Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation

Liang-Chieh Chen Yukun Zhu George Papandreou Florian Schroff Hartwig Adam
Google Inc.
{lcchen, yukun, gpapan, fschroff, hadam}@google.com

Abstract

Spatial pyramid pooling module or encode-decoder structure are used in deep neural networks for semantic segmentation task. The former networks are able to encode multi-scale contextual information by probing the incoming features with filters or pooling operations at multiple rates and multiple effective fields-of-view, while the latter networks can capture sharper object boundaries by gradually recovering the spatial information. In this work, we propose to combine the advantages from both methods. Specifically, our proposed model, DeepLabv3+, extends DeepLabv3 by adding a simple yet effective decoder module to refine the segmentation results especially along object boundaries. We further explore the Xception model and apply the depthwise separable convolution to both Atrous Spatial Pyramid Pooling and decoder modules, resulting in a faster and stronger encoder-decoder network. We demonstrate the effectiveness of the proposed model on the PASCAL VOC 2012 semantic image segmentation dataset and achieve a performance of 89% on the test set without any post-processing. Our paper is accompanied with a publicly available reference implementation of the proposed models in Tensorflow at `https://github.com/tensorflow/models/tree/master/research/deeplab`.

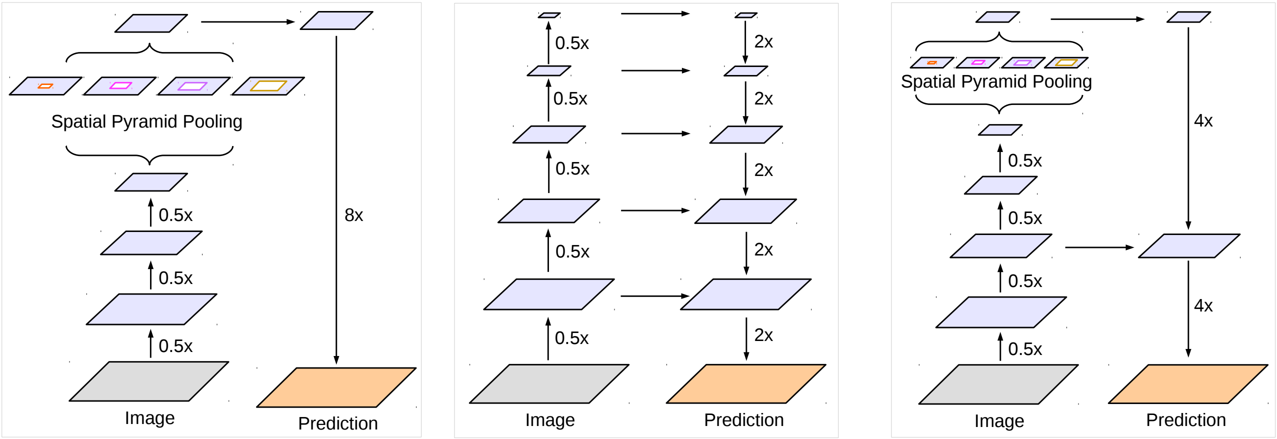
1. Introduction

Semantic segmentation with the goal to assign semantic labels to every pixel in an image [17, 52, 13, 83, 5] is one of the fundamental topics in computer vision. Deep convolutional neural networks [41, 38, 64, 68, 70] based on the Fully Convolutional Neural Network [64, 49] show striking improvement over systems relying on hand-crafted features [28, 65, 36, 39, 22, 79] on benchmark tasks. In this work, we consider two types of neural networks that use spatial pyramid pooling module [23, 40, 26] or encoder-decoder structure [61, 3] for semantic segmentation, where the former one captures rich contextual information by pooling features at different resolution while the latter one is able to obtain sharp object boundaries.

In order to capture the contextual information at multiple scales, DeepLabv3 [10] applies several parallel atrous convolution with different rates (called Atrous Spatial Pyramid Pooling, or ASPP), while PSPNet [81] performs pooling operations at different grid scales. Even though rich semantic information is encoded in the last feature map, detailed information related to object boundaries is missing due to the pooling or convolutions with striding operations within the network backbone. This could be alleviated by applying the atrous convolution to extract denser feature maps. However, given the design of state-of-art neural networks [38, 68, 70, 27, 12] and limited GPU memory, it is computationally prohibitive to extract output feature maps that are 8, or even 4 times smaller than the input resolution. Taking ResNet-101 [27] for example, when applying atrous convolution to extract output features that are 16 times smaller than input resolution, features within the last 3 residual blocks (9 layers) have to be dilated. Even worse, **26** residual blocks (**78** layers!) will be affected if output features that are 8 times smaller than input are desired. Thus, it is computationally intensive if denser output features are extracted for this type of models. On the other hand, encoder-decoder models [61, 3] lend themselves to faster computation (since no features are dilated) in the encoder path and gradually recover sharp object boundaries in the decoder path. Attempting to combine the advantages from both methods, we propose to enrich the encoder module in the encoder-decoder networks by incorporating the multi-scale contextual information.

In particular, our proposed model, called DeepLabv3+, extends DeepLabv3 [10] by adding a simple yet effective decoder module to recover the object boundaries, as illustrated in Fig. 1. The rich semantic information is encoded in the output of DeepLabv3, with atrous convolution allowing one to control the density of the encoder features, depending on the budget of computation resources. Furthermore, the decoder module allows detailed object boundary recovery.

