Stacked U-Nets: A No-Frills Approach to Natural Image Segmentation

Sohil Shah, Pallabi Ghosh, Larry S. Davis and Tom Goldstein

University of Maryland, College Park, MD USA.
{sohilas,lsdavis}@umd.edu,{pallabig,tomg}@cs.umd.edu

Abstract. Many imaging tasks require global information about all pixels in an image. Conventional bottom-up classification networks globalize information by decreasing resolution; features are pooled and downsampled into a single output. But for semantic segmentation and object detection tasks, a network must provide higher-resolution pixel-level outputs. To globalize information while preserving resolution, many researchers propose the inclusion of sophisticated auxiliary blocks, but these come at the cost of a considerable increase in network size and computational cost. This paper proposes stacked u-nets (SUNets), which iteratively combine features from different resolution scales while maintaining resolution. SUNets leverage the information globalization power of u-nets in a deeper network architectures that is capable of handling the complexity of natural images. SUNets perform extremely well on semantic segmentation tasks using a small number of parameters. The code is available at https://github.com/shahsohil/sunets.

1 Introduction

Semantic segmentation methods decompose an image into groups of pixels, each representing a common object class. While the output of a segmentation contains object labels assigned at the local (pixel) level, each label much have a global *field of view*; each such label depends on global information about the image, such as textures, colors, and object boundaries that may span large chunks of the image. Simple image classification algorithms consolidate global information by successively pooling features until the final output is a single label containing information from the entire image. In contrast, segmentation methods must output a full-resolution labeled image (rather than a single label). Thus, a successful segmentation method must address this key question: how can we learn long-distance contextual information while at the same time retaining high spatial resolution at the output for identifying small objects and sharp boundaries?

For natural image processing, most research has answered this question using one of two approaches. One approach is to use very few pooling layers, thus maintaining resolution (although methods may still require a small number of deconvolution layers [1,2,3,4,5]). Large fields of view are achieved using dilated convolutions, which span large regions. By maintaining resolution at each layer, this approach preserves substantial amounts of signal about smaller and less

salient objects. However, this is achieved at the cost of computationally expensive and memory exhaustive training/inference. The second related approach is to produce auxiliary context aggregation blocks [6,7,8,9,10,11,12,13,14,15] that contain features at different distance scales, and then merge these blocks to produce a final segmentation. This category includes many well-known techniques such as dense CRF [6] (conditional random fields) and spatial pyramid pooling [7].

These approaches suffer from the following challenges:

1. Deconvolutional (i.e., encoder-decoder) architectures perform significant non-linear computation at low resolutions, but do very little processing of high-resolution features. During the convolution/encoding stage, pooling layers can move information over large distances, but information about small objects is often lost. During the deconvolution/decoding stage, high- and low-resolution features are merged to produce upsampled feature maps. However, the high-resolution inputs to deconvolutional layers come from relatively shallow layers that do not effectively encode semantic information.
2. Image classification networks are parameter heavy (44.5M parameters for ResNet-101), and segmentation methods built on top of these classification networks are often even more burdensome. For example, on top of the resnet-101 architecture, PSPNet [14] uses 22M *additional* parameters for context aggregation, while the ASPP and Cascade versions of the Deeplab network utilize 14.5M [7] and 40M [15] *additional* parameters, respectively.

A popular and simple approach to segmentation is u-nets, which perform a chain of convolutional/downsampling operations, followed by a chain of deconvolutional/upsampling layers that see information from both low- and high-resolution scales. These u-net architectures are state-of-the art for medical image segmentation [16], but they do not perform well when confronted with the complex color profiles, lighting effects, perspectives, and occlusions present in natural images.

We expand the power of u-nets by stacking u-net blocks into deep architectures. This addresses the two challenges discussed above: As data passes through multiple u-net blocks, high-resolution features are mixed with low-resolution context information and processed through many layers to produce informative high-resolution features. Furthermore, stacked U-net models require fewer feature maps per layer than conventional architectures, and thus achieve higher performance with far fewer parameters. Our smallest model exceeds the performance of ResNet-101 on the PASCAL VOC 2012 semantic segmentation task by 4.5% mIoU, while having $\sim 7\times$ fewer parameters.

2 Related Work

Many models [15,14,17,7,18,5,1,3] have boosted the performance of semantic segmentation networks. These gains are mainly attributed to the use of pre-trained models, dilated convolutional layers [7,13] and fully convolutional architectures

(DCNN) [19]. These works employ a range of strategies to tap contextual information, which fall into three major categories.

Context Aggregation Modules: These architectures place a special module on top of a pre-trained network that integrates context information at different distance scales. The development of fast and efficient algorithm for DenseCRF [6] led to the development of numerous algorithms [7,8,9,10] incorporating it on top of the output belief map. Moreover, the joint training of CRF and CNN parameters was made possible by [11,12]. In [13], context information was integrated by processing a belief map using a cascade of dilated layers operating at progressively increasing dilation rates, and [18] proposed a hybrid dilation convolution framework to alleviate gridding artifacts. ParseNet [20] exploits image-level feature information at each layer to learn global contextual information. In contrast, [14,15,7] realized substantial performance improvements by employing parallel layers of spatial pyramid pooling. The work [14] spatially pools output feature maps at different scales, while [15,7] advocates applying dilated convolution at varying dilation rates.

Image Pyramid: The networks proposed in [21,22] learn context information by simultaneously processing inputs at different scales and merging the output from all scales. An attention mechanism was used to perform fusion of output maps in [22], while [21] concatenates all the feature maps produced by blocks of parallel layers, each learned exclusively for differently scaled inputs. Recently, [23] developed a deformable network that adaptively determines an object's scale and accordingly adjusts the receptive field size of each activation function. On the other hand, [24] proposed a new training paradigm for object detection networks that trains each object instance only using the proposals closest to the ground truth scale.

Encoder-Decoder: These models consist of an encoder network and one or many blocks of decoder layers. The decoder fine-tunes the pixel-level labels by merging the contextual information from feature maps learned at all the intermediate layers. Usually, a popular bottom-up pre-trained classification network such as ResNet [25], VGG [26] or DenseNet [27] serves as an encoder model. U-net [16] popularly employs skip connections between an encoder and its corresponding decoding layers. On a similar note, the decoder in Segnet [28] upsamples the lower resolution maps by reusing the pooling indices from the encoder. Deconvolution layers were stacked in [2,3,4], whereas [5] uses a Laplacian pyramid reconstruction network to selectively refine the low resolution maps. Refinenet [1] employs sophisticated decoder modules at each scale on top of the ResNet encoder, while [29] utilizes a simple two-level decoding of feature maps from the Xception network [30]. In short, the structure in [29,5] is a hybrid of decoding and context aggregation modules. Any task that requires extracting multi-scale information from inputs can benefit from an encoder-decoder structure. The recent works on object detection [31,32] also utilize this structure.

