Depth-Aware CNN for RGB-D Segmentation

Weiyue Wang(✉) and Ulrich Neumann

University of Southern California, Los Angeles, CA, USA
{weiyuewa,uneumann}@usc.edu

Abstract. Convolutional neural networks (CNN) are limited by the lack of capability to handle geometric information due to the fixed grid kernel structure. The availability of depth data enables progress in RGB-D semantic segmentation with CNNs. State-of-the-art methods either use depth as additional images or process spatial information in 3D volumes or point clouds. These methods suffer from high computation and memory cost. To address these issues, we present Depth-aware CNN by introducing two intuitive, flexible and effective operations: depth-aware convolution and depth-aware average pooling. By leveraging depth similarity between pixels in the process of information propagation, geometry is seamlessly incorporated into CNN. Without introducing any additional parameters, both operators can be easily integrated into existing CNNs. Extensive experiments and ablation studies on challenging RGB-D semantic segmentation benchmarks validate the effectiveness and flexibility of our approach.

Keywords: Geometry in CNN · RGB-D semantic segmentation

1 Introduction

Recent advances [4,28,36] in CNN have achieved significant success in scene understanding. With the help of range sensors (such as Kinect, LiDAR etc.), depth images are applicable along with RGB images. Taking advantages of the two complementary modalities with CNN is able to improve the performance of scene understanding. However, CNN is limited to model geometric variance due to the fixed grid computation structure. Incorporating the geometric information from depth images into CNN is important yet challenging.

Extensive studies [5,6,17,22,26,27,34] have been carried out on this task. FCN [28] and its successors treat depth as another input image and construct two CNNs to process RGB and depth separately. This doubles the number of network parameters and computation cost. In addition, the two-stream network architecture still suffers from the fixed geometric structures of CNN. Even if

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-01252-6_9) contains supplementary material, which is available to authorized users.

the geometric relations of two pixels are given, this relation cannot be used in information propagation of CNN. An alternative is to leverage 3D networks [26, 31, 33] to handle geometry. Nevertheless, both volumetric CNNs [31] and 3D point cloud graph networks [26] are computationally more expensive than 2D CNN. Despite the encouraging results of these progresses, we need to seek a more flexible and efficient way to exploit 3D geometric information in 2D CNN.

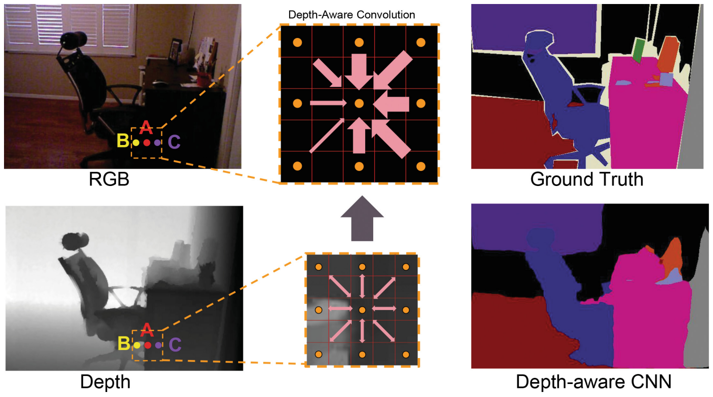


Fig. 1. Illustration of Depth-aware CNN. A and C are labeled as table and B is labeled as chair. They all have similar visual features in the RGB image, while they are separable in depth. Depth-aware CNN incorporate the geometric relations of pixels in both convolution and pooling. When A is the center of the receptive field, C then has more contribution to the output unit than B. Figures in the rightmost column shows the RGB-D semantic segmentation result of Depth-aware CNN.

To address the aforementioned problems, in this paper, we present an end-to-end network, Depth-aware CNN (D-CNN), for RGB-D segmentation. Two new operators are introduced: *depth-aware convolution* and *depth-aware average pooling*. *Depth-aware convolution* augments the standard convolution with a depth similarity term. We force pixels with similar depths with the center of the kernel to have more contribution to the output than others. This simple depth similarity term efficiently incorporates geometry in a convolution kernel and helps build a depth-aware receptive field, where convolution is not constrained to the fixed grid geometric structure.

The second introduced operator is *depth-ware average pooling*. Similarly, when a filter is applied on a local region of the feature map, the pairwise relations in depth between neighboring pixels are considered in computing mean of the local region. Visual features are able to propagate along with the geometric structure given in depth images. Such geometry-aware operation enables the localization of object boundaries with depth images.

Both operators are based on the intuition that pixels with the same semantic label and similar depths should have more impact on each other. We observe that two pixels with the same semantic labels have similar depths. As illustrated in Fig. 1, pixel A and pixel C should be more correlated with each other than pixel A and pixel B. This correlation difference is obvious in depth image while it is not captured in RGB image. By encoding the depth correlation in CNN, pixel C has more contribution to the output unit than pixel B in the process of information propagation.

The main advantages of depth-aware CNN are summarized as follows:

– By exploiting the nature of CNN kernel handling spatial information, geometry in depth image is able to be integrated into CNN seamlessly.
– Depth-aware CNN does not introduce any parameters and computation complexity to the conventional CNN.
– Both *depth-aware convolution* and *depth-ware average pooling* can replace their standard counterparts in conventional CNNs with minimal cost.

Depth-aware CNN is a general framework that bonds 2D CNN and 3D geometry. Comparison with the state-of-the-art methods and extensive ablation studies on RGB-D semantic segmentation illustrate the flexibility, efficiency and effectiveness of our approach.

