Smoothed Dilated Convolutions for Improved Dense Prediction

Zhengyang Wang
Washington State University
Pullman, Washington, USA
zwang6@eecs.wsu.edu

Shuiwang Ji
Washington State University
Pullman, Washington, USA
sji@eecs.wsu.edu

ABSTRACT

Dilated convolutions, also known as atrous convolutions, have been widely explored in deep convolutional neural networks (DCNNs) for various tasks like semantic image segmentation, object detection, audio generation, video modeling, and machine translation. However, dilated convolutions suffer from the gridding artifacts, which hampers the performance of DCNNs with dilated convolutions. In this work, we propose two simple yet effective degridding methods by studying a decomposition of dilated convolutions. Unlike existing models, which explore solutions by focusing on a block of cascaded dilated convolutional layers, our methods address the gridding artifacts by smoothing the dilated convolution itself. By analyzing them in both the original operation and the decomposition views, we further point out that the two degridding approaches are intrinsically related and define separable and shared (SS) operations, which generalize the proposed methods. We evaluate our methods thoroughly on two datasets and visualize the smoothing effect through effective receptive field analysis. Experimental results show that our methods yield significant and consistent improvements on the performance of DCNNs with dilated convolutions, while adding negligible amounts of extra training parameters.

CCS CONCEPTS

• **Computing methodologies** → **Structured outputs**; **Neural networks**; *Artificial intelligence*;

KEYWORDS

Deep learning; dilated convolutions; atrous convolutions; gridding artifacts

ACM Reference Format:
Zhengyang Wang and Shuiwang Ji. 2018. Smoothed Dilated Convolutions for Improved Dense Prediction. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3219944

1 INTRODUCTION

Dilated convolutions, also known as atrous convolutions, have been widely explored in deep convolutional neural networks (DCNNs) for various tasks, including semantic image segmentation [2, 3, 9–11, 18, 28–31], object detection [6, 15, 25, 26], audio generation [24], video modeling [17], and machine translation [16]. The idea of dilated filters was developed in the *algorithm à trous* for efficient wavelet decomposition in [14] and has been used in image pixel-wise prediction tasks to allow efficient computation [10, 18, 25, 26]. Dilation upsamples convolutional filters by inserting zeros between weights, as illustrated in Figure 1. It enlarges the receptive field, or field of view [2, 3, 11], but does not require training extra parameters in DCNNs. Dilated convolutions can be used in cascade to build multi-layer networks [16, 17, 24]. Another advantage of dilated convolutions is that they do not reduce the spatial resolution of responses. This is a key difference from downsampling layers, such as pooling layers or convolutions with stride larger than one, which also expand the receptive field of subsequent layers but also reduce the spatial resolution. This allows the transfer of classification models trained on ImageNet [7, 13] to semantic image segmentation tasks by removing downsampling layers and applying dilation in convolutions of subsequent layers [2, 3, 11, 21, 28–31]. Similar to standard convolutions, a layer consisting of a dilated convolution with an activation function is called a dilated convolutional layer.

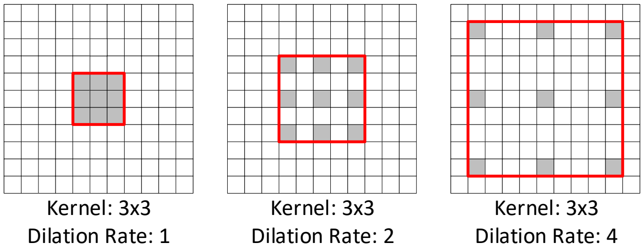

Figure 1: 2-D Dilated convolutions with a kernel size of 3×3. Note that when the dilation rate is 1, dilated convolutions are the same as standard convolutions. Dilated convolutions enlarge the receptive field while keeping the spatial resolution.

While DCNNs with dilated convolutions achieved success in a wide variety of deep learning tasks, it has been observed that dilations result in the so-called "gridding artifacts" [11, 28, 30]. For dilated convolutions with dilation rates larger than one, adjacent units in the output are computed from completely separate sets of units in the input. It results in inconsistency of local information and hampers the performance of DCNNs with dilated convolutions. As dilated convolutional layers are commonly stacked together in cascade in DCNNs, existing models focus on smoothing such gridding artifacts for a block of cascaded dilated convolutional layers. In [11, 30] the gridding problem was alleviated by adding more

KDD '18, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3219944

layers with millions of extra training parameters after the block of dilated convolutions. In [28] the hybrid dilated convolution (HDC) was proposed, which applies different dilation rates without a common factor for continuous dilated convolutional layers.

In this work, we address the gridding artifacts by smoothing the dilated convolution itself, instead of a block of stacked dilated convolutional layers. Our methods enjoy the unique advantage of being able to replace any single dilated convolutional layer in existing networks as they do not rely on other layers to solve the gridding problem. More importantly, our methods add minimal numbers of extra parameters to the model while some other degridding approaches increase the model parameters dramatically [11, 30]. Our methods are based on an interesting view of the dilated convolutional operation [1, 2, 27], which benefits from a decomposition of the operations. Based on this novel interpretation of dilated convolutions, we propose two simple yet effective methods to smooth the gridding artifacts. By analyzing these two methods in both the original operation and the decomposition views, we further notice that they are intrinsically related and define separable and shared (SS) operations that generalize the proposed methods. Experimental results show that our methods improve current DCNNs with dilated convolutions significantly and consistently, while only adding a few hundred extra parameters. We also employ the effective receptive field (ERF) analysis [22] to visualize the smoothing effect for DCNNs with our dilated convolutions.

2 BACKGROUND AND RELATED WORK

In this section, we describe dilated convolutions and DCNNs with them. We then discuss the gridding problem and current solutions in detail.

2.1 Dilated Convolutions

In the one-dimensional case, given a 1-D input f, the output o at location i of a dilated convolution with a filter w of size S is defined as

$$o[i] = \sum_{s=1}^{S} f[i + r \cdot s]w[i], \tag{1}$$

where r is known as the dilation rate. Higher dimensional cases can be easily generalized. When $r = 1$, dilated convolutions correspond to standard convolutions. An intuitive and direct way to understand dilated convolutions is that $r - 1$ zeros are inserted between every two adjacent weights in the standard convolutional filters. Dilated convolutions are also known as atrous convolutions in which "trous" means holes in French. Figure 1 contains an illustration of dilated convolution in the two-dimensional case.