2.1 Use of Pre-Trained Nets

Many of the networks described above make extensive use of image classification networks that were pre-trained for other purposes. ResNet employs an identity mapping [33] which, along with batch-normalization layers, facilitates efficient learning of very deep models. VGG was popular before the advent of ResNet. Although parameter heavy, much fundamental work on segmentation (FCN [19], dilated nets [13], u-nets [16] and CRF [11]) was built on VGG. All these architectures share common origins in that they were designed for the ImageNet competition and features are processed bottom-up. This prototype works well when the network has to identify only a single object without exact pixel localization. However, when extended to localization tasks such as segmentation and object detection, it is not clear whether the complete potential of these networks has been properly tapped. Recent work on object detection [24] also echoes a similar concern.

3 U-Nets Revisited

The original u-net architecture was introduced in [16], and produced almost perfect segmentation of cells in biomedical images using very little training data. The structure of u-nets makes it possible to capture context information at multiple scales and propagate them to the higher resolution layers. These higher order features have enabled u-nets to outperform previous deep models [19] on various tasks including semantic segmentation [34,35,4], depth-fusion [36], image translation [37] and human-pose estimation [38]. Moreover, driven by the initial success of u-nets, many recent works on semantic segmentation [1,5,28,2,3,4] and object detection [31,32] also propose an encoder-decoder deep architecture.

The u-net architecture evenly distributes its capacity among the encoder and decoder modules. Moreover, the complete network can be trained in an end-to-end setting. In contrast, the more recent architectures reviewed in Section 2 do not equally distribute the processing of top-down and bottom-up features. Since these architectures are built on top of pre-trained feature extractors [26,25,30], the decoder modules are trained separately and sometimes in multiple stages. To overcome these drawbacks, [4] proposed an equivalent u-net based on the Densenet [27] architecture. However, DenseNet is memory intensive, and adding additional decoder layers leads to a further increase in memory usage. Given these drawbacks, the effectiveness of these architectures on different datasets and applications is unclear.

The goal of this paper is to realize the benefits of u-nets (small size, easy trainability, high performance) for complex natural image segmentation problems. Specifically, we propose a new architecture composed of multiple stacks of u-nets. The network executes repeated processing of both top-down as well as bottom-up features and captures long-distance spatial information at multiple resolutions. The network is trained end-to-end on image classification tasks and can be seamlessly applied to semantic segmentation without any additional modules on top (except for replacing the final classifier).

Our stacked u-net (SUNet) architecture shares some similarity with other related stacked encoder-decoder structures [3,38]. Fu *et al.* [3] uses multiple stacks of de-convolutional networks (maximum of three) on top of a powerful encoder (DenseNet) while [38] applies multiple stacks of u-net modules for human-pose estimation. However, the processing of features inside each u-net module in [38] differs from ours. [38] replaces each convolutional block with a residual module and utilizes nearest-neighbor upsampling for deconvolution. In contrast, SUNets retain the basic u-net structure from [16]. Also, SUNet operates without any intermediate supervision and processes features by progressively downsampling while [38] operates at fix resolution.

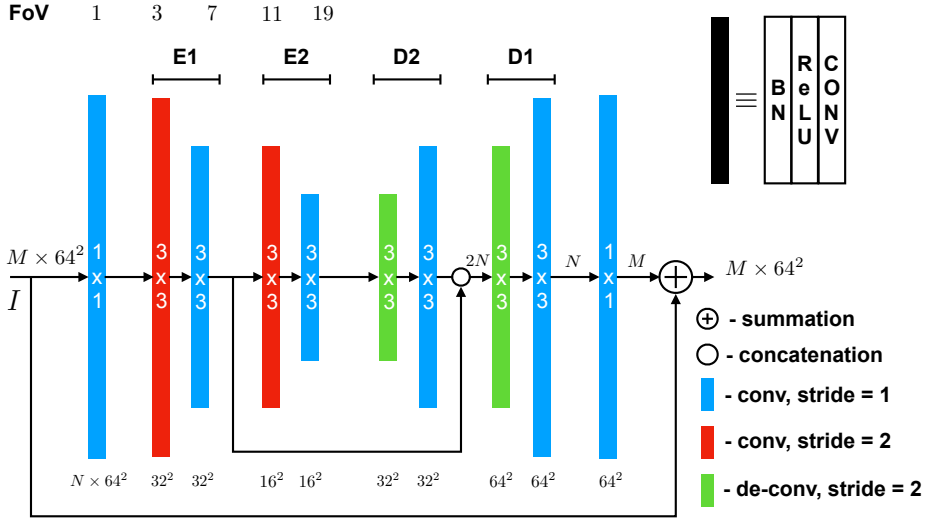

Fig. 1. A typical u-net module with outer residual connection. M is the number of input features. Across the u-net module, each layer has the same number of output feature maps (except for the final 1×1 filter), which we denote N. For better understanding the figure also includes the field of view (FoV) of each convolutional kernel (top) and the feature map size at the output of each filter (bottom), assuming a 64×64 input I. Best viewed in color.

3.1 U-Net Module Implementation

Figure 1 illustrates the design of the u-net module employed in our stacked architecture. Each module is composed of 10 pre-activated convolutional blocks each preceded by a batch-normalization and a ReLU non-linearity. The pooling/unpooling operation, handled by the strided convolutional/deconvolutional layers, facilitates information exchange between the lower and the higher resolution features. A skip connection branches off at the output of the first encoder block, $E1$. Following this, the $E2$ and $D2$ blocks capture long distance context information using lower resolution feature maps and merge the information back with the high resolution features from $E1$ at the output of $D2$. Every layer (except for the bottleneck layers) uses 3×3 kernels, and outputs a fixed number of

feature maps, N. To mitigate high frequency noise from the sampling operation, each strided conv/de-conv layer is followed by a convolution. Unlike traditional u-nets, the design of convolutional layers in our u-net module helps in retaining the original size of the feature maps at its output (no crop operation takes place). Consequently, multiple u-net modules can be stacked without loosing resolution.

In the following, we briefly highlight some of the design choices of the architecture. In comparison to traditional u-nets, the max-pooling operation is replaced with strided convolution for SUNets. The use of strided convolutions enables different filters in each u-net module to operate at different resolutions (see the discussion in Section 5). Moreover, the repeated use of max-pooling operations can cause gridding artifacts in dilated networks [39].