Motivated by the recent success of depthwise separable convolution [67, 71, 12, 31, 80], we also explore this op-



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

Figure 1. We propose to improve DeepLabv3, which employs the spatial pyramid pooling module (a), with the encoder-decoder structure (b). The proposed model, DeepLabv3+, contains rich semantic information from the encoder module, while the detailed object boundaries are recovered by the simple yet effective decoder module. The encoder module allows us to extract features at an arbitrary resolution by applying atrous convolution.

eration and show improvement in terms of both speed and accuracy by adapting the Xception model [12], similar to [60], for the task of semantic segmentation, and applying the atrous separable convolution to both the ASPP and decoder modules. Finally, we demonstrate the effectiveness of the proposed model on PASCAL VOC 2012 semantic segmentation benchmark and attain a performance of 89.0% on the *test* set without any post-processing, setting a new state-of-the-art.

In summary, our contributions are:

- We propose a novel encoder-decoder structure which employs DeepLabv3 as a powerful encoder module and a simple yet effective decoder module.
- In our proposed encoder-decoder structure, one can arbitrarily control the resolution of extracted encoder features by atrous convolution to trade-off precision and runtime, which is not possible with existing encoder-decoder models.
- We adapt the Xception model for the segmentation task and apply depthwise separable convolution to both ASPP module and decoder module, resulting in a faster and stronger encoder-decoder network.
- Our proposed model attains a new state-of-art performance on PASCAL VOC 2012 dataset. We also provide detailed analysis of design choices and model variants.
- We make our Tensorflow-based implementation of the proposed model publicly available at https://github.com/tensorflow/models/tree/master/research/deeplab.

2. Related Work

Models based on Fully Convolutional Networks (FCNs) [64, 49] have demonstrated significant improvement on several segmentation benchmarks [17, 52, 13, 83, 5]. There are several model variants proposed to exploit the contextual information for segmentation [28, 65, 36, 39, 22, 79, 51, 14], including those that employ multi-scale inputs (*i.e*., image pyramid) [18, 16, 58, 44, 11, 9] or those that adopt probabilistic graphical models (such as DenseCRF [37] with efficient inference algorithm [2]) [8, 4, 82, 44, 48, 55, 63, 34, 72, 6, 7, 9]. In this work, we mainly discuss about the models that use spatial pyramid pooling and encoder-decoder structure.

Spatial pyramid pooling: Models, such as PSPNet [81] or DeepLab [9, 10], perform spatial pyramid pooling [23, 40] at several grid scales (including image-level pooling [47]) or apply several parallel atrous convolution with different rates (called Atrous Spatial Pyramid Pooling, or ASPP). These models have shown promising results on several segmentation benchmarks by exploiting the multi-scale information.

Encoder-decoder: The encoder-decoder networks have been successfully applied to many computer vision tasks, including human pose estimation [53], object detection [45, 66, 19], and semantic segmentation [49, 54, 61, 3, 43, 59, 57, 33, 76, 20]. Typically, the encoder-decoder networks contain (1) an encoder module that gradually reduces the feature maps and captures higher semantic information, and (2) a decoder module that gradually recovers the spatial information. Building on top of this idea, we propose to use DeepLabv3 [10] as the encoder module and add a simple yet effective decoder module to obtain sharper segmentations.

Depthwise separable convolution: Depthwise separable convolution [67, 71] or group convolution [38, 78], a powerful operation to reduce the computation cost and number of parameters while maintaining similar (or slightly better) performance. This operation has been adopted in many recent neural network designs [35, 74, 12, 31, 80, 60, 84]. In particular, we explore the Xception model [12], similar to [60] for their COCO 2017 detection challenge submission, and show improvement in terms of both accuracy and speed for the task of semantic segmentation.

3. Methods

In this section, we briefly introduce atrous convolution [30, 21, 64, 56, 8] and depthwise separable convolution [67, 71, 74, 12, 31]. We then review DeepLabv3 [10] which is used as our encoder module before discussing the proposed decoder module appended to the encoder output. We also present a modified Xception model [12, 60] which further improves the performance with faster computation.

3.1. Encoder-Decoder with Atrous Convolution

Atrous convolution: Atrous convolution, a powerful tool that allows us to explicitly control the resolution of features computed by deep convolutional neural networks and adjust filter's field-of-view in order to capture multi-scale information, generalizes standard convolution operation. In particular, in the case of two-dimensional signals, for each location $\boldsymbol{i}$ on the output feature map $\boldsymbol{y}$ and a convolution filter $\boldsymbol{w}$, atrous convolution is applied over the input feature map $\boldsymbol{x}$ as follows:

$$\boldsymbol{y}[\boldsymbol{i}] = \sum_{\boldsymbol{k}} \boldsymbol{x}[\boldsymbol{i} + r \cdot \boldsymbol{k}]\boldsymbol{w}[\boldsymbol{k}] \tag{1}$$

where the atrous rate r determines the stride with which we sample the input signal. We refer interested readers to [9] for more details. Note that standard convolution is a special case in which rate $r = 1$. The filter's field-of-view is adaptively modified by changing the rate value.

Depthwise separable convolution: Depthwise separable convolution, factorizing a standard convolution into a *depthwise convolution* followed by a *pointwise convolution* (*i.e.*, 1×1 convolution), drastically reduces computation complexity. Specifically, the depthwise convolution performs a spatial convolution independently for each input channel, while the pointwise convolution is employed to combine the output from the depthwise convolution. In the TensorFlow [1] implementation of depthwise separable convolution, atrous convolution has been supported in the depthwise convolution (*i.e.*, the spatial convolution). In this work, we refer the resulting convolution as *atrous separable convolution*, and found that atrous separable convolution significantly reduces the computation complexity of proposed model while maintaining similar (or better) performance.