As mentioned in Section 1, in most cases, DCNNs use dilated convolutions in cascade, which means several dilated convolutional layers are stacked together. The reasons for using this cascaded pattern differ for different tasks. In the task of semantic image segmentation [2, 3, 11, 21, 28–31], in order to have output feature maps of larger sizes while maintaining the size of the receptive field, dilated convolutions are employed to replace standard convolutions in layers after the removed downsampling layers. For example, if we treat standard convolutions as dilated convolutions with a dilation rate of $r = 1$, when a downsampling layer with a subsampling rate of 2 is removed, the dilation rates of all subsequent convolutional layers should be multiplied by 2. This results in dilated convolutional layers with dilation rates of $r = 2, 4, 8$, etc. In other tasks, such as audio generation [24], video modeling [17], and machine translation [16], the use of dilated convolutions aims at enlarging the receptive fields of outputs. As pointed out in [17, 24, 29], cascaded dilated convolutional layers expand the receptive field exponentially in the number of layers in DCNNs, as opposed to linearly. In these studies, the dilation rate is doubled for every forward layer, starting from 1 up to a limit before the pattern is repeated.

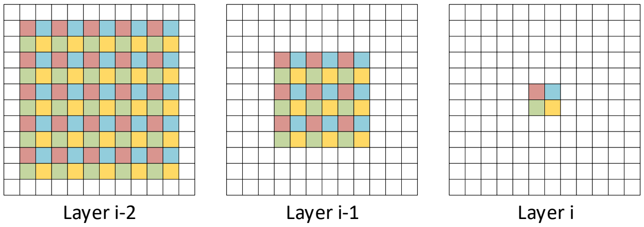

Figure 2: An illustration of gridding artifacts. The operations between layers are both dilated convolutions with a kernel size of 3×3 and a dilation rate of $r = 2$. For four neighboring units in layer i indicated by different colors, we mark their actual receptive fields in layer $i - 1$ and $i - 2$ using the same color, respectively. Clearly, their actual receptive fields are completely separate sets of units.

Note that when using dilated convolutions in cascade, the gridding artifacts affect the models more significantly. This is because the dilation rates of continuously stacked layers have a common factor of 2 in all of these DCNNs that use dilated convolutional layers in cascade, as discussed in [28] and Section 2.2. In [2, 3] dilated convolutions in parallel to form the output layer were explored.

2.2 Gridding in Dilated Convolutions

Dilated convolutions with dilation rates larger than one will produce the so called gridding artifacts; that is, adjacent units in the output are computed from completely separate sets of units in the input and thus have totally different actual receptive fields. To view the gridding problem clearly, we first look into a single dilated convolution. Considering the second case in Figure 1 as an example, a 2-D dilated convolution with a kernel size of 3 × 3 and a dilation rate of $r = 2$ has a 5 × 5 receptive field. However, the number of pixels that are actually involved in the computation is only 9 out of 25, which implies that the actual receptive field is still 3 × 3, but sparsely distributed. If we further consider the neighboring units in the output, the gridding problem can be seen from Figure 2. Suppose we have two consecutive dilated convolutional layers in cascade, and both dilated convolutions have a kernel size of 3 × 3 and a dilation rate of $r = 2$. For four adjacent units indicated by different colors in layer i, we show their actual receptive fields in layer $i - 1$ and $i - 2$ using the same color. We can see that four completely separate sets of units in layer $i - 1$ contribute to the computation of the four units in layer i. Moreover, since the dilation rates for both layers are 2, which have a common factor of 2, the gridding problem also exists in layer $i - 2$. Indeed, whenever the dilation rates of dilated convolutional layers in cascade have a common factor relationship, such as 2, 2, 2 or 2, 4, 8, the gridding problem is propagated to all layers, as pointed out in [28]. For a block of such layers, neighboring outputs of the block are computed from totally different sets of inputs. This results in the inconsistency of local information and hampers the performance of DCNNs with dilated convolutions.

The gridding artifacts were observed and addressed in several recent studies for semantic image segmentation [11, 28, 30]. As described in Section 2.1, dilated convolutions are mostly employed in cascade in DCNNs. Therefore, these studies focused on solving the gridding problem in terms of a block of stacked dilated convolutional layers. Specifically, hybrid dilated convolution (HDC) was proposed in [28], which groups several dilated convolutional layers and applies dilation rates without a common factor relationship. For example, for a block of dilated convolutions with a dilation rate of $r = 2$, every three consecutive layers are grouped together and the corresponding dilation rates are changed to 1, 2, 3 instead of 2, 2, 2. For a similar block with a dilation rate of $r = 4$, the same grouping principle is applied and the dilation rates become 3, 4, 5, instead of 4, 4, 4. When used together with their proposed dense upsampling convolution (DUC), this approach improved DCNNs for semantic image segmentation. This strategy was also adopted as the "multi-grid" method in recent work [3]. Prior to [28], the degridding was performed mainly by adding more layers after the block of dilated convolutional layers [11, 30]. It was proposed in [30] to add two more standard convolutional layers without residual connections while [11] proposed to add a block of dilated convolutional layers with decreasing dilation rates. The main drawback of such methods is the requirement for learning a large amount of extra parameters.

3 SMOOTHED DILATED CONVOLUTIONS

In this section, we discuss a decomposition view of dilated convolutions. We then propose two approaches for smoothing the gridding artifacts. We also analyze the relationship between the proposed two methods and define separable and shared (SS) operations to generalize them.

3.1 A Decomposition View of Dilated Convolutions

There are two ways to understand dilated convolutions. As introduced in Section 2.1, the first and more intuitive way is to think of dilated convolutional filters with dilation rate r as upsampled standard convolutional filters, by inserting zeros (holes) [25]. Another way to view dilated convolutions is based on a decomposition of the operation [27]. A dilated convolution with a dilation rate of r can be decomposed into three steps. First, the input feature maps are periodically subsampled by a factor of r. As a result, the inputs are deinterlaced to r^d groups of feature maps of reduced resolution, where d is the spatial dimension of the inputs. Second, these groups of intermediate feature maps are fed into a standard convolution. This convolution has filters with the same weights as the original dilated convolution after removing all inserted zeros. More importantly, it is shared for all the groups, which means each group of reduced resolution maps goes through the same standard convolution. The third step is to reinterlace the r^d groups of feature maps to the original resolution and produce the outputs of the dilated convolution.