Unlike the u-nets of [16,38], our u-net module is comprised of only two levels of depth. We considered two major factors in choosing the depth: field of view (FoV) of the innermost conv filter and the total number of parameters in a single u-net module. The number of parameters influences the total number of stacks in SUNets. While keeping the total parameters of SUNet approximately constant, we experimented with a higher depth of three and four. We found that the increase in depth indeed led to a decline in performance for the image classification task. This may not be surprising, given that a SUNet with depth of two is able to stack more u-net modules. Moreover, deeper u-net modules make it harder to train the inner-most convolutional layers due to the vanishing gradients problem [40]. For instance, in our current design, the maximum length of the gradient path is six. The popular classification networks [25,27] are known to operate primarily on features with 28^2 and 14^2 resolution. At this scale, the effective FoV of 19 is more than sufficient to capture long-distance contextual information. Moreover, the stacking of multiple u-net modules will also serve to increase the effective FoV of higher layers.

SUNets train best when there is sufficient gradient flow to the bottom-most u-net layers. To avoid vanishing gradients, we include a skip connection [25,27] around each u-net module. Also, inspired by the design of bottleneck blocks [25], we also include 1×1 convolutional layers. Bottleneck layers restricts the number of input features to a small number (N), avoiding parameter inflation.

When stacking multiple u-nets it makes sense for each u-net module to reuse the raw feature maps from all the preceding u-net modules. Thus we also explored replacing the identity connection with dense connectivity [27]. This new network is memory intensive[1] which in turn prevented proper learning of the batch-norm parameters. Instead, we chose to utilize dense connectivity only within each u-net, i.e., while reusing feature maps from $E1$ at $D1$. Thus the proposed u-net module leverages skip connectivity without getting burdened.

4 SUNets: Stacked U-Nets for Classification

Before addressing segmentation, we describe a stacked u-net (SUNet) architecture that is appropriate for image classification. Because the amount of labeled

[1] Restricted by the current implementation of deep learning packages

Layers	Output Size	SUNet-64	SUNet-128	SUNet-7-128
Convolution	112×112	7×7 conv, 64, stride 2		
Residual Block	56×56	$\begin{bmatrix} 3 \times 3 \text{ conv, } 128, \text{ stride } 2 \\ 3 \times 3 \text{ conv, } 128, \text{ stride } 1 \end{bmatrix} \times 1$		
UNets Block (1)	56×56	$\begin{bmatrix} 1 \times 1 \text{ conv, } 64 \\ \text{U-Net, N=64} \\ 1 \times 1 \text{ conv, } 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1 \text{ conv, } 128 \\ \text{U-Net, N=128} \\ 1 \times 1 \text{ conv, } 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1 \text{ conv, } 128 \\ \text{U-Net, N=128} \\ 1 \times 1 \text{ conv, } 512 \end{bmatrix} \times 2$
Transition Layer	28×28	2×2 average pool, stride 2		
UNets Block (2)	28×28	$\begin{bmatrix} 1 \times 1 \text{ conv, } 64 \\ \text{U-Net, N=64} \\ 1 \times 1 \text{ conv, } 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1 \text{ conv, } 128 \\ \text{U-Net, N=128} \\ 1 \times 1 \text{ conv, } 1024 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1 \text{ conv, } 128 \\ \text{U-Net, N=128} \\ 1 \times 1 \text{ conv, } 1280 \end{bmatrix} \times 7$
Transition Layer	14×14	2×2 average pool, stride 2		
UNets Block (3)	14×14	$\begin{bmatrix} 1 \times 1 \text{ conv, } 64 \\ \text{U-Net, N=64} \\ 1 \times 1 \text{ conv, } 768 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1 \text{ conv, } 128 \\ \text{U-Net, N=128} \\ 1 \times 1 \text{ conv, } 1536 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1 \text{ conv, } 128 \\ \text{U-Net, N=128} \\ 1 \times 1 \text{ conv, } 2048 \end{bmatrix} \times 7$
Transition Layer	7×7	2×2 average pool, stride 2		
UNets Block (4)	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv, } 64 \\ \text{U-Net}^+\text{, N=64} \\ 1 \times 1 \text{ conv, } 1024 \end{bmatrix} \times 1$	$\begin{bmatrix} 1 \times 1 \text{ conv, } 128 \\ \text{U-Net}^+\text{, N=128} \\ 1 \times 1 \text{ conv, } 2048 \end{bmatrix} \times 1$	$\begin{bmatrix} 1 \times 1 \text{ conv, } 128 \\ \text{U-Net}^+\text{, N=128} \\ 1 \times 1 \text{ conv, } 2304 \end{bmatrix} \times 1$
Classification Layer	1×1	7×7 global average pool		
		1000D fully-connected, softmax		
Total Layers		110	110	170
Params		$6.9M$	$24.6M$	$37.7M$

Table 1. SUNet architectures for ImageNet. N denotes the number of filters per convolutional layer. Note that the building block in bracket refers to the integrated u-net module shown in Figure 1.

data available for classification is much larger than for segmentation, classification tasks are often used to pre-train feature extraction networks, which are then adapted to perform segmentation.

The network design of SUNets for ImageNet classification is summarized in Table 1. Note that each "conv" layer shown in the table corresponds to a sequence of "BN-ReLU-Conv" layers. The three listed configurations mainly differ in the number of output feature maps N of each convolutional layer and the total number of stacks in blocks 2 and 3. Input images are processed using a 7×7 conv filter followed by a residual block. Inspired by the work on dilated resnet [39], the conventional max-pooling layer at this stage is replaced by a strided convolutional layer inside the residual block. Subsequently, the feature maps are processed bottom-up as well as top-down by multiple stacks of u-nets at different scales and with regularly decreasing resolutions. The feature map input size to block 4 is 7×7 and is further reduced to 2×2 at the input to the encoder $E2$ of the u-net module. At this resolution, it is not possible to have $E2$ and $D2$ layers, and hence a trimmed version of u-nets (u-net$^+$) are employed in block 4. The u-net$^+$ includes a single level of encoder and decoder $(E1, D1)$ processing. Towards the end of block 4, a batch normalization is performed and a ReLU non-linearity is applied. Following this, a global average pooling is performed on features and transferred to a softmax classifier.