DeepLabv3 as encoder: DeepLabv3 [10] employs atrous convolution [30, 21, 64, 56] to extract the features computed by deep convolutional neural networks at an arbitrary resolution. Here, we denote *output stride* as the ratio of input image spatial resolution to the final output resolution (before global pooling or fully-connected layer). For the task of image classification, the spatial resolution of the final feature maps is usually 32 times smaller than the input image resolution and thus *output stride* $= 32$. For the task of semantic segmentation, one can adopt *output stride* $= 16$ (or 8) for denser feature extraction by removing the striding in the last one (or two) block(s) and applying the atrous convolution correspondingly (*e.g.*, we apply $rate = 2$ and $rate = 4$ to the last two blocks respectively for *output stride* $= 8$). Additionally, DeepLabv3 augments the Atrous Spatial Pyramid Pooling module, which probes convolutional features at multiple scales by applying atrous convolution with different rates, with the image-level features [47]. We use the last feature map before logits in the original DeepLabv3 as the encoder output in our proposed encoder-decoder structure. Note the encoder output feature map contains 256 channels and rich semantic information. Besides, one could extract features at an arbitrary resolution by applying the atrous convolution, depending on the computation budget.

Proposed decoder: The encoder features from DeepLabv3 are usually computed with *output stride* $= 16$. In the work of [10], the features are bilinearly upsampled by a factor of 16, which could be considered a naive decoder module. However, this naive decoder module may not successfully recover object segmentation details. We thus propose a simple yet effective decoder module, as illustrated in Fig. 2. The encoder features are first bilinearly upsampled by a factor of 4 and then concatenated with the corresponding low-level features [25] from the network backbone that have the same spatial resolution (*e.g.*, Conv2 before striding in ResNet-101 [27]). We apply another 1×1 convolution on the low-level features to reduce the number of channels, since the corresponding low-level features usually contain a large number of channels (*e.g.*, 256 or 512) which may outweigh the importance of the rich encoder features (only 256 channels in our model) and make the training harder. After the concatenation, we apply a few 3×3 convolutions to refine the features followed by another simple bilinear upsampling by a factor of 4. We show in Sec. 4 that using *output stride* $= 16$ for the encoder module strikes the best trade-off between speed and accuracy. The performance is marginally improved when using *output stride* $= 8$ for the encoder module at the cost of extra computation complexity.

3.2. Modified Aligned Xception

The Xception model [12] has shown promising image classification results on ImageNet [62] with fast computation. More recently, the MSRA team [60] modifies the Xception model (called Aligned Xception) and further pushes the performance in the task of object detection. Motivated by

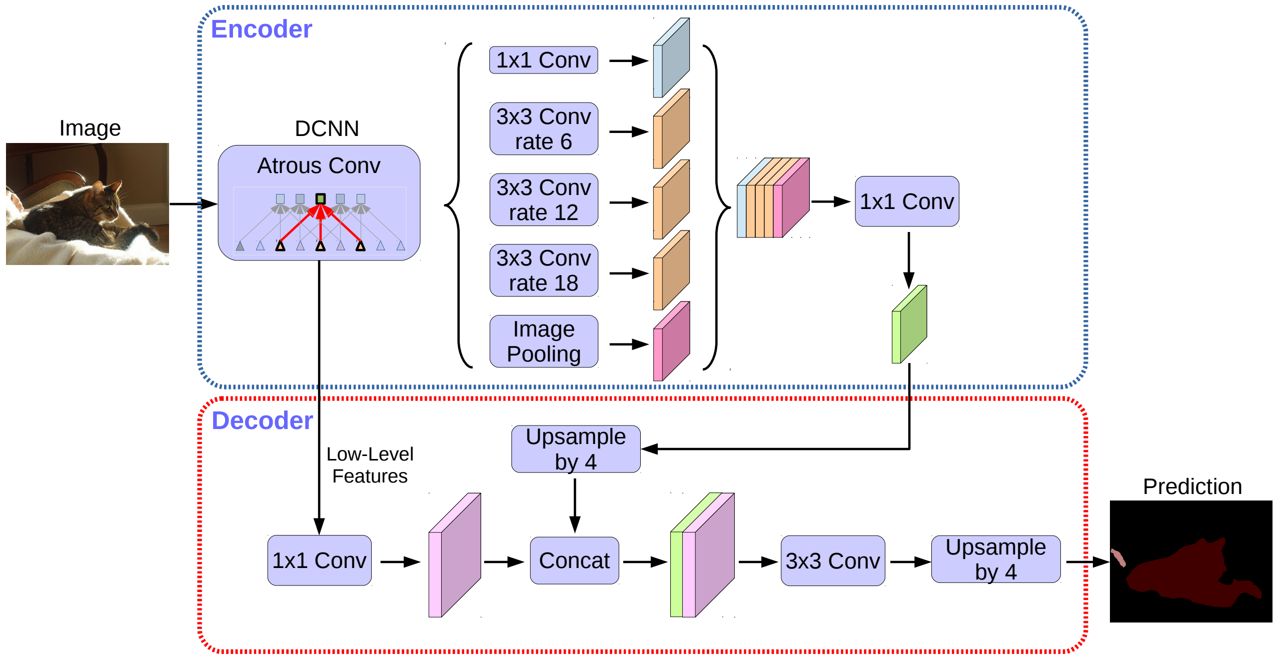


Figure 2. Our proposed DeepLabv3+ extends DeepLabv3 by employing a encoder-decoder structure. The encoder module encodes multi-scale contextual information by applying atrous convolution at multiple scales, while the simple yet effective decoder module refines the segmentation results along object boundaries.