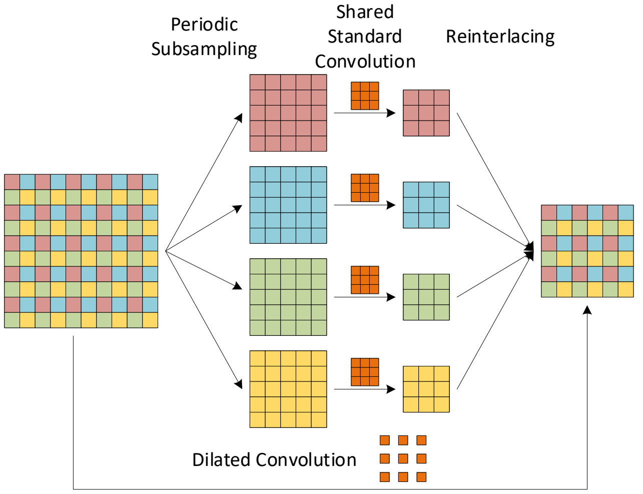

Figure 3: An example of the decomposition of a dilated convolution with a kernel size of 3×3 and a dilation rate of $r = 2$ on a 2-D input feature map. The decomposition has three steps; namely periodic subsampling, shared standard convolution and reinterlacing. This example will also be used in Figures 3 to 7.

Figure 3 gives an example of the decomposition in the 2-D case. To simplify the discussion, we assume the number of input channels and output channels is both 1. Given a 10×10 feature map, a dilated convolution with a kernel size of 3×3 and a dilation rate of $r = 2$ will output a 6×6 feature map without any padding. In the decomposition of this dilated convolution, the input feature map is periodically subsampled into $2^2 = 4$ groups of 5×5 feature maps of reduced resolution. Then a shared standard convolution, which has the same weights as the dilated convolution without padding, is applied to these 4 groups of feature maps and obtains 4 groups of 3×3 feature maps. Finally, they are reinterlaced to the original resolution and produce exactly the same 6×6 output feature map as the original dilated convolution. This decomposition reduces dilated convolutions into standard convolutions and allows more efficient implementation [1, 2, 10, 26].

We notice that the decomposition view provides a clear explanation of the gridding artifacts; that is, the r^d groups of intermediate feature maps, either before or after the shared standard convolution, have no dependency among each other and thus collect potentially inconsistent local information. Based on this insight, we overcome gridding by adding dependencies among the r^d groups in different steps of the decomposition. We propose two effective approaches in the next two sections.

3.2 Smoothed Dilated Convolutions by Group Interaction Layers

Our first degridding method attempts to build dependencies among different groups in the third step of the decomposition. We propose to add a group interaction layer before reinterlacing the intermediate feature maps to the original resolution. For a dilated convolution with a dilation rate of r on d-dimensional input feature maps, the second step of the decomposition produces r^d groups of feature maps of reduced resolution, denoted as $\{f_i\}_{i=1}^{r^d}$, after the shared convolution. Note that each f_i represents a group of feature maps, rather than a single feature map. We define a group interaction layer with a weight matrix $W \in \mathbb{R}^{r^d \times r^d}$ given as

$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} & \cdots & w_{1,r^d} \\ w_{21} & w_{22} & w_{23} & \cdots & w_{2,r^d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{r^d,1} & w_{r^d,2} & w_{r^d,3} & \cdots & w_{r^d,r^d} \end{bmatrix}. \quad (2)$$

The outputs of this layer are still r^d groups of feature maps, denoted as $\{\hat{f}_i\}_{i=1}^{r^d}$, computed by

$$\hat{f}_i = \sum_{j=1}^{r^d} w_{ij} \cdot f_j, \quad (3)$$

for $i = 1, 2, \ldots, r^d$. Note that the connections of this layer are between groups instead of feature maps. In fact, every $\hat{f}_i$ is a linear combination of $\{f_i\}_{i=1}^{r^d}$, weighted by the weight matrix W. Through this layer, each $\hat{f}_i$ collects local information from all r^d groups of feature maps, which adds dependencies among different groups. After the group interaction layer, the r^d groups are reinterlaced to the original resolution and form the final output of the dilated convolutions. The number of extra training parameters in such smoothed dilated convolutions is r^{2d}, independent of the number of input and output channels. DCNNs with dilated convolutions are commonly used in one-dimensional or two-dimensional cases, which means $d = 1, 2$. In practice, choices of r are usually 2, 4, 8. The proposed group interaction layer only requires learning thousands of extra parameters in the worst cases, while the original dilated convolutions usually have millions of training parameters.

We use the same example in Section 3.1 to illustrate the idea in Figure 4. Given the outputs of the second step in the decomposition, the 4 groups of intermediate feature maps build dependencies among each other through the group interaction layer, whose number of weights is only $2^{2 \cdot 2} = 16$, represented by 16 connections. We use the gray color to represent feature maps after degridding.

3.3 Smoothed Dilated Convolutions by Separable and Shared Convolutions

We further explore an approach to establish dependencies among different groups in the first step of the decomposition; that is, before deinterlacing the input feature maps. Considering a dilated convolution with a dilation rate of r on d-dimensional input feature maps, the periodic subsampling during deinterlacing distributes each unit in a local area of size r^d in the inputs to a separate group. Therefore, for units in a particular group, all the neighboring units are in the other independent $r^d - 1$ groups, thereby resulting in local inconsistency. If the local information can be incorporated before periodic sampling, it is possible to alleviate the gridding artifacts. In order to achieve this, we propose separable and shared (SS) convolutions, based on separable convolutions [4, 23]. Given inputs of C channels and corresponding outputs of C channels, separable convolutions are the same as standard convolutions, except that separable convolutions handle each channel separately. Standard convolutions connect all C channels in inputs to all C channels in outputs, leading to C^2 different filters. In contrast, separable convolutions only connect the ith output channel to the ith input channel, yielding only C filters. In the proposed SS convolutions, "shared" means that, based on separable convolutions, the C filters are the same and shared by all pairs of input and output channels. For inputs and outputs of C channels, SS convolutions only have one filter scanning all spatial locations and share this filter across all channels. In terms of smoothing dilated convolutions, we apply SS convolutions to incorporate neighboring information for each unit in the input feature maps. Specifically, an SS convolution with a kernel size of $(2r - 1)^d$ is inserted before deinterlacing, thereby adding dependencies among each other to the r^d groups of feature maps produced by periodic subsampling.

