DeeplabV3+

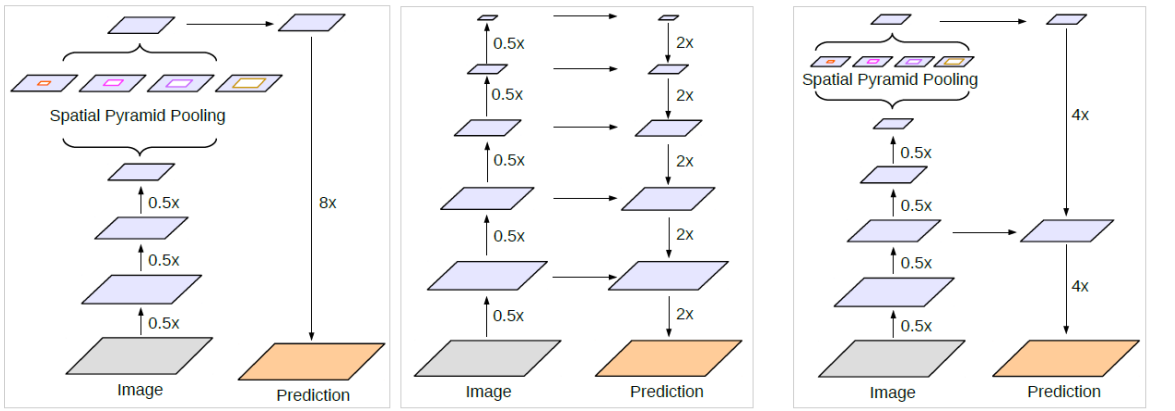
Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation

Introduction

Motivation

Two Approaches

1. spatial pyramid pooling module → captures rich contextual information at different resolution
2. encoder-decoder structure → obtain sharp object boundaries.



(a) Spatial Pyramid Pooling (b) Encoder-Decoder (c) Encoder-Decoder with Atrous Conv

1. spatial pyramid pooling module

- Deeplab applies Atrous Spatial Pyramid Pooling (Dilated Conv with different rates)
- PSPNet performs pooling operations at different grid scales

But, detailed information related to object boundaries is missing!!

How to solve? → **applying the "atrous convolution" to extract denser feature maps**

But, computationally expensive! (Why?:

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Dilated Conv w/o reducing feature map size (e.g., ASPP) but have the same depth and layers?

2. encoder-decoder models are faster and gradually recover sharp object boundaries

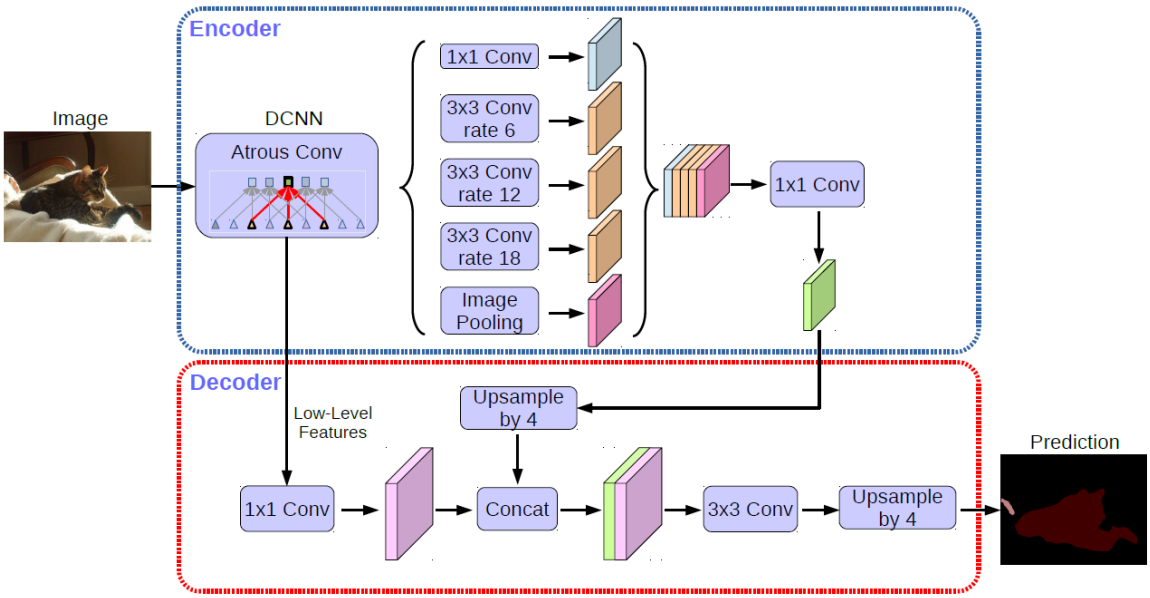
Thus, combine 1. and 2.