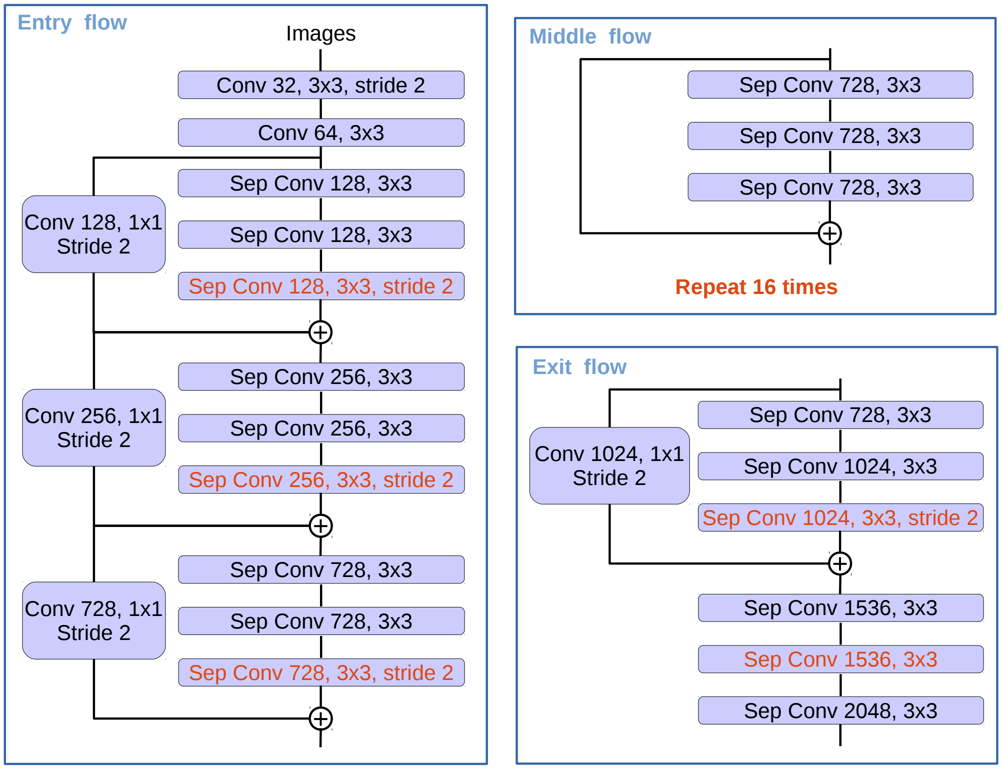


Figure 3. The Xception model is modified as follows: (1) more layers (same as MSRA's modification except the changes in Entry flow), (2) all the max pooling operations are replaced by depthwise separable convolutions with striding, and (3) extra batch normalization and ReLU are added after each 3 × 3 depthwise convolution, similar to MobileNet.

these findings, we work in the same direction to adapt the Xception model for the task of semantic image segmentation. In particular, we make a few more changes on top of MSRA's modifications, namely (1) deeper Xception same as in [60] except that we do not modify the entry flow network structure for fast computation and memory efficiency, (2) all max pooling operations are replaced by depthwise separable convolution with striding, which enables us to apply *atrous separable convolution* to extract feature maps at an arbitrary resolution (another option is to extend the atrous algorithm to max pooling operations), and (3) extra batch normalization [32] and ReLU activation are added after each 3×3 depthwise convolution, similar to MobileNet design [31]. The modified Xception structure is shown in Fig. 3.

4. Experimental Evaluation

We employ ResNet-101 [27] or modified aligned Xception [12, 60], which are pretrained on the ImageNet-1k dataset [62], to extract dense feature maps by atrous convolution. Our implementation is built on TensorFlow [1] and is made publicly available.

The proposed models are evaluated on the PASCAL VOC 2012 semantic segmentation benchmark [17] which contains 20 foreground object classes and one background class. The original dataset contains $1,464$ (*train*), $1,449$ (*val*), and $1,456$ (*test*) pixel-level annotated images. We augment the dataset by the extra annotations provided by [24], resulting in $10,582$ (*trainaug*) training images. The performance is measured in terms of pixel intersection-over-union averaged across the 21 classes (mIOU).

We follow the same training protocol as in [10] and refer the interested readers to [10] for details. In short, we employ the same learning rate schedule (*i.e*., "poly" policy [47] and same initial learning rate 0.007), crop size 513×513, fine-tuning batch normalization parameters [32] when *output stride* $= 16$, and random scale data augmentation during training. Note that we also include batch normalization parameters in the proposed decoder module. Our proposed model is trained end-to-end without piecewise pretraining of each component.

4.1. Decoder Design Choices

We first define "DeepLabv3 feature map" as the last feature map computed by DeepLabv3 (*i.e*., the features that contain ASPP features, image-level features and so on), and $[k \times k, f]$ as a convolution operation with kernel size $k \times k$ and f filters.

When employing *output stride* $= 16$, ResNet-101 based DeepLabv3 [10] bilinearly upsamples the logits by 16 during both training and evaluation. This simple bilinear upsampling could be considered as a naive decoder design, attaining the performance of 77.21% [10] on PASCAL VOC 2012 *val* set and is 1.2% better than not using this naive decoder during training (*i.e*., downsampling groundtruth during training). To improve over this naive baseline, our proposed model "DeepLabv3+" adds the decoder module on top of the encoder output, as shown in Fig. 2. In the decoder module, we consider three places for different design choices, namely (1) the 1×1 convolution used to reduce the channels of the low-level feature map from the encoder module, (2) the 3×3 convolution used to obtain sharper segmentation results, and (3) what encoder low-level features should be used.

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

Table 1. 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 (before striding). Performance on VOC 2012 *val* set.