The example in Figure 5 illustrates the idea of inserting SS convolutions. Here, the kernel size of the inserted SS convolution is $(2 \cdot 2 - 1)^2 = 3 \times 3$. Note that because the inputs only have one channel, SS convolutions, separable convolutions and standard convolutions are equivalent in this example. However, they become different if the inputs have $C > 1$ channels. Importantly, for inputs with multiple channels, the number of training parameters does not change for SS convolutions, as opposed to the other two

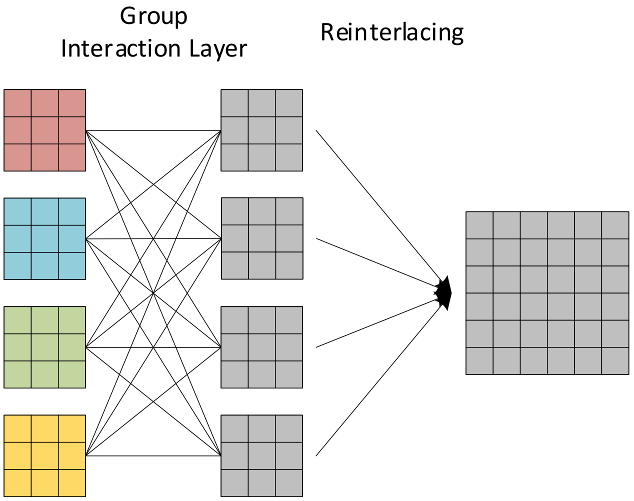

Figure 4: Illustration of the degridding method in Section 3.2 for a dilated convolution with a kernel size of 3×3 and a dilation rate of $r = 2$ on a 2-D input feature map. By using a group interaction layer before reinterlacing, dependencies among intermediate groups are established. The same gray color denotes consistent local information.

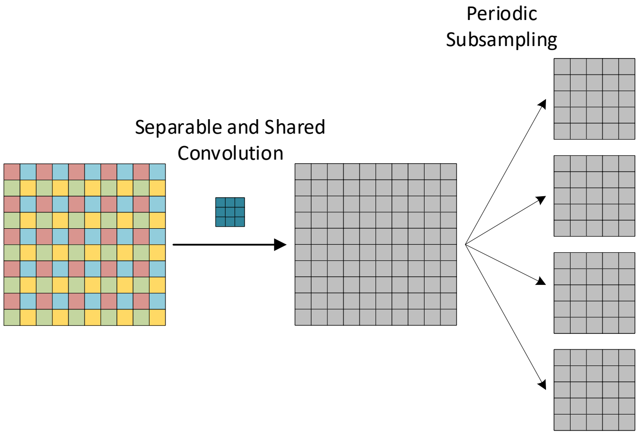

Figure 5: Illustration of the degridding method in Section 3.3 for a dilated convolution with a kernel size of 3×3 and a dilation rate of $r = 2$ on a 2-D input feature map. By adding the separable and shared convolution, the 4 groups created by periodic subsampling have dependencies among each other. The same gray color represents smoothed feature maps.

kinds of convolutions. It means the proposed degridding method has $(2r-1)^d$ parameters, independent of the number of channels, which corresponds to only tens of extra parameters at most in practice.

3.4 Relationship between the Two Methods

Both of the proposed approaches are derived from the decomposition view of dilated convolutions. Now we combine all steps and analyze them in view of the original operation. For the second method in Section 3.3, it is straightforward as the separable and shared (SS) convolution is inserted before the first step of decomposition and actually does not affect the original dilated convolution. Consequently, it is equivalent to adding an SS convolution before the dilated convolution, as shown in Figure 7. However, the first method in Section 3.2 performs degridding through the group-wise fully-connected layer between the second and the third steps of the decomposition. To see how to perform the combination, we refer to the example in Figure 4. Before the final step, we have four groups of feature maps and each group has only one feature map. Considering the units in the upper left corner of the four feature maps, without the group interaction layer, these four units form the upper left 2×2 block of the output feature map after reinterlacing. If we insert the group-wise fully-connected layer, the four new units in the upper left corner become linear combinations of the previous ones and form the upper left 2×2 block of the output feature map instead. As a result, the new upper left 2×2 block of the output feature map is computed by a fully-connected operation on the previous one. By examining other units, we find that the fully-connected operation is shared for every non-overlapping 2×2 blocks, scanning the output feature map with a stride of 2. Figure 6 provides an illustration. By generalizing this example, we can see that the degridding method is equivalent to a dilated convolution followed by the following operation: use a window of size r^d to scan the output feature map with stride r and obtain non-overlapping blocks; for each block, perform the same fully-connected operation that outputs a block of the same spatial size. Note that if the outputs have multiple channels, the operation is shared across channels. This operation is similar to the SS convolution as they both scan spatial locations using a single kernel shared across all channels. Thus, we name it as the SS block-wise fully-connected layer. Based on it as well as the SS convolution, we further define operations which scan spatial locations of inputs using a single filter shared across all channels as SS operations.

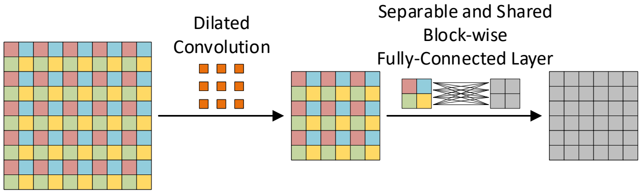

Figure 6: Another illustration of the proposed method in Section 3.2, corresponding to Figure 4. The method is equivalent to adding an SS block-wise fully-connected layer after the dilated convolution.