Contributions (Deeplab V3+)

- Combine encoder-decoder structure and ASPP (Deeplab)
- Arbitrarily control the resolution of extracted encoder features by atrous convolution
- Introduced Xception model and depthwise separable convolution to both ASPP module and decoder module (faster and stronger!)

https://github.com/tensorflow/models/tree/master/research/deeplab

Methods



Atrous Conv

- Control the resolution of features

(Q. Isn't precisely that Atrous Conv, Dilated Conv without Pooling keeps high resolution features?)

(Q. Isn't that just conv without pooling many times will have same effect?)

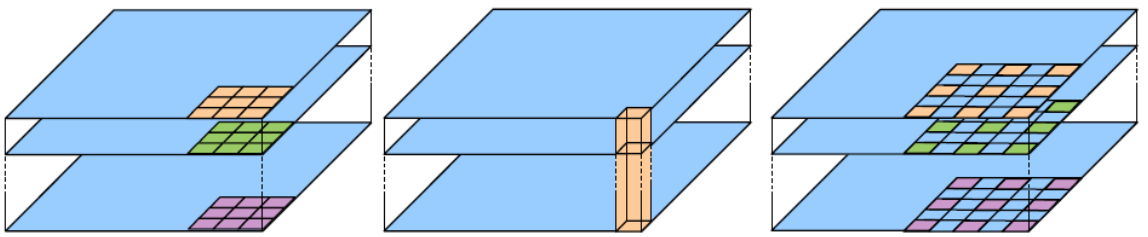
- Increase filter's FoV

$$y[i] = \sum_{k} x[i + r \cdot k] w[k]$$

Depthwise separable convolution

depthwise convolution followed by a pointwise convolution (i.e., 1×1 convolution)

- Reduces computation complexity
- Here applies atrous separable convolution



(a) Depthwise conv. (b) Pointwise conv. (c) Atrous depthwise conv.

DeepLabv3 as encoder

Term "output stride": as the ratio of input image spatial resolution to the final output resolution

Typically OS = 32 for image classification

- OS = 8 or 16 (and apply atrous convolution)

(e.g., rate = 2 and rate = 4 to the last two blocks respectively for output stride = 8).

- Atrous Spatial Pyramid Pooling module
- encoder output = last feature map before logits as in the original DeepLabv3

Proposed decoder

DeeplabV3 → bilinearly upsampled by a factor of 16

DeeplabV3+ → bilinearly upsampled by a factor of 4 + concat w/ low-level features

(1) bilinearly upsampled by a factor of 4

(2) concatenated with the corresponding low-level features

(3) apply another 1 × 1 convolution on the low-level features to reduce the number of channels