To evaluate the effect of the 1×1 convolution in the decoder module, we employ $[3 \times 3, 256]$ and the Conv2 features from ResNet-101 network backbone, *i.e*., the last feature map in res2x residual block (to be concrete, we use the feature map before striding). As shown in Tab. 1, reducing the channels of the low-level feature map from the encoder module to either 48 or 32 leads to better performance. We thus adopt $[1 \times 1, 48]$ for channel reduction.

We then design the 3×3 convolution structure for the decoder module and report the findings in Tab. 2. We find that after concatenating the Conv2 feature map (before striding) with DeepLabv3 feature map, it is 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. We also experiment with the case where both Conv2 and Conv3 feature maps are exploited in the decoder module. In this case, the decoder feature map are gradually upsampled by 2, concatenated with Conv3 first and then Conv2, and each will be refined by the $[3 \times 3, 256]$ operation. The whole decoding procedure is then similar to the U-Net/SegNet design [61, 3]. However, we have not observed significant improvement. Thus, in the end, we adopt the very simple yet effective decoder module: the concatenation of the DeepLabv3 feature map and the channel-reduced Conv2 feature map are refined by two $[3 \times 3, 256]$ operations. Note that our proposed DeepLabv3+ model has *output stride* $= 4$. We do not pursue further denser output feature map (*i.e*., *output stride* < 4) given the limited GPU resources.

4.2. ResNet-101 as Network Backbone

To compare the model variants in terms of both accuracy and speed, we report mIOU and Multiply-Adds in Tab. 3 when using ResNet-101 [27] as network backbone in the proposed DeepLabv3+ model. Thanks to atrous convolution, we are able to obtain features at different resolutions during training and evaluation using a single model.

Features		3×3 Conv	mIOU
Conv2	Conv3	Structure	
✓		$[3 \times 3, 256]$	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.

Baseline: The first row block in Tab. 3 contains the results from [10] showing that extracting denser feature maps during evaluation (*i.e.*, *eval output stride* $= 8$) and adopting multi-scale inputs increases performance. Besides, adding left-right flipped inputs doubles the computation complexity with only marginal performance improvement.

Adding decoder: The second row block in Tab. 3 contains the results when adopting the proposed decoder structure. The performance is improved from 77.21% to 78.85% or 78.51% to 79.35% when using *eval output stride* $= 16$ or 8, respectively, at the cost of about 20B extra computation overhead. The performance is further improved when using multi-scale and left-right flipped inputs.

Coarser feature maps: We also experiment with the case when using *train output stride* $= 32$ (*i.e.*, no atrous convolution at all during training) for fast computation. As shown in the third row block in Tab. 3, adding the decoder brings about 2% improvement while only 74.20B Multiply-Adds are required. However, the performance is always about 1% to 1.5% below the case in which we employ *train output stride* $= 16$ and different *eval output stride* values. We thus prefer using *output stride* $= 16$ or 8 during training or evaluation depending on the complexity budget.

4.3. Xception as Network Backbone

We further employ the more powerful Xception [12] as network backbone. Following [60], we make a few more changes, as described in Sec. 3.2.

ImageNet pretraining: The proposed Xception network is pretrained on ImageNet-1k dataset [62] with similar training protocol in [12]. Specifically, we adopt Nesterov momentum optimizer with momentum = 0.9, initial learning rate = 0.05, rate decay = 0.94 every 2 epochs, and weight decay $4e-5$. We use asynchronous training with 50 GPUs and each GPU has batch size 32 with image size 299×299. We did not tune the hyper-parameters very hard as the goal is to pretrain the model on ImageNet for semantic segmentation. We report the *single-model* error rates on the validation set in Tab. 4 along with the baseline reproduced ResNet-101 [27] under the same training protocol. We have observed 0.75% and 0.29% performance degradation for Top1 and Top5 accuracy when not adding the extra batch normalization and ReLU after each 3×3 depthwise convolution in the modified Xception.

The results of using the proposed Xception as network backbone for semantic segmentation are reported in Tab. 5.

Baseline: We first report the results without using the proposed decoder in the first row block in Tab. 5, which shows that employing Xception as network backbone improves the performance by about 2% when *train output stride* $=$ *eval output stride* $= 16$ over the case where ResNet-101 is used. Further improvement can also be obtained by using *eval output stride* $= 8$, multi-scale inputs during inference and adding left-right flipped inputs. Note that we do not employ the multi-grid method [75, 15, 10], which we found does not improve the performance.

Adding decoder: As shown in the second row block in Tab. 5, adding decoder brings about 0.8% improvement when using *eval output stride* $= 16$ for all the different inference strategies. The improvement becomes less when using *eval output stride* $= 8$.

Using depthwise separable convolution: Motivated by the efficient computation of depthwise separable convolution, we further adopt it in the ASPP and the decoder modules. As shown in the third row block in Tab. 5, the computation complexity in terms of Multiply-Adds is significantly reduced by 33% to 41%, while similar mIOU performance is obtained.

Pretraining on COCO: For comparison with other state-of-art models, we further pretrain our proposed DeepLabv3+ model on MS-COCO dataset [46], which yields about extra 2% improvement for all different inference strategies.

Pretraining on JFT: Similar to [10], we also employ the proposed Xception model that has been pretrained on both ImageNet-1k [62] and JFT-300M dataset [29, 12, 69], which brings extra 0.8% to 1% improvement.

Test set results: Since the computation complexity is not considered in the benchmark evaluation, we thus opt for the best performance model and train it with *output stride* $= 8$ and frozen batch normalization parameters. In the end, our 'DeepLabv3+' achieves the performance of 87.8% and 89.0% without and with JFT dataset pretraining.

Qualitative results: We provide visual results of our best model in Fig. 6. As shown in the figure, our model is able to segment objects very well without any post-processing.