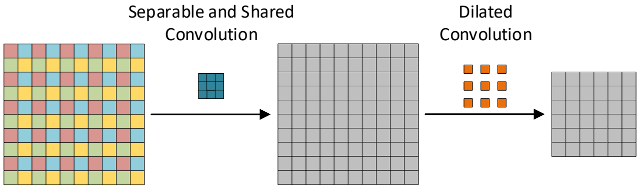

Figure 7: Another illustration of the proposed method in Section 3.3, corresponding to Figure 5. The method is equivalent to inserting an SS convolution before the dilated convolution.

As DCNNs commonly employ dilated convolutional layers in cascade, we also look into our proposed methods in this case. As explained above, the first degridding approach is equivalent to adding an SS block-wise fully-connected layer after the dilated convolution, while the second one corresponds to inserting an SS convolution before the dilated convolution. However, for a block of cascaded dilated convolutional layers with the same dilation rate, the order between the dilated convolution and the SS operation only affects the very first and last layers. As a result, the two proposed degridding methods can be generalized as combining appropriate SS operations with dilated convolutions.

4 EXPERIMENTAL STUDIES

In this section, we evaluate our methods on the PASCAL VOC 2012 [8] and Cityscapes [5] datasets. Our proposed approaches result in significant and consistent improvements for DCNNs with dilated convolutions. We also perform the effective receptive field (ERF) analysis [22] to visualize the smoothing effect.

4.1 Basic Setup

To conduct our experiments, we choose the task of semantic image segmentation because the gridding artifacts were mainly observed in studies for this task [11, 28, 30]. The consistency of local information is important for such a pixel-wise prediction task on images. In addition, the smoothing effect is easy to visualize on two-dimensional data.

The baseline model in our experiments is the DeepLabv2 [2] with ResNet-101 [13]. It is a fair benchmark to evaluate our smoothed dilated convolutions in three aspects. First, it employed dilated convolutions to adapt ResNet pre-trained on ImageNet [7]; namely from image classification to semantic image segmentation. Most semantic image segmentation models adopted this transfer learning strategy [2, 3, 10, 11, 18, 21, 28–31] and ResNet is one of the most accurate DCNNs for image classification with pre-trained models available. Second, models that achieved the state-of-the-arts in segmentation tasks recently [3, 28, 31] were developed from DeepLabv2. In [31] the output layer was replaced with a pyramid pooling module. [28] also changed the output layer and additionally proposed changing dilation rates, as mentioned in Section 2.2. The current best model [3] followed the suggestions of [28] and meanwhile, explored going deeper with more dilated convolutional blocks. Third, we intend to compare our degridding methods with existing approaches [11, 28, 30]. While [11, 30] addressed the gridding artifacts by adding more layers that considerably increased the number of training parameters, our methods only require learning hundreds of extra parameters. Thus, we perform the comparison with the idea proposed in [28], which is based on DeepLabv2.

DeepLabv2 is composed of two parts: the encoder and the output layers. The encoder is a pre-trained ResNet-101 model modified with dilated convolutions, and it extracts feature maps from raw images. As introduced in Section 2.1, the last two downsampling layers in ResNet-101 were removed and subsequent standard convolutional layers were replaced by dilated convolutional layers with dilation rates of $r = 2, 4$, respectively. To be specific, after the modification, the last two blocks are a block of 23 stacked dilated convolutional layers with a dilation rate of $r = 2$ followed by a block of 3 cascaded dilated convolutions with a dilation rate of $r = 4$. The output layer performs pixel-wise classification by aggregating information from the output feature maps of encoder.

We re-implement DeepLabv2 in Tensorflow and perform experimental studies based on our implementation. Our code is publicly available[1]. We improve the baseline by addressing the gridding artifacts in the last two blocks of the encoder. To make the comparison independent of the output layer, we conduct experiments with different output layers. In order to eliminate the bias of different datasets, we evaluate our methods on two datasets. All the models are evaluated by pixel intersection over union (IoU) defined as

$$IoU = \frac{true\_positive}{true\_positive + false\_positive + false\_negative}. \quad (4)$$

4.2 PASCAL VOC2012

The PASCAL VOC 2012 semantic image segmentation dataset [8] provides pixel-wise annotated natural images. It has been split into *train*, *val* and *test* sets with 1, 464, 1, 449 and 1, 456 images, respectively. The annotations include 21 classes, which are 20 foreground object classes and 1 class for background. An augmented version with extra annotations [12] increases the size of the *train* set to 10, 582. In our experiments, we train all the models using the augmented *train* set and evaluate them on the *val* set. When reproducing the baseline DeepLabv2, we do not employ multi-scale inputs with max fusion for testing due to our limited GPU memory. We perform no post-processing such as conditional random fields (CRF) [2], which is not related to our goals. Following DeepLabv2, we train the model with randomly cropped patches of size of 321 × 321 and batch size of 10. Data augmentation by randomly scaling the inputs for training is applied. We set the initial learning rate to 0.00025 and adopt the “poly” learning rate policy [20] as

$$current\_lr = (1 - \frac{iter}{max\_iter})^{power} \cdot initial\_lr, \quad (5)$$

where $power = 0.9$, $iter$ denotes current iteration number, and lr denotes learning rate, as in [2, 3, 28]. The model is trained for $max\_iter = 20,000$ iterations with a momentum of 0.9 and a weight decay of 0.0005.

We implement our proposed methods by inserting appropriate separable and shared (SS) operations before or after each dilated convolution as shown in Figures 6 and 7. An important step is to change the initial learning rate, detailed in each experiment. To make the comparisons solid, we also train the baseline with different initial learning rates and observe the original setting of 0.00025 yields the best performance. The initialization of SS operations is to set them to be identity operations. Specifically, for a group interaction layer with a dilation rate of $r = 2$, the initial filter is

$$W = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (6)$$

while for an SS convolution with a dilation rate of $r = 2$, it is

$$W = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \quad (7)$$

The original DeepLabv2 used pre-training on MS-COCO [19], which results in more training data and higher performances. Our experiments are conducted under both settings; namely with and without MS-COCO pre-training. The results are given in Tables 1 and 2, respectively. In the tables, “G Interact” denotes the degridding method with a group interaction layer, *i.e.*, adding an SS block-wise fully-connected layer after the dilated convolution and “SS Conv” represents the one with an SS convolution inserted before the dilated convolution. In these experiments with MS-COCO pre-training, the initial learning rates for “G Interact” and “SS Conv” are both 0.001. Otherwise, they are set to 0.001 and 0.00075, respectively. Clearly, both proposed methods improve the IoU for most classes as well as the mean IoU (mIoU) over the baseline under both settings. It is worth noting that “G Interact” only requires training 1, 136(= 16 × 23 + 256 × 3) extra parameters and “SS Conv” requires 354(= 9 × 23 + 49 × 3) extra parameters, which are negligible compared to the total number of parameters in the models.