The residual connection in all but the first u-net in each block is implemented as an identity mapping. In the first u-nets the skip connection is implemented using an expansion layer i.e., a 1×1 conv filter. The number of feature map outputs from each block approximately equates to the total number of feature maps generated by all the preceding u-net modules. This arrangement allows flexibility for the network to retain all the raw feature maps of the preceding modules. Moreover among all other possibilities, the above architectures were picked because their performance on the image classification task is roughly equivalent to the ResNet-18, 50 and 101 network architectures (discussed in Section 5.3), albeit with fewer parameters. However, in contrast to the work on residual net [41], our experimentation with wider nets (i.e., $N > 128$) did not yield any performance improvements on ImageNet.

As in ResNet [25] and DenseNet [27], most of the processing in SUNet is performed at the feature scale of 14×14 (46 conv layers) and 7×7 (44 conv layers). However, the order at which the local information is processed can lead to a substantial gap in performance between ResNet and SUNet when extending these popular architectures to object localization, detection, and image segmentation tasks. All these task demands pixel-level localization and hence require a deep architecture that can efficiently integrate local and global cues. The development of SUNet is a first step towards achieving this objective. Intuitively, multiple stacks of u-nets can be seen as multiple iterations of the message passing operation in a CRF [11].

5 Dilated SUNets for Segmentation

We now explain how pre-trained SUNet models can be adapted to perform semantic segmentation (see Section 5.3). One can directly extend SUNet to segmentation by removing a global average pooling layer (to increase output resolution) and operating the network in fully convolutional mode. Akin to other works on semantic segmentation [14,1,15], the output feature maps are rescaled using

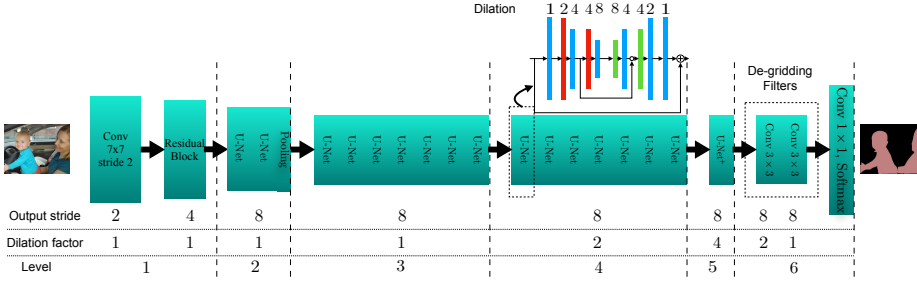

Fig. 2. Dilated SUNet-7-128 network for segmentation at *output_stride* = 8. For dilation > 1, the feature maps are processed with a varying range of dilation factors inside each u-net module (for example, see inset). The de-gridding filters smooth out aliasing artifacts that occur during deconvolution.

bilinear interpolation to the input image size before passing into the softmax layer with multi-class cross-entropy loss.

5.1 Dilation

For an input image of 512×512, the output map size at the softmax is 16×16 i.e., subsampled by a factor of 32. This is insufficient to preserve precise pixel-level localization information at its output. The precision can be improved by increasing the output map size of the network. This is realized by dropping the pooling stride at the transition layer. Merely eliminating stride leads to the reduction in the receptive field of the subsequent layers by a factor of two. Consequently, this reduces the influence of long-distance context information on the output prediction. Nevertheless, the receptive field is restored to that of the original network by operating each convolutional filter in the subsequent layers at a dilation factor of 2 [7,13].

5.2 Multigrid

Figure 2 shows a sample dilated SUNet architecture used for the semantic segmentation task. Similar to [15], we define *output_stride* to be the ratio of resolution of an input image to that of its output feature map.

To sample at an *output_stride* of 8 the pooling layers preceding blocks (3) and (4) are discarded. Following this, the dilation factor for each u-net module in blocks 3 and 4 is fixed at 2 and 4, respectively. In each subsequent u-net module the 3×3 conv layers are operated with $stride = 1$. To keep the receptive field of the low-resolution layers in these modules constant, a dilation is applied. This arrangement facilitates the network to preserve spatial information learned from the prior modules (because there is no downsampling in the final u-net block) while preserving the distance scale of features within each block. As an example, the inset in Figure 2 displays the effective dilation rate for each layer in the u-net module at block 3. Similarly, the dilation rate of each layer (except for bottleneck layers) in the u-net$^+$ module will be twice that of the corresponding layers in block 3. The steady increase and decrease of dilation factors inside each u-net module is analogous to multigrid solvers for linear systems [42,43], which use grids at different scales to move information globally. Many recent works [23,18,15] on deep networks advocate the use of special structures for information globalization. In SUNet, the multigrid structure is baked into the model, and no further "frills" are needed to globalize information.

5.3 De-gridding Filters

By adopting dilated SUNets, we observe a vast improvement in segmentation performance. However, for *output_stride* $= 8$ the segmentation map displays gridding artifacts [18,39]. This aliasing artifact is introduced when the sampling rate of the dilated layer is lower than the high-frequency content of input feature

maps. The final 3×3 conv filter of u-net$^+$ operates at the dilation factor of 4. Directly transferring u-net$^+$'s feature map output to a classification layer can cause gridding artifacts. Following [39], the u-net$^+$ module is followed by two layers of de-gridding filters with progressively decreasing dilation factor. Each filter is a 3×3 conv layer and outputs 512 feature maps.

SUNet does not require any additional post-hoc structural changes popularized by recent works such as decoding layers [1,3], appending context aggregation blocks [15,14,13,7] and learning conditional random fields [11,10]. Hence we regard SUNet as a "no-frills" network.

6 Experiments

6.1 ImageNet Classification

In this section, we evaluate three SUNet architectures on the ILSVRC-2012 classification dataset, which contains $1.28M$ training images and $50,000$ images for validation, with labels distributed over 1000 classes. Training utilized the same data augmentation scheme used for ResNet [25] and DenseNet [27]. Following common practice [25,33], we apply a 224×224 center crop on test images and report Top-1 and Top-5 error on the validation set.

Implementation Details: All the models were implemented using the PyTorch deep learning framework and trained using four P6000 GPUs on a single node. We use SGD with a batch size of 256. For our largest model, 7-128, we were limited to a batch size of 212, due to the GPUs memory constraints. The initial learning rate was set to 0.01 and decreased by a factor of 10 every 30 epochs. We use a weight decay of $5e^{-4}$ and Nesterov momentum of 0.9 without dampening. The weights were initialized as in [44] and all the models were trained from scratch for a total of 100 epochs.

Table 2 compares the performance of SUNet against other classification networks. The comparison is restricted to only ResNet and DenseNet models as most recent work on segmentation builds on top of them. The notable point about the result is that the repeated top-down and the bottom-up processing of features performs equivalently to state-of-the-art classification networks.

