, 26, 48] perform much worse than our best decoder, potentially because they are designed for backbones that do not have a large field-of-view.

4.6. Training technique ablation studies

In Tab. 6, we experiment with random resize and random crop hyperparameters. We find that the popular [512, 2048] random resize performs worse than the [400, 1600] resize, across many crop sizes. [400, 1600] is centered around 1000, while [512, 2048] is centered around 1280, so the [400, 1600] random resize is more aligned with the our validation size of 1024. This is in line with the recent Copy-Paste paper [14], where they use $[0.1 * valsize, 2.0 * valsize]$ random resize on COCO [29], and the FixRes [43] paper,
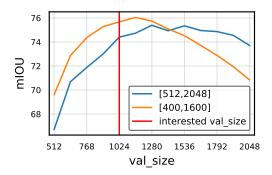
Figure 6. mIOUs under different val sizes. $[400, 1600]$ performs better on smaller val sizes, while $[512, 2048]$ performs better on larger val sizes

$[400, 1600]$	OHEM	1000 epochs	mIOU^R
			76.70
✓			77.16
✓	✓		77.55
✓		✓	78.3

Table 7. More training settings. We see that $[400, 1600]$, OHEM, and longer training are helpful.

where they fix the train-test discrepancy by using a larger test size or a smaller train size on ImageNet. In Fig. 6, we show that $[400, 1600]$ performs better with smaller validation sizes, while $[512, 2048]$ performs better with larger validation sizes.

In Tab. 7, we combine the best backbone and decoder that we have found in the previous ablation studies and experiment with more training settings. For the experiments in this table, we use images stored at full resolution instead of half resolution. We reconfirm that $[400, 1600]$ resize is helpful. Training for 1000 epochs instead of 500 epochs and using OHEM loss give huge performance gains. We do not use 1024×1024 random crop and OHEM loss except when we submit to the test server.

4.7. Timing

We time RegSeg using a single T4 GPU with mixed precision. We use PyTorch 1.9 [34] with CUDA 10.2. The input size is $1 \times 3 \times 1024 \times 2048$ on Cityscapes, and $1 \times 3 \times 720 \times 960$ 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().

4.8. Comparison on CamVid

As shown in Tab. 8, RegSeg achieves 80.9 mIOU on CamVid test set at 70 FPS. It outperforms the previous SOTA DDRNet-23 by 0.8%, and BiSeNetV2-L by 2.4%.

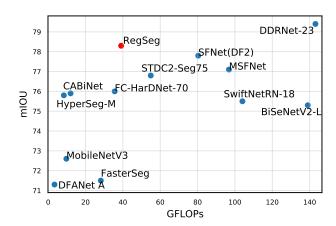Figure 7. GFLOPs vs mIOU on the Cityscapes test set. Our model is in red, while other models are in blue. We achieve SOTA flops-accuracy trade-off.

Model	Extra data	mIOU	FPS
STDC2-Seg [13]	IM	73.9	152.2
GAS [28]	-	72.8	153.1
CAS [49]	-	71.2	169
SFNet(DF2) [26]	IM	70.4	134
SFNet(ResNet-18) [26]	IM	73.8	36
MSFNet [40]	IM	75.4	91
HyperSeg-S [32]	IM	78.4	38.0
TD4-PSP18 [21]	IM	72.6	25.0
VideoGCRF [2]	C	75.2	-
BiSeNetV2 [47]	C	76.7	124
BiSeNetV2-L [47]	C	78.5	33
CCNet3D [22]	C	79.1	-
DDRNet-23 [18]	C	80.1±0.4	94
RegSeg	C	80.9±0.07	70

Table 8. Accuracy and speed comparison on CamVid, IM: ImageNet, C: Cityscapes

The results show that RegSeg may generalize better than DDRNet-23. Note that two models' FPS are not directly comparable since they might have used different timing setups.

4.9. Comparison on Cityscapes

We compare against other real-time models, whether they are pretrained on ImageNet or not. As shown in Fig. 1 and Fig. 7, RegSeg achieves the best parameter-accuracy and flops-accuracy trade-offs. In Tab. 9, we show the accuracy and speed comparison on Cityscapes. Again, note that the FPS across models are not directly comparable. RegSeg outperforms HarDNet [3], which is the previous SOTA model without extra data, by 1.5%, and outperforms SFNet(DF2) [26], which has the best peer-reviewed results, by 0.5%. RegSeg also outperforms the popular BiSeNetV2-