[1] https://github.com/divelab/dilated/

Table 1: Experimental results of models with the ASPP output layer and MS-COCO pre-training on PASCAL VOC 2012 *val* set. Class 1 is the background class and Class 2 − 21 represent "aeroplane, bicycle, bird, boat, bottle, bus, car, cat, chair, cow, diningtable, dog, horse, motorbike, person, potteplant, sheep, sofa, train, tvmonitor", respectively. This is the same for Tables 1 to 4.

Models	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	mIoU
DeepLabv2	93.8	85.9	38.8	84.8	64.3	**79.0**	93.7	85.5	**91.7**	34.1	83.0	57.0	**86.1**	83.0	81.0	85.0	58.2	83.4	**48.2**	**87.2**	74.0	75.1
Multigrid	93.6	85.4	38.9	82.2	66.9	76.6	93.2	85.3	90.7	35.7	82.5	53.7	83.1	**84.2**	82.2	84.6	56.9	84.3	45.6	85.5	73.1	74.5
G Interact	93.7	**86.9**	**39.6**	84.1	**68.9**	76.4	**93.8**	86.2	**91.7**	**36.1**	**83.7**	55.3	85.7	84.0	82.2	84.9	**59.5**	**85.7**	46.5	85.0	73.0	**75.4**
SS Conv	**93.9**	86.7	39.5	**86.2**	68.1	77.3	**93.8**	**86.4**	91.5	35.4	83.2	**59.0**	85.2	83.6	**82.4**	**85.2**	57.3	82.1	45.8	86.1	**75.2**	**75.4**

Table 2: Experimental results of models with the ASPP output layer but no MS-COCO pre-training on PASCAL VOC 2012 *val* set.

Models	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	mIoU
DeepLabv2	92.9	85.0	38.1	82.8	66.2	76.5	91.1	82.7	88.4	**33.8**	77.7	49.9	80.7	78.6	77.9	82.0	51.5	76.6	43.1	82.8	66.6	71.7
Multigrid	92.8	84.9	37.4	81.8	65.6	76.0	90.4	81.3	86.9	32.6	76.8	52.3	80.2	79.5	77.4	81.9	50.7	78.4	41.9	82.7	66.0	71.3
G Interact	**93.0**	85.1	37.4	**83.4**	**66.9**	76.6	90.7	82.0	88.1	**33.8**	**81.1**	**54.3**	81.6	**80.2**	76.7	81.9	**53.7**	**78.7**	43.1	**83.9**	66.4	**72.3**
SS Conv	**93.0**	**85.8**	**38.3**	82.5	66.3	**77.9**	**91.6**	**83.5**	**88.5**	32.4	77.8	52.5	**81.9**	78.1	**79.3**	**82.1**	49.8	78.4	**44.4**	83.0	**67.9**	72.1

We also compare our methods with existing degridding method proposed in [28] and used in [3] as the "multigrid" method. As introduced in Section 2.2, the idea is to group several dilated convolutional layers and change the dilation factors. As we know, for the modified ResNet-101 with dilated convolutions, the last two blocks are a block of 23 stacked dilated convolutional layers with a dilation rate of $r = 2$ followed by a block of 3 cascaded dilated convolutions with a dilation rate of $r = 4$. For the first block, we group every 3 layers together and replace the dilation rates from $r = 2, 2, 2$ to $r = 1, 2, 3$. We keep $r = 2, 2$ for the left 2 layers. For the second block, the 3 dilation factors $r = 4, 4, 4$ are changed to $r = 3, 4, 5$. We make the modification and train the models under the same setting as the baseline. The results, denoted as "Multigrid", are shown in the second lines of Tables 1 and 2. Surprisingly, our implementation indicates that the approach does not improve the performance. An explanation of the results is that the method should be applied together with other modifications, as both [28] and [3] conduct experiments together with other changes over DeepLabv2, such as dense upsamling convolution (DUC) and deeper encoders.

As we address the gridding artifacts in the last two blocks of the encoder, we also run experiments with different output layers in order to make the comparisons independent of the output layer. We replace the original atrous spatial pyramid pooling (ASPP) output layer of DeepLabv2 by the large field of view (LargeFOV) layer, which was applied earlier in [2]. We train the models with the same settings above, with and without MS-COCO pre-training, and show the results in Tables 3 and 4, respectively. Again, the proposed degridding methods result in significant improvements consistently.

4.3 Cityscapes

We further compare our proposed methods on the Cityscapes dataset [5]. Cityscapes collects $5,000$ 2048×1024 images of street scenes from 50 different cities and provides high quality pixel-wise annotations of 19 classes. The $5,000$ images are divided into *train*, *val* and *test* with $2,975$, 500 and $1,525$ images, respectively. Again, we train models on the *train* set and perform evaluation on the *val* set. The training batch size is 3, where each batch contains randomly cropped patches of size 571×571. The initial learning rates for all models are set to 0.0005. All the other settings are the same as those in Section 4.2.

Experiments are still conducted under both settings, *i.e.*, with and without MS-COCO pre-training, and the results are given in Tables 5 and 6, respectively. We can see that both of the proposed methods increase the mIoU over the baseline, which shows that the improvements are independent of datasets.

4.4 Effective Receptive Field Analysis

Since we are addressing the gridding artifacts, we perform the effective receptive field (ERF) analysis [11, 22] to visualize the smoothing effect of our methods. These experiments further verify that the improvements of the proposed methods come from degridding. Given a block in DCNNs, the ERF analysis is an approach to characterize how much each unit in the input of the block affects a particular output unit of the block mathematically [22], instead of theoretically.