We emphasize that our objective is not to surpass classification accuracy but instead to build a better architecture for segmentation by pre-training on a classification task. Indeed, each SUNet model was selected such that it is the counterpart for the corresponding ResNet model.

7 Semantic Segmentation

Semantic segmentation networks were built using the dilated version of the ImageNet pre-trained SUNet models (Section 5). We evaluate on the PASCAL VOC 2012 semantic segmentation benchmark [45] and urban scene understanding Cityscape [46] datasets. The performance on each of these datasets is reported using intersection-over-union (IoU) averaged over all classes.

Model	Top-1	Top-5	Depth	Params
ResNet-18†	30.24	10.92	18	11.7M
ResNet-50†	23.85	7.13	50	25.6M
ResNet-101†	22.63	6.44	101	44.5M
DenseNet-201†	22.80	6.43	201	20M
DenseNet-161†	22.35	6.20	161	28.5M
SUNet-64	29.28	10.21	111	6.9M
SUNet-128	23.64	7.42	111	24.6M
SUNet-7-128	22.47	6.85	171	37.7M

Table 2. Error rates for classification networks on the ImageNet 2012 validation set. '†' denotes error rates from the official PyTorch implementation.

Model	mIoU
ResNet-101 [15]	68.39
SUNet-64	72.85
SUNet-128	77.16
SUNet-7-128	78.95

Table 3. The semantic segmentation performance of dilated SUNet and ResNet-101 networks on PASCAL VOC 2012 validation set trained with *output_stride* = 16. Relative to the ResNet-101 network, all SUNets perform very well.

7.1 Datasets

PASCAL VOC 2012: This dataset contains 1,464 train, 1,449 validation and 1,456 test images. The pixel-level annotation for 20 objects and one background class is made available for the train and validation set. Following common practice, the train set is augmented with additional annotated data from [47] which finally provides a total of 10,582 (trainaug) training images.

Cityscape: This dataset consists of finely annotated images of urban scenes covering multiple instances of cars, roads, pedestrians, buildings, etc. In total, it contains 19 classes on $2,975$ finely annotated training and 500 validation images.

7.2 Implementation Details

We use the SGD optimizer with a momentum of 0.95 and weight decay of 10^{-4}. Each model is fine-tuned starting with an initial learning rate of 0.0002 which is decreased every iteration by a factor of $0.5 \times \left(1 + \cos\left(\pi \frac{\text{iter}}{\text{max iters}}\right)\right)$ [48]. The batch-norm parameters are learned with a decay rate of 0.99 and the input crop size for each training image is set to 512×512. We train each model using two P6000 GPUs and the batch size 22. On PASCAL VOC, each model is trained for 45K iterations while for Cityscapes we use 90K iterations.

Unless mentioned, for all our experiments we set *output_stride* = 16. This means only the u-net modules in the final block (4) operate at dilation factor of two; all other modules use the same stride as in the original classification model. Furthermore, *output_stride* = 16 enables larger batch sizes than smaller stride choices, and hence leads to efficient learning of batch norm parameters. Also, training and inference are 2× faster with *output_stride* = 16 rather than 8.

Data Augmentation: To prevent overfitting during the training process, each training image is resized with a random scale from 0.5 to 2 following which the input image is randomly cropped. Additionally, the input is randomly flipped horizontally and also randomly rotated between $-10°$ to $10°$.

OS	Strided conv	Multigrid
8	65.99	78.64
16	78.25	78.95

Table 4. Performance comparison of multigrid dilation against strided convolution inside each u-net module, using the SUNet-7-128 model and evaluated using mean IoU. **OS** denotes *output_stride* during training.

train OS	eval OS	DL	MS	Flip	mIoU
32	32				76.03
32	32		✓		77.58
32	32		✓	✓	77.57
16	16				78.95
16	16	✓			78.10
16	16		✓		80.22
16	16		✓	✓	80.40
8	8				78.64
8	8	✓			78.88
8	8		✓		80.37
8	8		✓	✓	80.50

Table 5. Performance comparison at various *output_stride* and inference strategies. **MS:** Multi-scale, **DL:** with Degridding Layers

7.3 Ablation Study

We experiment with different SUNet variants on the PASCAL VOC 2012 dataset. **SUNets vs ResNet-101:** We compare the performance of the dilated SUNet architecture on semantic segmentation against the popular dilated ResNet-101 model. Models were fine-tuned on the "trainaug" set without the degridding layers and evaluated on the validation set.

The performance of the plain dilated SUNets surpasses that of ResNet-101 by a wide margin when trained with *output_stride* $= 16$ (Table 3). In fact, the smallest SUNet model, SUNet-64 with $6.7M$ parameters, beats ResNet-101 (with $44.5M$) by an absolute margin of 4.5% IoU while SUNet-7-128, the counterpart network to ResNet-101, improves by over 10.5% IoU. This is substantial, given that the gap between the ResNet-101 and VGG-16 models is $\sim 3\%$ [7] (at *output_stride* $= 8$). This contrasts with the negligible performance differences observed on classification, and suggests that specialized segmentation network architectures can surpass architectures adapted from classification.

Finally, we note that, although SUNets were designed for pixel-level localization tasks, the selected models were chosen only based on their classification performance. By linking the model selection process to the primary task (segmentation and object detection) there is a possibility of improving performance. **Multigrid vs Downsampling:** We compare the performance of multigrid dilation (as shown in Figure 2) inside each u-net against the usual downsampling (Figure 1). We consider a dilated SUNet-7-128 network and report performance at different training *output_stride*. For *output_stride* $= 8$, the network was trained with a batch size of 12. The result is summarized in Table 4. For a dilated network, replacing strided convolutional layers with the corresponding dilated layers is more logical as well as beneficial. This is because, when operating dilated convolutional layers with *stride* > 1, alternate feature points are dropped without being processed by any of the filters, leading to high frequency noise at

the decoder output. Furthermore, due to a skip connection, the features from the lower layers are also corrupted. Due to error propagation, this effect is more prominent in a network with many dilated modules (for eg., $output\_stride = 8$).
Output Stride and Inference Strategy: Finally, we experiment with three different training output strides (8,16,32) and multi-scale inference at test time. For $output\_stride = 32$, none of the layers are dilated and hence de-gridding layers were not used. Training with an $output\_stride = 32$ is equivalent to fine-tuning a classification network with a global pooling layer removed. For multi-scale inference, each input image is scaled and tested using multiple scales $\{0.5, 0.75, 1.0, 1.25\}$ and its left-right flipped image. The average over all output maps is used in the final prediction. See results in Table 5. We note that:

1. The network trained with $OS = 32$ performs 0.7 IoU better (with single scale) than the Resnet-101 and Resnet-152 models [49] each trained at $OS = 8$. This is significant, since the SUNet output contains $16\times$ fewer pixels. This leads to $4\times$ faster training/inference without a performance drop.
2. The degridding layers do not improve performance at $OS = 16$. In this case, there is only a small change in dilation factor between the final layer of SUNet and the classification layer, so aliasing is not problematic.
3. The margin of performance improvement decreases with increase in training OS. Given this and the above fact, subsequently we only report performance for models trained at $OS = 16$ without any degridding layers.