Failure mode: As shown in the last row of Fig. 6, our model has difficulty in segmenting (a) sofa *vs*. chair, (b) heavily occluded objects, and (c) objects with rare view.

4.4. Improvement along Object Boundaries

In this subsection, we evaluate the segmentation accuracy with the trimap experiment [36, 37, 9] to quantify the accuracy of the proposed decoder module near object boundaries.

Encoder		Decoder	MS	Flip	mIOU	Multiply-Adds
train OS	eval OS					
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 when using *ResNet-101* as feature extractor. **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.

Model	Top-1 Error	Top-5 Error
Reproduced ResNet-101	22.40%	6.02%
Modified Xception	20.19%	5.17%

Table 4. *Single-model* error rates on ImageNet-1K validation set.

Specifically, we apply the morphological dilation on 'void' label annotations on *val* set, which typically occurs around object boundaries. We then compute the mean IOU for those pixels that are within the dilated band (called trimap) of 'void' labels. As shown in Fig. 4, employing the proposed decoder for both ResNet-101 [27] and Xception [12] network backbones improves the performance compared to the naive bilinear upsampling. The improvement is more significant when the dilated band is narrow. We have observed 4.8% and 5.4% mIOU improvement for ResNet-101 and Xception respectively at the smallest trimap width as shown in the figure. We also visualize the effect of employing the proposed decoder in Fig. 5.

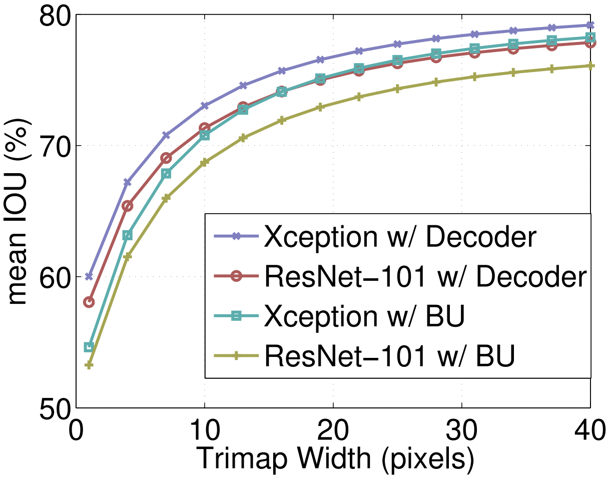


Figure 4. mIOU as a function of the trimap band width around the object boundaries when employing *train output stride* = *eval output stride* = 16. **BU**: Bilinear upsampling.

5. Conclusion

Our proposed model "DeepLabv3+" employs the encoder-decoder structure where DeepLabv3 is used to encode the rich contextual information and a simple yet effective decoder module is adopted to recover the object boundaries. One could also apply the atrous convolution to extract the encoder features at an arbitrary resolution, depending on the available computation resources. We also explore the Xception model and atrous separable convolution to make the proposed model faster and stronger. Finally, our experimental results show that the proposed model sets a new state-of-the-art performance on the PASCAL VOC 2012 semantic image segmentation benchmark.

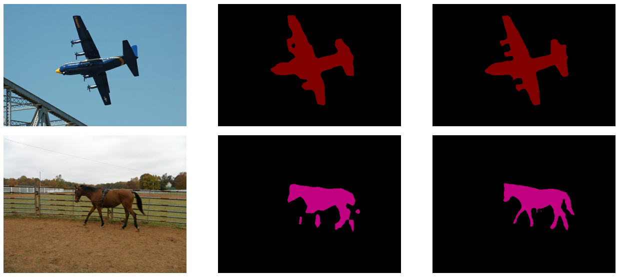
Acknowledgments We would like to acknowledge the valuable discussions with Haozhi Qi and Jifeng Dai about Aligned Xception, the feedback from Chen Sun, and the support from Google Mobile Vision team.

Encoder		Decoder	MS	Flip	SC	COCO	JFT	mIOU	Multiply-Adds
train OS	eval OS								
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* as feature extractor. **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. **SC**: Adopting depthwise separable convolution for both ASPP and decoder modules. **COCO**: Models pretrained on MS-COCO dataset. **JFT**: Models pretrained on JFT dataset.

Method	**mIOU**
Deep Layer Cascade (LC) [42]	82.7
TuSimple [75]	83.1
Large_Kernel_Matters [57]	83.6
Multipath-RefineNet [43]	84.2
ResNet-38_MS_COCO [77]	84.9
PSPNet [81]	85.4
IDW-CNN [73]	86.3
CASIA_IVA_SDN [20]	86.6
DIS [50]	86.8
DeepLabv3 [10]	85.7
DeepLabv3-JFT [10]	86.9
DeepLabv3+ (Xception)	87.8
DeepLabv3+ (Xception-JFT)	89.0

Table 6. PASCAL VOC 2012 *test* set results with top-performing models. We refer interested readers to leaderboard for details.



(a) Image (b) w/ BU (c) w/ Decoder

Figure 5. 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.

References

[1] M. Abadi, A. Agarwal, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv:1603.04467*, 2016.