Following the steps in [11, 22], we analyze the models on PASCAL VOC 2012, with the ASPP output layer and MS-COCO pre-training. We compute the ERF for chosen blocks of the baseline and both of the proposed methods. Specifically, suppose the input and output feature maps of a block are x and y, respectively. The

Table 3: Experimental results of models with the LargeFOV output layer and MS-COCO pre-training on PASCAL VOC 2012 *val* set.

Models	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	mIoU
DeepLabv2	93.7	**85.7**	39.4	85.9	67.6	**79.0**	93.1	86.0	90.7	36.2	79.8	54.6	83.7	80.9	**81.4**	85.0	57.5	83.5	45.5	84.5	74.1	74.7
G Interact	**93.8**	85.5	**40.0**	86.5	67.5	78.1	92.9	86.2	90.4	**37.2**	80.6	**56.5**	82.6	80.3	81.0	85.0	58.1	**84.8**	**46.6**	84.4	74.8	74.9
SS Conv	**93.8**	85.3	39.7	**86.8**	**68.7**	77.9	**94.0**	**86.3**	**90.8**	35.2	**83.1**	55.4	**84.5**	**83.8**	79.6	**85.6**	**59.3**	83.2	46.2	**86.2**	**75.5**	**75.3**

Table 4: Experimental results of models with the LargeFOV output layer but no MS-COCO pre-training on PASCAL VOC 2012 *val* set.

Models	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	mIoU
DeepLabv2	92.8	84.1	37.9	82.9	65.2	**76.5**	89.9	82.7	87.9	33.2	74.9	50.2	80.6	76.6	78.6	82.1	52.2	77.4	40.8	80.1	66.6	71.1
G Interact	**93.0**	84.5	37.8	**84.2**	**66.5**	75.9	90.5	83.1	**88.4**	**34.6**	**75.4**	**52.3**	**81.7**	75.5	77.4	82.1	52.8	78.2	41.5	**81.7**	**67.9**	**71.7**
SS Conv	92.9	**85.5**	**38.1**	83.2	**66.5**	73.1	**91.2**	**84.0**	88.3	34.5	75.2	49.9	81.0	**77.2**	**79.5**	**82.5**	**53.7**	**78.6**	**42.0**	80.0	67.7	71.6

spatial locations of the feature maps are indexed by (i, j) with $(0, 0)$ representing the center. The ERF is measured by the partial derivative $\partial y_{0,0}/\partial x_{i,j}$. To compute it without an explicit loss function, we set the error gradient with respect to $y_{0,0}$ to 1 while for $y_{i,j}$ with $i \neq 0$ or $j \neq 0$, we set it to 0. Then the error gradient can be back-propagated to x and the error gradient with respect to $x_{i,j}$ equals to $\partial y_{0,0}/\partial x_{i,j}$ [22]. However, the results are input-dependent. So $\partial y_{0,0}/\partial x_{i,j}$ are computed for all images in the *val* set and their absolute values are averaged. Finally, we sum the values over all channels of x to get a visualization of the ERF.

In our experiments, we choose two blocks of the DCNNs to visualize the smoothing effect and enlarge the spatial size of visualizations ten times for display. The first block is the very last layer of the encoder, which is a dilated convolution with a kernel size of 3 × 3 and a dilation rate of $r = 4$. The ERF analysis results are presented in Figure 8. The ERF of the original dilated convolution in the baseline is obvious. It corresponds to a 3 × 3 filter with zeros inserted between non-zero weights. Such a filter results in the gridding problem. For our proposed degridding methods, we can see that they smooth the ERF and thus perform degridding. In addition, both methods expand the rectangular size of the ERF due to the SS operations. The second chosen block is the entire block composed of dilated convolutional layers, which includes the last two blocks of the encoder. Figure 9 shows the ERF visualization. The gridding artifacts are clearly smoothed in both proposed methods. In fact, only the leftmost visualization for the baseline has black pixels that represent zero weights. Particularly, we note that "SS FC" still has a grid-like visualization. A reason of this is the block-wise operation may result in larger grids in terms of blocks. Nevertheless, it alleviates the inconsistency of pixel-wise local information and improves DCNNs with dilated convolutions.

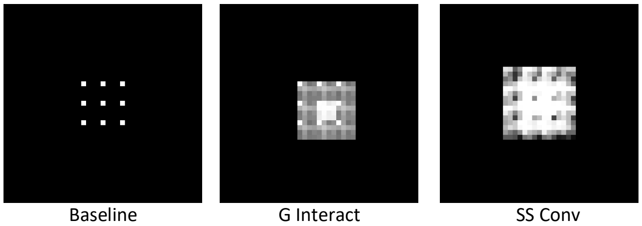

Figure 8: ERF visualization for the single dilated convolution with a kernel size of 3 × 3 and a dilation factor of $r = 4$. Black pixels represent zero weights.

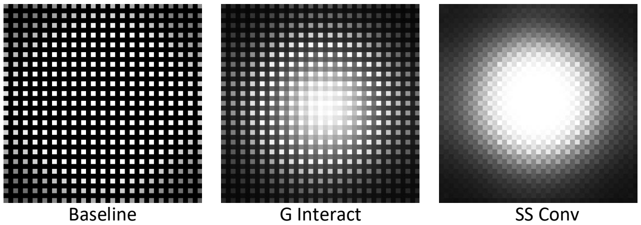

Figure 9: ERF visualization for the entire dilated convolutional block. Note that only the leftmost map has black pixels that represent zero weights.