Pretraining with MS-COCO: Following common practice [14,15,29], we pretrain SUNet-7-128 with the MS-COCO dataset [50]. The MS-COCO dataset contains pixel-level annotation for 80 object classes. Except for the PASCAL VOC classes, the pixel annotation for all other classes is set to the background class. Following this, we use all the images from the MS-COCO "trainval" set except for those having < 1000 annotated pixel labels. This yields $90,000$ training images. After pretraining, the model is fine-tuned on "trainaug" for 5K iterations with $10\times$ smaller initial learning rate. In the end, our model achieves 83.27% mIoU on the validation set. This performance is slightly better than the ResNet+ASPP model [15] (82.70%) and equivalent to Xception+ASPP+Decoder model [29] (83.34%).

7.4 Results on Test set

PASCAL VOC 2012: Before submitting test set output to an evaluation server, the above model was further fine-tuned on the "trainval" set with batch-norm parameters frozen and at $10\times$ smaller initial learning rate. Table 6 compares the test set results against other state-of-the-art methods. PSPNet performs slightly better than SUNet, but at the cost of $30M$ more parameters while training at an $output\_stride = 8$. Figure 3 and 4 displays some qualitative results on validation and test sets.
Cityscapes: A similar training strategy as in PASCAL is adopted except that the multi-scale inference is performed on additional scales $\{1.5, 1.75, 2.0, 2.25, 2.5\}$.

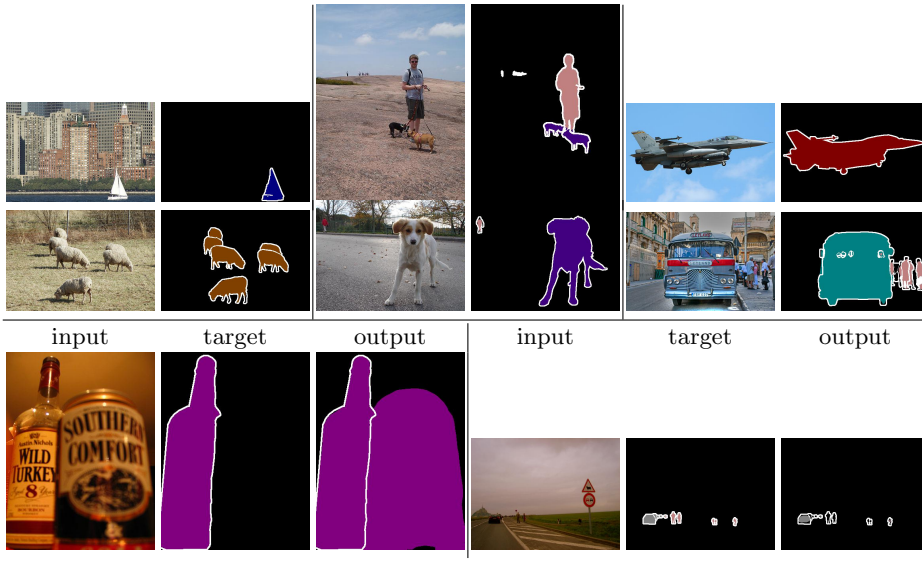

Fig. 3. Visualization of the segmentation output on PASCAL VOC 2012 *val* set when trained at an *output_stride* = 16 using SUNet-7-128 network + MS-COCO. Final row shows couple of failure case which happens due to, ambiguous annotation and inability in detecting low resolution objects.

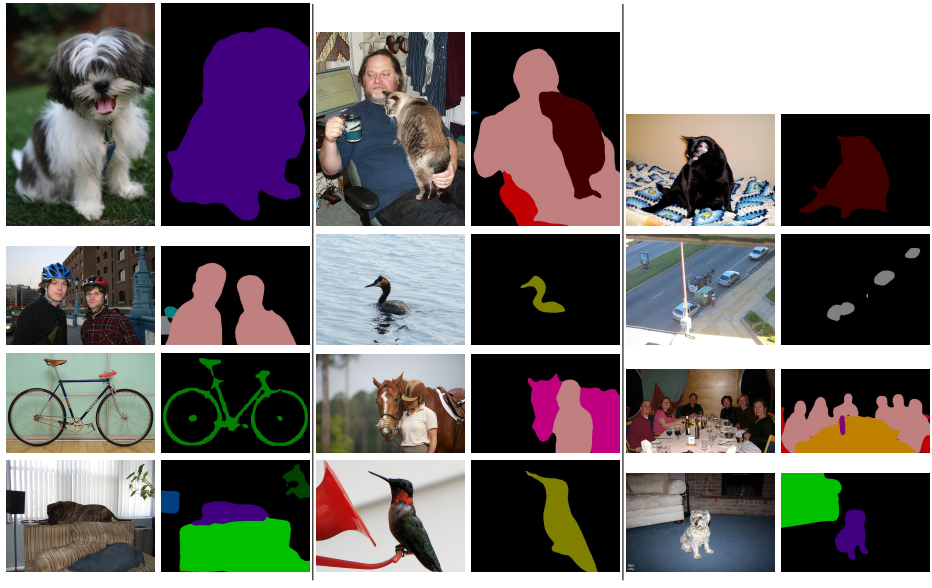

Fig. 4. Visualization of the segmentation output on PASCAL VOC 2012 *test* set.

Only the *finer* annotation set was used for training. The comparison on the Cityscapes test set results are displayed in Table 7.

Methods	mIoU
Piecewise (VGG16) [21]	78.0
LRR+CRF [5]	77.3
DeepLabv2+CRF [7]	79.7
Large-Kernel+CRF [17]	82.2
Deep Layer Cascade* [51]	82.7
Understanding Conv [18]	83.1
RefineNet [1]	82.4
RefineNet-ResNet152 [1]	83.4
PSPNet [14]	**85.4**
SUNet-7-128	84.3[2]

Table 6. Performance comparison on PASCAL VOC 2012 test set. For fair comparison, only the methods pre-trained using MS-COCO are displayed.