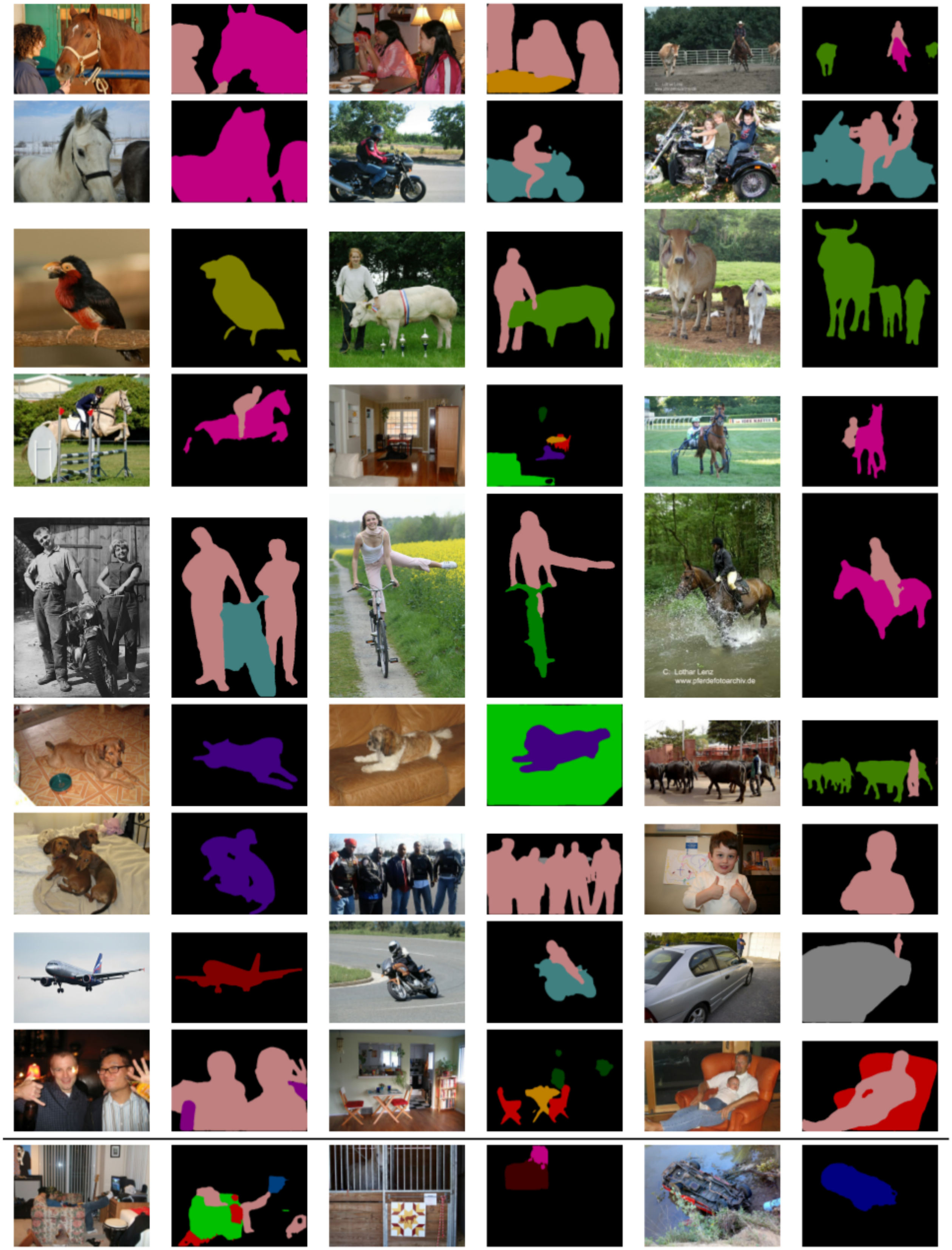


Figure 6. Visualization results on the PASCAL VOC 2012 *val* set when employing our best model. The last row shows a failure mode.

[2] A. Adams, J. Baek, and M. A. Davis. Fast high-dimensional filtering using the permutohedral lattice. In *Eurographics*, 2010.

[3] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv:1511.00561*, 2015.

[4] S. Bell, P. Upchurch, N. Snavely, and K. Bala. Material recognition in the wild with the materials in context database. In *CVPR*, 2015.

[5] H. Caesar, J. Uijlings, and V. Ferrari. COCO-Stuff: Thing and stuff classes in context. *arXiv:1612.03716*, 2016.

[6] S. Chandra and I. Kokkinos. Fast, exact and multi-scale inference for semantic image segmentation with deep Gaussian CRFs. In *ECCV*, 2016.

[7] S. Chandra, N. Usunier, and I. Kokkinos. Dense and low-rank gaussian crfs using deep embeddings. In *ICCV*, 2017.

[8] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015.

[9] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 2017.

[10] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv:1706.05587*, 2017.

[11] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille. Attention to scale: Scale-aware semantic image segmentation. In *CVPR*, 2016.

[12] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017.

[13] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.

[14] J. Dai, K. He, and J. Sun. Convolutional feature masking for joint object and stuff segmentation. In *CVPR*, 2015.

[15] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In *ICCV*, 2017.

[16] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015.

[17] M. Everingham, S. M. A. Eslami, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge a retrospective. *IJCV*, 2014.

[18] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *PAMI*, 2013.

[19] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. Dssd: Deconvolutional single shot detector. *arXiv:1701.06659*, 2017.

[20] J. Fu, J. Liu, Y. Wang, and H. Lu. Stacked deconvolutional network for semantic segmentation. *arXiv:1708.04943*, 2017.

[21] A. Giusti, D. Ciresan, J. Masci, L. Gambardella, and J. Schmidhuber. Fast image scanning with deep max-pooling convolutional neural networks. In *ICIP*, 2013.

[22] S. Gould, R. Fulton, and D. Koller. Decomposing a scene into geometric and semantically consistent regions. In *ICCV*. IEEE, 2009.

[23] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV*, 2005.

[24] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *ICCV*, 2011.

[25] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015.

[26] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014.

[27] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[28] X. He, R. S. Zemel, and M. Carreira-Perpindn. Multiscale conditional random fields for image labeling. In *CVPR*, 2004.