5 CONCLUSIONS

In this work, we propose two simple yet effective degridding methods based on a decomposition of dilated convolutions. The proposed methods differ from existing degridding approaches in two aspects. First, we address the gridding artifacts in terms of a single dilated convolution operation instead of multiple layers in cascade. Second, our methods only require learning a negligible amount of extra parameters. Experimental results show that they improve DCNNs with dilated convolutions significantly and consistently. The smoothing effect is also visualized in the effective receptive field (ERF) analysis. Through further analysis, we relate both proposed methods together and define the separable and shared operations. The newly defined separable and shared convolution operation is a general neural network operation and may result in a general degridding strategy. We will explore this direction in our

Table 5: Experimental results of models with the ASPP output layer and MS-COCO pre-training on Cityscapes *val* set. Class $1 - 19$ **represent "road, sidewalk, building, wall, fence, pole, traffic light, traffic sign, vegetation, terrain, sky, person, rider, car, truck, bus, train, motorcycle, bicycle", respectively. This is the same for Tables 5 and 6.**

Models	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	mIoU
DeepLabv2	97.2	**79.7**	90.1	47.4	49.2	50.3	57.3	69.0	**90.6**	59.8	**92.8**	75.9	55.6	92.5	67.5	80.5	64.8	59.7	71.7	71.1
G Interact	**97.3**	79.6	90.2	50.4	49.9	**50.5**	**58.5**	69.1	90.5	58.7	92.7	75.9	55.4	92.5	70.9	80.2	65.0	**60.6**	**71.8**	71.6
SS Conv	97.2	**79.7**	**90.3**	**51.1**	**50.5**	50.2	58.1	**69.3**	90.5	**60.0**	92.7	**76.1**	**55.9**	**92.7**	**72.7**	**81.9**	**66.0**	59.7	**71.8**	**71.9**

Table 6: Experimental results of models with the ASPP output layer but no MS-COCO pre-training on Cityscapes *val* set.

Models	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	mIoU
DeepLabv2	97.0	77.9	89.4	44.6	48.6	48.7	54.1	66.7	**90.3**	**58.0**	**92.5**	73.9	51.9	91.6	**59.9**	75.5	60.5	56.3	69.6	68.8
G Interact	**97.1**	**78.7**	**89.6**	44.7	49.2	48.6	54.2	**67.0**	**90.3**	57.6	92.1	**74.3**	**52.2**	91.7	59.0	**77.1**	60.5	56.8	**70.1**	69.0
SS Conv	97.0	78.3	**89.6**	**45.2**	**49.4**	**48.9**	**54.6**	66.5	90.2	57.1	92.0	74.1	52.1	**91.8**	59.5	76.8	**63.5**	**58.8**	69.7	**69.2**

future work. The current study focuses on degridding in the 2-D cases, but the methods are generic and can be applied to other settings. We will explore their applications in the 1-D case in the context of text analysis.

ACKNOWLEDGMENTS

This work was supported in part by National Science Foundation grant IIS-1633359 and Defense Advanced Research Projects Agency grant N66001-17-2-4031.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* (2016).

[2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. 2016. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915* (2016).

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

[4] François Chollet. 2016. Xception: Deep Learning with Depthwise Separable Convolutions. *arXiv preprint arXiv:1610.02357* (2016).

[5] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. 2016. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3213–3223.

[6] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. 2016. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*. 379–387.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 248–255.

[8] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. 2010. The pascal visual object classes (voc) challenge. *International journal of computer vision* 88, 2 (2010), 303–338.

[9] Hongyang Gao, Hao Yuan, Zhengyang Wang, and Shuiwang Ji. 2017. Pixel Deconvolutional Networks. *arXiv preprint arXiv:1705.06820* (2017).

[10] Alessandro Giusti, Dan C Ciresan, Jonathan Masci, Luca M Gambardella, and Jurgen Schmidhuber. 2013. Fast image scanning with deep max-pooling convolutional neural networks. In *Image Processing (ICIP), 2013 20th IEEE International Conference on*. IEEE, 4034–4038.

[11] Ryuhei Hamaguchi, Aito Fujita, Keisuke Nemoto, Tomoyuki Imaizumi, and Shuhei Hikosaka. 2017. Effective Use of Dilated Convolutions for Segmenting Small Object Instances in Remote Sensing Imagery. *arXiv preprint arXiv:1709.00179* (2017).

[12] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. 2011. Semantic contours from inverse detectors. In *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 991–998.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[14] Matthias Holschneider, Richard Kronland-Martinet, Jean Morlet, and Ph Tchamitchian. 1990. A real-time algorithm for signal analysis with the help of the wavelet transform. In *Wavelets*. Springer, 286–297.

[15] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. 2016. Speed/accuracy trade-offs for modern convolutional object detectors. *arXiv preprint arXiv:1611.10012* (2016).

[16] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099* (2016).

[17] Nal Kalchbrenner, Aaron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. 2016. Video pixel networks. *arXiv preprint arXiv:1610.00527* (2016).

[18] Hongsheng Li, Rui Zhao, and Xiaogang Wang. 2014. Highly efficient forward and backward propagation of convolutional neural networks for pixelwise classification. *arXiv preprint arXiv:1412.4526* (2014).

[19] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 740–755.

[20] Wei Liu, Andrew Rabinovich, and Alexander C Berg. 2015. Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579* (2015).

[21] Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3431–3440.

[22] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. 2016. Understanding the effective receptive field in deep convolutional neural networks. In *Advances in Neural Information Processing Systems*. 4898–4906.

[23] Franck Mamalet and Christophe Garcia. 2012. Simplifying convnets for fast learning. *Artificial Neural Networks and Machine Learning–ICANN 2012* (2012), 58–65.

[24] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499* (2016).

[25] George Papandreou, Iasonas Kokkinos, and Pierre-André Savalle. 2015. Modeling local and global deformations in deep learning: Epitomic convolution, multiple instance learning, and sliding window detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 390–399.

[26] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. 2013. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229* (2013).

[27] Mark J Shensa. 1992. The discrete wavelet transform: wedding the a trous and Mallat algorithms. *IEEE Transactions on signal processing* 40, 10 (1992), 2464–2482.

[28] Panqu Wang, Pengfei Chen, Ye Yuan, Ding Liu, Zehua Huang, Xiaodi Hou, and Garrison Cottrell. 2017. Understanding convolution for semantic segmentation. *arXiv preprint arXiv:1702.08502* (2017).

[29] Fisher Yu and Vladlen Koltun. 2015. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122* (2015).

[30] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. 2017. Dilated residual networks. *arXiv preprint arXiv:1705.09914* (2017).

[31] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. 2016. Pyramid scene parsing network. *arXiv preprint arXiv:1612.01105* (2016).