Methods	mIoU
LRR (VGG16) [5]	69.7
DeepLabv2+CRF [7]	70.4
Deep Layer Cascade* [51]	71.1
Piecewise (VGG16) [21]	71.6
RefineNet [1]	73.6
Understanding Conv [18]	77.6
PSPNet [14]	**78.4**
SUNet-7-128	75.3[3]

Table 7. Performance comparison on Cityscapes *test* set. All methods were trained only using the "fine" set. All nets utilize ResNet-101 as a base network, except if specified or marked with *.

7.5 Activation Maps

Figure 5 shows the activation map recorded at the end of each level (as indicated in figure 2) for an example input image of an "Aeroplane." As noted earlier, the inclusion of strided convolutions instead of multigrid dilations leads to noisy feature maps (see col 3; rows 4-6). The addition of de-gridding layers serves to produce a coherent prediction map at the output (see col 2; row 6).

8 Discussion and Conclusion

The fundamental structure of conventional bottom-up classification networks limits their efficacy on secondary tasks involving pixel-level localization or classification. To overcome this drawback, a new network architecture, stacked u-nets (SUNets), is discussed in this paper. SUNets leverage the information globalization power of u-nets in a deeper network architecture that is capable of handling the complexity of natural images. SUNets perform exceptionally well on semantic segmentation tasks while achieving fair performance on ImageNet classification.

There are several directions for future research that may improve upon the performance achievable using a simple SUNet. It may be advantageous to replace each convolution block by their corresponding depthwise separable convolution [30], as done in [52,29,53,54]. The inclusion of post-hoc context [14,15] or decoder networks [29] on top of SUNets may also help. Given the huge margin of improvement over ResNet models for semantic segmentation tasks, it is obvious to extend SUNets to object detection tasks. Finally, as suggested in the paper, rigorous hyper-parameter search and numerical analysis [55] on SUNets may yield better generic as well as task-specific models.

[3] http://host.robots.ox.ac.uk:8080/anonymous/KT7JGJ.html
[3] https://www.cityscapes-dataset.com/method-details/?submissionID=1151

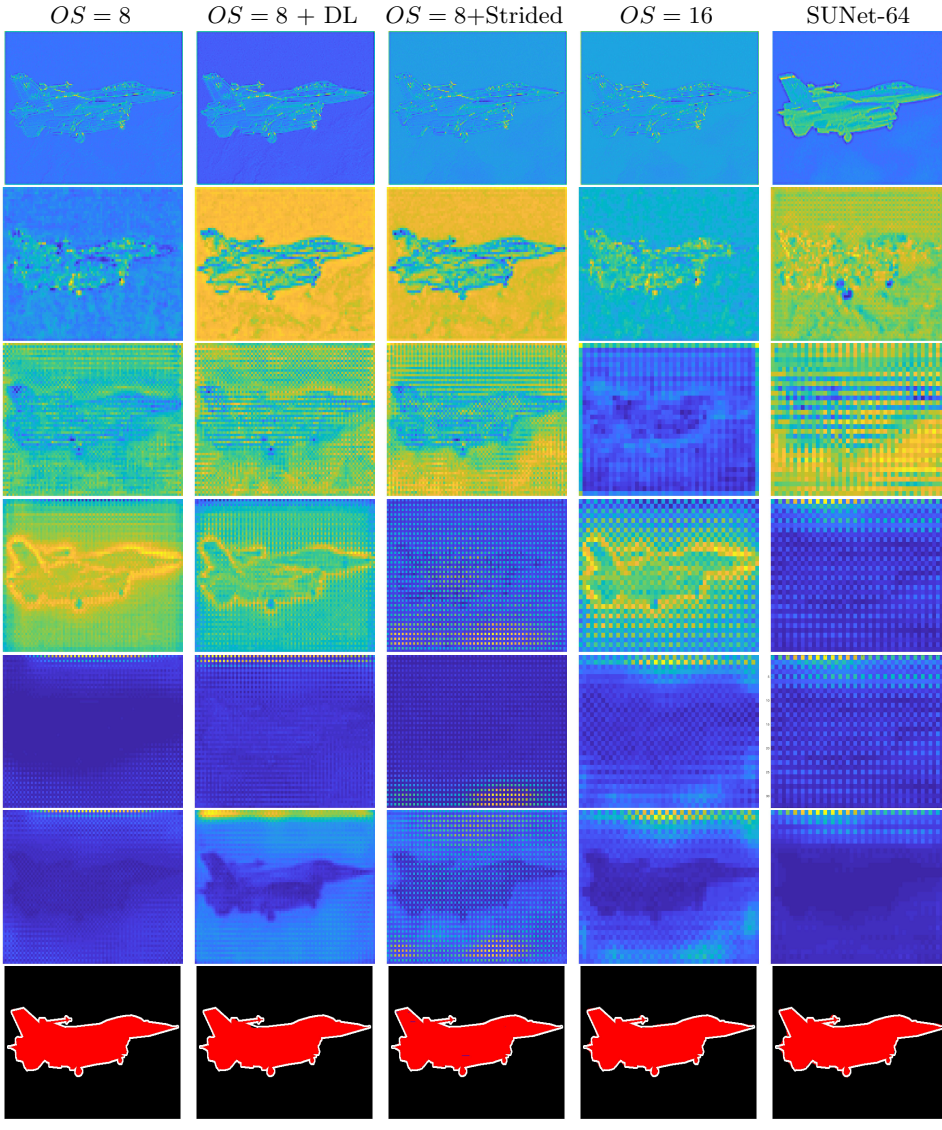

Fig. 5. Activation map recorded at the end of each level of the dilated SUNet for an example input image of an 'Aeroplane'. The activation map with total highest magnitude were selected from among all feature map outputs at the corresponding layer. **Top** to **Bottom**: output at end of level $1-6$ followed by classification output. The level 6 output is simply a prediction map before bilinear interpolation.