[29] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. In *NIPS*, 2014.

[30] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian. A real-time algorithm for signal analysis with the help of the wavelet transform. In *Wavelets: Time-Frequency Methods and Phase Space*, pages 289–297. 1989.

[31] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv:1704.04861*, 2017.

[32] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[33] M. A. Islam, M. Rochan, N. D. Bruce, and Y. Wang. Gated feedback refinement network for dense image labeling. In *CVPR*, 2017.

[34] V. Jampani, M. Kiefel, and P. V. Gehler. Learning sparse high dimensional filters: Image filtering, dense crfs and bilateral neural networks. In *CVPR*, 2016.

[35] J. Jin, A. Dundar, and E. Culurciello. Flattened convolutional neural networks for feedforward acceleration. *arXiv:1412.5474*, 2014.

[36] P. Kohli, P. H. Torr, et al. Robust higher order potentials for enforcing label consistency. *IJCV*, 82(3):302–324, 2009.

[37] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *NIPS*, 2011.

[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[39] L. Ladicky, C. Russell, P. Kohli, and P. H. Torr. Associative hierarchical crfs for object class image segmentation. In *ICCV*, 2009.

[40] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[41] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proc. IEEE*, 1998.

[42] X. Li, Z. Liu, P. Luo, C. C. Loy, and X. Tang. Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade. *arXiv:1704.01344*, 2017.

[43] G. Lin, A. Milan, C. Shen, and I. Reid. Refinenet: Multi-path refinement networks with identity mappings for high-resolution semantic segmentation. In *CVPR*, 2017.

[44] G. Lin, C. Shen, A. van den Hengel, and I. Reid. Efficient piecewise training of deep structured models for semantic segmentation. In *CVPR*, 2016.

[45] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[46] T.-Y. Lin et al. Microsoft COCO: Common objects in context. In *ECCV*, 2014.

[47] W. Liu, A. Rabinovich, and A. C. Berg. Parsenet: Looking wider to see better. *arXiv:1506.04579*, 2015.

[48] Z. Liu, X. Li, P. Luo, C. C. Loy, and X. Tang. Semantic image segmentation via deep parsing network. In *ICCV*, 2015.

[49] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[50] P. Luo, G. Wang, L. Lin, and X. Wang. Deep dual learning for semantic image segmentation. In *ICCV*, 2017.

[51] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich. Feedforward semantic segmentation with zoom-out features. In *CVPR*, 2015.

[52] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille. The role of context for object detection and semantic segmentation in the wild. In *CVPR*, 2014.

[53] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016.

[54] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *ICCV*, 2015.

[55] G. Papandreou, L.-C. Chen, K. Murphy, and A. L. Yuille. Weakly- and semi-supervised learning of a dcnn for semantic image segmentation. In *ICCV*, 2015.

[56] G. Papandreou, I. Kokkinos, and P.-A. Savalle. Modeling local and global deformations in deep learning: Epitomic convolution, multiple instance learning, and sliding window detection. In *CVPR*, 2015.

[57] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun. Large kernel matters–improve semantic segmentation by global convolutional network. In *CVPR*, 2017.

[58] P. Pinheiro and R. Collobert. Recurrent convolutional neural networks for scene labeling. In *ICML*, 2014.

[59] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe. Full-resolution residual networks for semantic segmentation in street scenes. In *CVPR*, 2017.

[60] H. Qi, Z. Zhang, B. Xiao, H. Hu, B. Cheng, Y. Wei, and J. Dai. Deformable convolutional networks – coco detection and segmentation challenge 2017 entry. *ICCV COCO Challenge Workshop*, 2017.

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

[62] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015.

[63] A. G. Schwing and R. Urtasun. Fully connected deep structured networks. *arXiv:1503.02351*, 2015.

[64] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014.

[65] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *IJCV*, 2009.

[66] A. Shrivastava, R. Sukthankar, J. Malik, and A. Gupta. Beyond skip connections: Top-down modulation for object detection. *arXiv:1612.06851*, 2016.

[67] L. Sifre. Rigid-motion scattering for image classification. *PhD thesis*, 2014.

[68] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[69] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, 2017.

[70] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[71] V. Vanhoucke. Learning visual representations at scale. *ICLR invited talk*, 2014.

[72] R. Vemulapalli, O. Tuzel, M.-Y. Liu, and R. Chellappa. Gaussian conditional random field network for semantic segmentation. In *CVPR*, 2016.

[73] G. Wang, P. Luo, L. Lin, and X. Wang. Learning object interactions and descriptions for semantic image segmentation. In *CVPR*, 2017.

[74] M. Wang, B. Liu, and H. Foroosh. Design of efficient convolutional layers using single intra-channel convolution, topological subdivisioning and spatial ”bottleneck” structure. *arXiv:1608.04337*, 2016.

[75] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. Cottrell. Understanding convolution for semantic segmentation. *arXiv:1702.08502*, 2017.

[76] Z. Wojna, V. Ferrari, S. Guadarrama, N. Silberman, L.-C. Chen, A. Fathi, and J. Uijlings. The devil is in the decoder. In *BMVC*, 2017.

[77] Z. Wu, C. Shen, and A. van den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *arXiv:1611.10080*, 2016.

[78] S. Xie, R. Girshick, P. Dollr, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

[79] J. Yao, S. Fidler, and R. Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *CVPR*, 2012.

[80] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *arXiv:1707.01083*, 2017.

[81] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, 2017.

[82] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr. Conditional random fields as recurrent neural networks. In *ICCV*, 2015.

[83] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene parsing through ade20k dataset. In *CVPR*, 2017.

[84] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. *arXiv:1707.07012*, 2017.