(4) apply a few 3 × 3 convolutions to refine the features

(5) another simple bilinear upsampling by a factor of 4

e.g., Conv2 before striding in ResNet-101

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

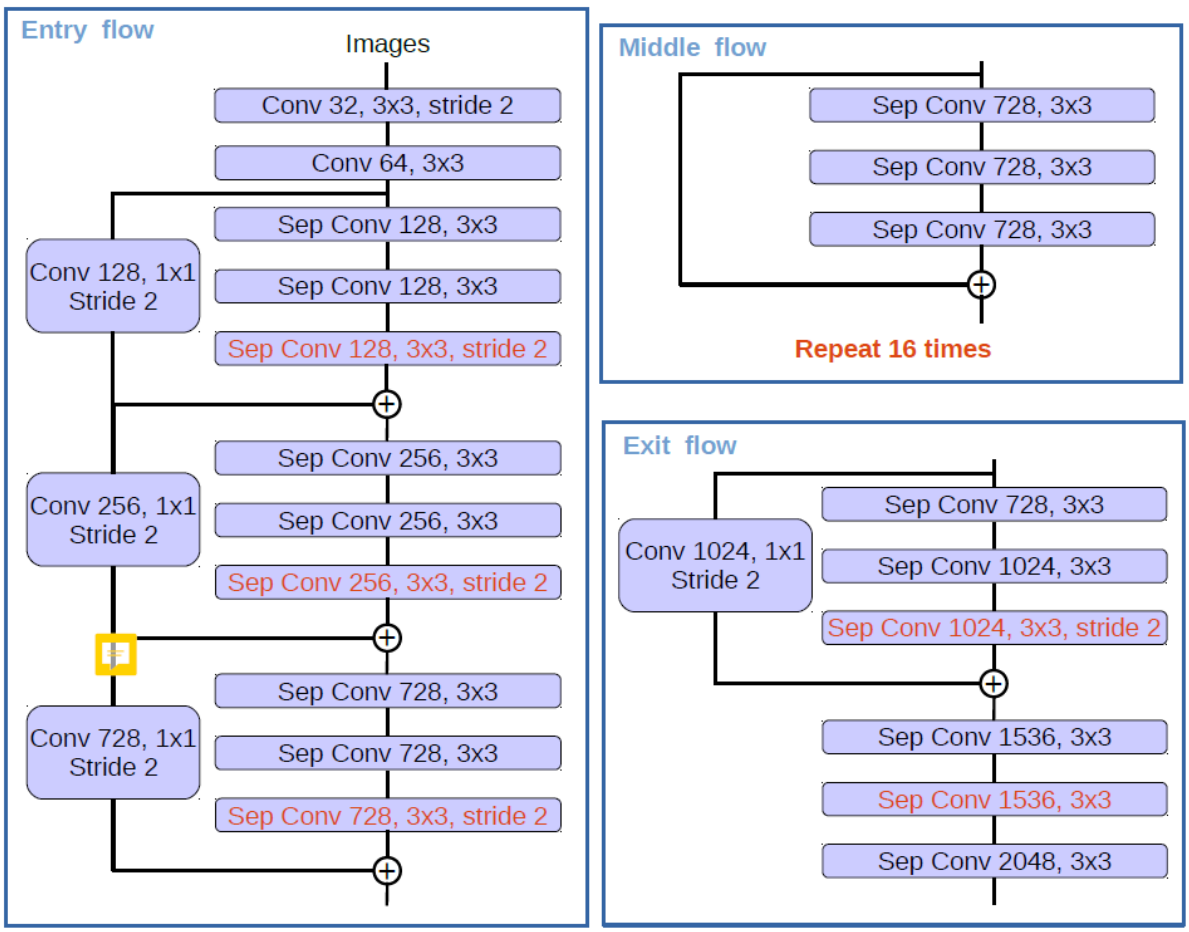
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Modifed Xception

(1) more layers

(2) all the max pooling operations are replaced by depthwise separable convolutions with striding

(3) extra batch normalization and ReLU are added after each 3 × 3 depthwise convolution, similar to MobileNet



Experimental Evaluation

- ImageNet-1k pretrained ResNet-101 or modified aligned Xception
- PASCAL VOC 2012 (21 classes mIOU)
- crop size 513 × 513 (OS=16)

Decoder Design Choices

When employing output stride = 16, naive decoder is 1.2% better than not using this naive decoder during training (i.e., downsampling groundtruth during training)

In the decoder module,

(1) the 1×1 convolution used to reduce the channels of the low-level feature map from the encoder module

- employ [3× 3, 256] and Conv2 features from ResNet-101
- Choose [1 × 1, 48] for channel reduction.

Channels	8	16	32	48	64
mIOU	77.61%	77.92%	78.16%	**78.21%**	77.94%

Table 1. PASCAL VOC 2012 *val* set. Effect of decoder 1×1 convolution used to reduce the channels of low-level feature map from the encoder module. We fix the other components in the decoder structure as using $[3 \times 3, 256]$ and Conv2.

(2) the 3 × 3 convolution used to obtain sharper segmentation results

- Do after concatenating the Conv2 feature map (before striding) with DeepLabv3 feature map
- more effective to employ two 3×3 convolution with 256 filters than using simply one or three convolutions.
- Changing the number of filters from 256 to 128 or the kernel size from 3 × 3 to 1×1 degrades performance
- Using both Conv2 and Conv3 feature maps didn't help

Features Conv2	Features Conv3	3×3 Conv Structure	mIOU
✓		$[3 \times 3, 256]$	78.21%
✓		$[3 \times 3, 256] \times 2$	**78.85%**
✓		$[3 \times 3, 256] \times 3$	78.02%
✓		$[3 \times 3, 128]$	77.25%
✓		$[1 \times 1, 256]$	78.07%
✓	✓	$[3 \times 3, 256]$	78.61%

Table 2. Effect of decoder structure when fixing $[1 \times 1, 48]$ to reduce the encoder feature channels. We found that it is most effective to use the Conv2 (before striding) feature map and two extra $[3 \times 3, 256]$ operations. Performance on VOC 2012 *val* set.