References

1. Lin, G., Milan, A., Shen, C., Reid, I.: Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In: IEEE Conference on Computer Vision

and Pattern Recognition (CVPR). (2017) 1, 2, 3, 4, 8, 10, 15
2. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 1520–1528 1, 3, 4
3. Fu, J., Liu, J., Wang, Y., Lu, H.: Stacked deconvolutional network for semantic segmentation. arXiv preprint arXiv:1708.04943 (2017) 1, 2, 3, 4, 5, 10
4. Jégou, S., Drozdzal, M., Vazquez, D., Romero, A., Bengio, Y.: The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In: Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on, IEEE (2017) 1175–1183 1, 3, 4
5. Ghiasi, G., Fowlkes, C.C.: Laplacian pyramid reconstruction and refinement for semantic segmentation. In: European Conference on Computer Vision, Springer (2016) 519–534 1, 2, 3, 4, 15
6. Krähenbühl, P., Koltun, V.: Efficient inference in fully connected crfs with gaussian edge potentials. In: Advances in neural information processing systems. (2011) 109–117 2, 3
7. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. arXiv preprint arXiv:1606.00915 (2016) 2, 3, 9, 10, 12, 15
8. Liang-Chieh, C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.: Semantic image segmentation with deep convolutional nets and fully connected crfs. In: International Conference on Learning Representations. (2015) 2, 3
9. Chandra, S., Kokkinos, I.: Fast, exact and multi-scale inference for semantic image segmentation with deep gaussian crfs. In: European Conference on Computer Vision, Springer (2016) 402–418 2, 3
10. Chandra, S., Usunier, N., Kokkinos, I.: Dense and low-rank gaussian crfs using deep embeddings. In: ICCV 2017-International Conference on Computer Vision. (2017) 2, 3, 10
11. Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P.H.: Conditional random fields as recurrent neural networks. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 1529–1537 2, 3, 4, 8, 10
12. Schwing, A.G., Urtasun, R.: Fully connected deep structured networks. arXiv preprint arXiv:1503.02351 (2015) 2, 3
13. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122 (2015) 2, 3, 4, 9, 10
14. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). (2017) 2881–2890 2, 3, 8, 10, 13, 15
15. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587 (2017) 2, 3, 8, 9, 10, 11, 13, 15
16. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention, Springer (2015) 234–241 2, 3, 4, 5, 6
17. Peng, C., Zhang, X., Yu, G., Luo, G., Sun, J.: Large kernel matters–improve semantic segmentation by global convolutional network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2017) 4353–4361 2, 15

18. Wang, P., Chen, P., Yuan, Y., Liu, D., Huang, Z., Hou, X., Cottrell, G.: Understanding convolution for semantic segmentation. arXiv preprint arXiv:1702.08502 (2017) 2, 3, 9, 15
19. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2015) 3431–3440 3, 4
20. Liu, W., Rabinovich, A., Berg, A.C.: Parsenet: Looking wider to see better. arXiv preprint arXiv:1506.04579 (2015) 3
21. Lin, G., Shen, C., Van Den Hengel, A., Reid, I.: Efficient piecewise training of deep structured models for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 3194–3203 3, 15
22. Chen, L.C., Yang, Y., Wang, J., Xu, W., Yuille, A.L.: Attention to scale: Scale-aware semantic image segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 3640–3649 3
23. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2017) 764–773 3, 9
24. Singh, B., Davis, L.S.: An analysis of scale invariance in object detection-snip. arXiv preprint arXiv:1711.08189 (2017) 3, 4
25. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 770–778 3, 4, 6, 8, 10
26. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014) 3, 4
27. Huang, G., Liu, Z., Weinberger, K.Q., van der Maaten, L.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. Volume 1. (2017) 3 3, 4, 6, 8, 10
28. Badrinarayanan, V., Kendall, A., Cipolla, R.: Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE transactions on pattern analysis and machine intelligence **39**(12) (2017) 2481–2495 3, 4
29. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. arXiv preprint arXiv:1802.02611 (2018) 3, 13, 15
30. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2017) 1251–1258 3, 4, 15
31. Shrivastava, A., Sukthankar, R., Malik, J., Gupta, A.: Beyond skip connections: Top-down modulation for object detection. arXiv preprint arXiv:1612.06851 (2016) 3, 4
32. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR. Volume 1. (2017) 4 3, 4
33. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: European Conference on Computer Vision, Springer (2016) 630–645 4, 10
34. Milletari, F., Navab, N., Ahmadi, S.A.: V-net: Fully convolutional neural networks for volumetric medical image segmentation. In: 3D Vision (3DV), 2016 Fourth International Conference on, IEEE (2016) 565–571 4
35. Çiçek, Ö., Abdulkadir, A., Lienkamp, S.S., Brox, T., Ronneberger, O.: 3d u-net: learning dense volumetric segmentation from sparse annotation. In: International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer (2016) 424–432 4

36. Riegler, G., Ulusoy, A.O., Bischof, H., Geiger, A.: Octnetfusion: Learning depth fusion from data. In: Proceedings of the International Conference on 3D Vision. (2017) 4
37. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. CVPR (2017) 4
38. Newell, A., Yang, K., Deng, J.: Stacked hourglass networks for human pose estimation. In: European Conference on Computer Vision, Springer (2016) 483–499 4, 5, 6
39. Yu, F., Koltun, V., Funkhouser, T.: Dilated residual networks. In: Computer Vision and Pattern Recognition. Volume 1. (2017) 6, 7, 9, 10
40. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks **5**(2) (1994) 157–166 6
41. Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv preprint arXiv:1605.07146 (2016) 8
42. Brandt, A.: Multi-level adaptive solutions to boundary-value problems. Mathematics of computation **31**(138) (1977) 333–390 9
43. Briggs, W.L., McCormick, S.F., et al.: A multigrid tutorial. Volume 72. Siam (2000) 9
44. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision. (2015) 1026–1034 10
45. Everingham, M., Eslami, S.A., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes challenge: A retrospective. International journal of computer vision **111**(1) (2015) 98–136 10
46. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 3213–3223 10
47. Hariharan, B., Arbelaez, P., Bourdev, L., Maji, S., Malik, J.: Semantic contours from inverse detectors. In: International Conference on Computer Vision (ICCV). (2011) 11
48. Loshchilov, I., Hutter, F.: Sgdr: stochastic gradient descent with restarts. arXiv preprint arXiv:1608.03983 (2016) 11
49. Wu, Z., Shen, C., Hengel, A.v.d.: Wider or deeper: Revisiting the resnet model for visual recognition. arXiv preprint arXiv:1611.10080 (2016) 13
50. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision, Springer (2014) 740–755 13
51. Li, X., Liu, Z., Luo, P., Change Loy, C., Tang, X.: Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2017) 3193–3202 15
52. Qi, H., Zhang, Z., Xiao, B., Hu, H., Cheng, B., Wei, Y., Dai, J.: Deformable convolutional networks – coco detection and segmentation challenge 2017 entry. ICCV COCO Challenge Workshop (2017) 15
53. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. arXiv preprint arXiv:1707.01083 (2017) 15

54. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017) 15
55. Li, H., Xu, Z., Taylor, G., Goldstein, T.: Visualizing the loss landscape of neural nets. arXiv preprint arXiv:1712.09913 (2017) 15