Model	Extra data	val mIOU	test mIOU	FPS	Resolution	Params (M)	GFLOPs
CAS [49]	IM	71.6	70.5	108	768x1536	-	-
DFANet A [24]	IM	-	71.3	100	1024x1024	7.8	3.4
FasterSeg [6]	None	73.1	71.5	163.9	1024x2048	4.4	28.2
GAS [28]	IM	72.4	71.8	108.4	769x1537	-	-
MobileNetV3 [19]	None	72.36	72.6	-	1024x2048	1.51	9.74
HMSeg [25]	None	-	74.3	83.2	768x1536	-	-
TD4-Bise18 [21]	IM	75.0	74.9	47.6	1024x2048	-	-
BiSeNetV2-L [47]	None	75.8	75.3	47.3	512x1024	4.59	139
DF2-Seg2 [27]	IM	76.9	75.3	56.3	1024x2048	-	-
SwiftNetRN-18 [33]	IM	-	75.5	39.9	1024x2048	11.8	104
HyperSeg-M [32]	IM	76.2	75.8	36.9	512x1024	10.3	8.4
CABiNet [46]	IM	76.6	75.9	76.5	1024x2048	2.64	12
FC-HarDNet-70 [3]	None	77.7	76.0	53	1024x2048	4.12	35.6
STDC2-Seg75 [13]	IM	77.0	76.8	97.0	768x1536	16.1	54.9
MSFNet [40]	IM	-	77.1	41	1024x2048	-	96.8
SFNet(DF2) [26]	IM	-	77.8	53	1024x2048	17.9	80.4
DDRNet-23 [18]	IM	79.1±0.3	79.4	37.1	1024x2048	20.1	143.1
RegSeg	None	78.13±0.48	78.3	30	1024x2048	3.34	39.1

Table 9. Accuracy and speed comparison on Cityscapes, IM: ImageNet

Model	FPS	val mIOU	Params (M)	GFLOPs
DDRNet-23	30	77.59 ± 0.09	20.1	143.1
RegSeg	30	78.13 ± 0.48	3.34	39.1

Table 10. Comparison against DDRNet-23 under the exact same training setting. FPS of both models are calculated using Sec. 4.7. RegSeg achieves higher accuracy and better parameters and flops efficiency, while running at similar speeds.

L [47] by 3.0%, and MobileNetV3+LRASPP [19] by 5.7%. Although the concurrent work of DDRNet-23 [18] achieves higher test set accuracy than our model, we argue that their success is due to their ImageNet pretraining, while also noting that RegSeg has much lower params and flops. To illustrate our point, we show that RegSeg outperforms DDRNet-23 when trained using the exact same training settings. In Tab. 10, we train each model 3 times and display the mean ± one standard deviation. Using the training settings explained in Sec. 4.2, RegSeg outperforms DDRNet-23 by 0.5% and has better parameters and flops efficiency. The FPS of both models are calculated using Sec. 4.7.

5. 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 introduce the novel D block that can increase the field-of-view without losing the local details, and experiment with the dilation rates in RegSeg's backbone. We also pro-

pose a lightweight decoder that performs better than common alternatives. Together, RegSeg pushes the state-of-the-art on real-time semantic segmentation datasets such as Cityscapes and CamVid, without ImageNet pretraining.

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. 1, 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. 3, 7, 8

[4] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 1, 2

[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, 3, 4

[6] Wuyang Chen, Xinyu Gong, Xianming Liu, Qian Zhang, Yuan Li, and Zhangyang Wang. Fasterseg: Searching for faster real-time semantic segmentation. In *ICLR*, 2019. 8

[7] 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

[8] 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. 1, 5

[9] 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. 5

[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

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

[12] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, June 2010. 1

[13] 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. 3, 7, 8

[14] Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D Cubuk, Quoc V Le, and Barret Zoph. Simple copy-paste is a strong data augmentation method for instance segmentation. In *CVPR*, pages 2918–2928, 2021. 6

[15] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noord-huis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large mini-batch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 5

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 1, 3

[17] 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

[18] 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. 2, 3, 5, 7, 8

[19] 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. 1, 2, 3, 6, 8

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

[21] 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, 8

[22] 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

[23] 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, 6

[24] Hanchao Li, Pengfei Xiong, Haoqiang Fan, and Jian Sun. Dfanet: Deep feature aggregation for real-time semantic segmentation. In *CVPR*, June 2019. 5, 8

[25] Peike Li, Xuanyi Dong, Xin Yu, and Yi Yang. When humans meet machines: Towards efficient segmentation networks. In *BMVC*, 2020. 8

[26] 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, 3, 6, 7, 8

[27] Xin Li, Yiming Zhou, Zheng Pan, and Jiashi Feng. Partial order pruning: for best speed/accuracy trade-off in neural architecture search. In *CVPR*, pages 9145–9153, 2019. 8

[28] 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, 8

[29] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755. Springer, 2014. 6

[30] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *CVPR*, pages 82–92, 2019. 2

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

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

[33] 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. 5, 8

[34] 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. 5, 7

[35] 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. 2

[36] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *CVPR*, pages 10428–10436, 2020. 2, 3, 6

[37] Abdullah Rashwan, Xianzhi Du, Xiaoqi Yin, and Jing Li. Dilated spinenet for semantic segmentation. *arXiv preprint arXiv:2103.12270*, 2021. 2

[38] 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

[39] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *CVPR*, pages 761–769, 2016. 5

[40] 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, 8

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

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

[43] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Hervé Jégou. Fixing the train-test resolution discrepancy. In *NeurIPS*, 2019. 6

[44] 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. 3

[45] 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

[46] Michael Ying Yang, Saumya Kumaar, Ye Lyu, and Francesco Nex. Real-time semantic segmentation with context aggregation network. *ISPRS Journal of Photogrammetry and Remote Sensing*, 178:124–134, 2021. 8

[47] 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. 3, 7, 8

[48] 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. 3, 5, 6

[49] 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, 8

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

[51] 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. 5