Note: Proposed DeepLabv3+ model has output stride = 4. Not pursue further denser output feature map (i.e., output stride < 4) given the limited GPU resources!?

(3) Further enconding

ResNet-101 as Network Backbone

- Atrous Conv → able to obtain features at different resolutions during training and evaluation
- extracting denser feature maps during evaluation (i.e., eval output stride = 8) and adopting multi-scale inputs increases performance

Encoder train OS	Encoder eval OS	Decoder	MS	Flip	mIOU	Multiply-Adds
16	16				77.21%	81.02B
16	8				78.51%	276.18B
16	8		✓		79.45%	2435.37B
16	8		✓	✓	79.77%	4870.59B
16	16	✓			78.85%	101.28B
16	16	✓	✓		80.09%	898.69B
16	16	✓	✓	✓	80.22%	1797.23B
16	8	✓			79.35%	297.92B
16	8	✓	✓		80.43%	2623.61B
16	8	✓	✓	✓	80.57%	5247.07B
32	32				75.43%	52.43B
32	32	✓			77.37%	74.20B
32	16	✓			77.80%	101.28B
32	8	✓			77.92%	297.92B

Table 3. Inference strategy on the PASCAL VOC 2012 *val* set using *ResNet-101*. **train OS**: The *output stride* used during training. **eval OS**: The *output stride* used during evaluation. **Decoder**: Employing the proposed decoder structure. **MS**: Multi-scale inputs during evaluation. **Flip**: Adding left-right flipped inputs.

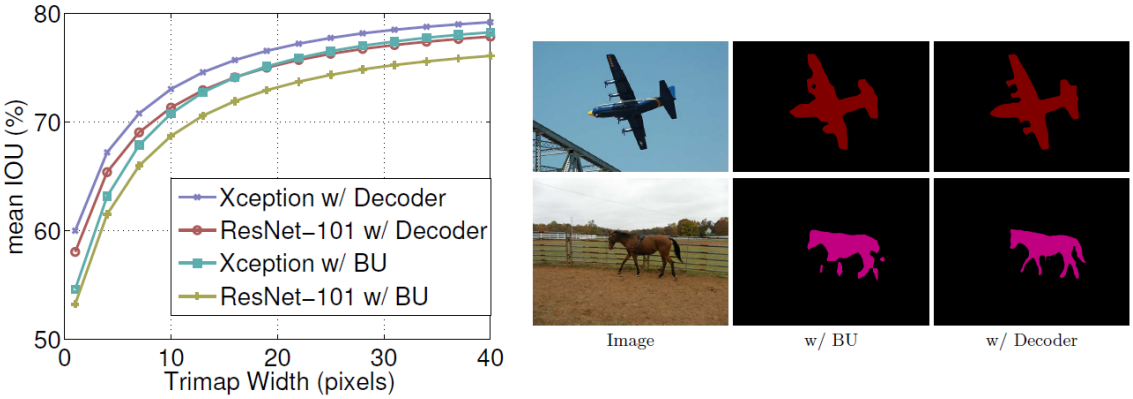
Improvement along Object Boundaries

Encoder train OS	Encoder eval OS	Decoder	MS	Flip	SC	COCO	JFT	mIOU	Multiply-Adds
16	16							79.17%	68.00B
16	16		✓					80.57%	601.74B
16	16		✓	✓				80.79%	1203.34B
16	8							79.64%	240.85B
16	8		✓					81.15%	2149.91B
16	8		✓	✓				81.34%	4299.68B
16	16	✓						79.93%	89.76B
16	16	✓	✓					81.38%	790.12B
16	16	✓	✓	✓				81.44%	1580.10B
16	8	✓						80.22%	262.59B
16	8	✓	✓					81.60%	2338.15B
16	8	✓	✓	✓				81.63%	4676.16B
16	16	✓			✓			79.79%	54.17B
16	16	✓	✓	✓	✓			81.21%	928.81B
16	8	✓			✓			80.02%	177.10B
16	8	✓	✓	✓	✓			81.39%	3055.35B
16	16	✓			✓	✓		82.20%	54.17B
16	16	✓	✓	✓	✓	✓		83.34%	928.81B
16	8	✓			✓	✓		82.45%	177.10B
16	8	✓	✓	✓	✓	✓		83.58%	3055.35B
16	16	✓			✓	✓	✓	83.03%	54.17B
16	16	✓	✓	✓	✓	✓	✓	84.22%	928.81B
16	8	✓			✓	✓	✓	83.39%	177.10B
16	8	✓	✓	✓	✓	✓	✓	84.56%	3055.35B

Table 5. Inference strategy on the PASCAL VOC 2012 *val* set when using modified *Xception*. **train OS**: The *output stride* used during training. **eval OS**: The *output stride* used during evaluation. **Decoder**: Employing the proposed decoder structure. **MS**: Multi-scale inputs during evaluation. **Flip**: Adding left-right flipped inputs.

Method	mIOU
Deep Layer Cascade (LC) [82]	82.7
TuSimple [77]	83.1
Large_Kernel_Matters [60]	83.6
Multipath-RefineNet [58]	84.2
ResNet-38_MS_COCO [83]	84.9
PSPNet [24]	85.4
IDW-CNN [84]	86.3
CASIA_IVA_SDN [63]	86.6
DIS [85]	86.8
DeepLabv3 [23]	85.7
DeepLabv3-JFT [23]	86.9
DeepLabv3+ (Xception)	87.8
DeepLabv3+ (Xception-JFT)	89.0

Table 6. PASCAL VOC 2012 *test* set results with top-performing models.



(a) mIOU *vs*. Trimap width (b) Decoder effect

Fig. 5. (a) mIOU as a function of trimap band width around the object boundaries when employing *train output stride* = *eval output stride* = 16. **BU**: Bilinear upsampling. (b) Qualitative effect of employing the proposed decoder module compared with the naive bilinear upsampling (denoted as **BU**). In the examples, we adopt Xception as feature extractor and *train output stride* = *eval output stride* = 16.

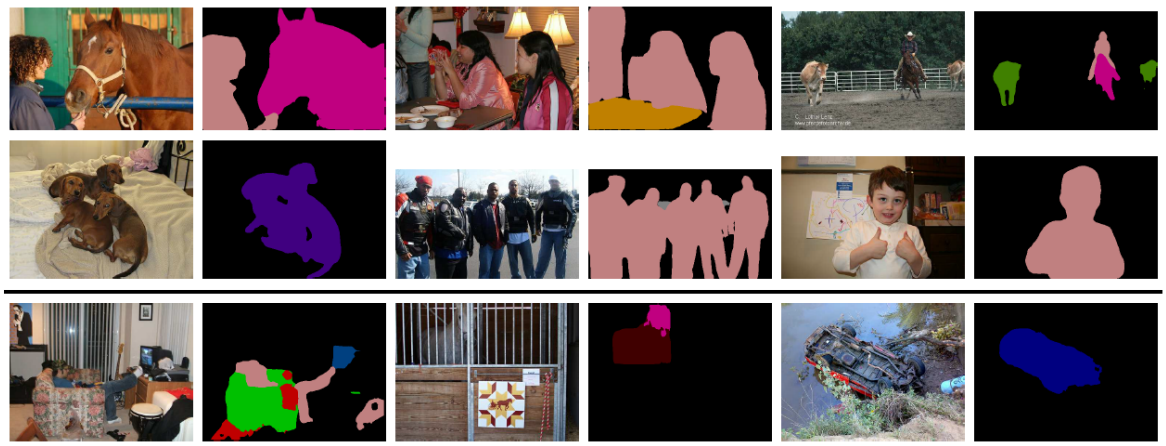


Fig. 6. Visualization results on *val* set. The last row shows a failure mode.

Experimental Results on Cityscapes

Backbone	Decoder	ASPP	Image-Level	mIOU
X-65		✓	✓	77.33
X-65	✓	✓	✓	78.79
X-65	✓	✓		79.14
X-71	✓	✓		79.55

(a) *val* set results

Method	**Coarse**	**mIOU**
ResNet-38 [83]	✓	80.6
PSPNet [24]	✓	81.2
Mapillary [86]	✓	82.0
DeepLabv3	✓	81.3
DeepLabv3+	✓	82.1

(b) *test* set results

Table 7. (a) DeepLabv3+ on the Cityscapes *val* set when trained with *train_fine* set. (b) DeepLabv3+ on Cityscapes *test* set. **Coarse**: Use *train_extra* set (coarse annotations) as well. Only a few top models are listed in this table.