2 Related Works

2.1 RGB-D Semantic Segmentation

With the help of CNNs, semantic segmentation on 2D images have achieved promising results [4,14,28,36]. These advances in 2D CNN and the availability of depth sensors enables progresses in RGB-D segmentation. Compared to the RGB settings, RGB-D segmentation is able to integrate geometry into scene understanding. In [8,10,21,32], depth is simply treated as additional channels and directly fed into CNN. Several works [9,10,18,23,28] encode depth to HHA image, which has three channels: horizontal disparity, height above ground, and norm angle. RGB image and HHA image are fed into two separate networks, and the two predictions are summed up in the last layer. The two-stream network doubles the number of parameters and forward time compared to the conventional 2D network. Moreover, CNNs per se are limited in their ability to model geometric transformations due to their fixed grid computation. Cheng et al. [5] propose a locality-sensitive deconvolution network with gated fusion. They build a feature affinity matrix to perform weighted average pooling and unpooling. Lin et al. [19] discretize depth and build different branches for different discrete depth value. He et al. [12] use spatio-temporal correspondences across frames to aggregate information over space and time. This requires heavy pre and post-processing such as optical flow and superpixel computation.

Alternatively, many works [30,31] attempt to solve the problem with 3D CNNs. However, the volumetric representation prevents scaling up due to high memory and computation cost. Recently, deep learning frameworks [13,24–26,35] on point cloud are introduced to address the limitations of 3D volume.

Qi et al. [26] built a 3D k-nearest neighbor (kNN) graph neural network on a point cloud with extracted features from a CNN and achieved the state-of-the-art on RGB-D segmentation. Although their method is more efficient than 3D CNNs, the kNN operation suffers from high computation complexity and lack of flexibility. Instead of using 3D representations, we use the raw depth input and integrate 3D geometry into 2D CNN in a more efficient and flexible fashion.

2.2 Spatial Transformations in CNN

Standard CNNs are limited to model geometric transformations due to the fixed structure of convolution kernels. Recently, many works are focused on dealing with this issue. Dilated convolutions [4,36] increases the receptive field size with keeping the same complexity in parameters. This operator achieves better performance on vision tasks such as semantic segmentation. Spatial transform networks [15] warps feature maps with a learned global spatial transformation. Deformable CNN [7] learns kernel offsets to augment the spatial sampling locations. These methods have shown geometric transformations enable performance boost on different vision tasks.

With the advances in 3D sensors, depth is applicable at low cost. The geometric information that resides in depth is highly correlated with the spatial transformation in CNN. Bilateral filters [2,3] is widely used in computer graphics for image smoothness with edge preserving. They use a Gaussian term to weight neighboring pixels. Similarly as bilateral filter, our method integrates the geometric relation of pixels into basic operations of CNN, i.e. convolution and pooling, where we use a weighted kernel and force every neuron to have different contributions to the output. This weighted kernel is defined by depth and is able to incorporate geometric relationships without introducing any parameter.

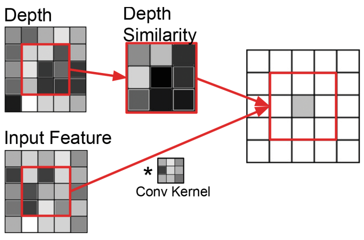
3 Depth-Aware CNN

In this section, we introduce two depth-aware operations: depth-aware convolution and depth-aware average pooling. They are both simple and intuitive. Both operations require two inputs: the input feature map $\mathbf{x} \in \mathbb{R}^{c_i \times h \times w}$ and the depth image $\mathbf{D} \in \mathbb{R}^{h \times w}$, where c_i is the number of input feature channels, h is the height and w is the width. The output feature map is denoted as $\mathbf{y} \in \mathbb{R}^{c_o \times h \times w}$, where c_o is the number of output feature channels. Although $\mathbf{x}$ and $\mathbf{y}$ are both 3D tensors, the operations are explained in 2D spatial domain for notation clarity and they remain the same across different channels.

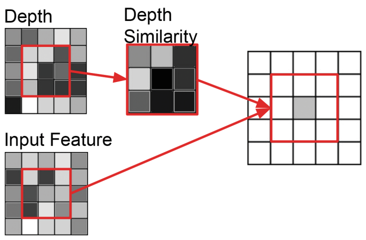
3.1 Depth-Aware Convolution

A standard 2D convolution operation is the weighted sum of a local grid. For each pixel location $\boldsymbol{p_0}$ on $\mathbf{y}$, the output of standard 2D convolution is

$$\mathbf{y}(\mathbf{p}_0) = \sum_{\mathbf{p}_n \in \mathcal{R}} \mathbf{w}(\mathbf{p}_n) \cdot \mathbf{x}(\mathbf{p}_0 + \mathbf{p}_n), \tag{1}$$



(a) Depth-aware Convolution



(b) Depth-aware Average Pooling

Fig. 2. Illustration of information propagation in Depth-aware CNN. Without loss of generality, we only show one filter window with kernel size 3 × 3. In depth similarity shown in figure, darker color indicates higher similarity while lighter color represents that two pixels are less similar in depth. In (a), the output activation of depth-aware convolution is the multiplication of depth similarity window and the convolved window on input feature map. Similarly in (b), the output of depth-aware average pooling is the average value of the input window weighted by the depth similarity. (Color figure online)

where $\mathcal{R}$ is the local grid around $\boldsymbol{p_0}$ in $\mathbf{x}$ and $\mathbf{w}$ is the convolution kernel. $\mathcal{R}$ can be a regular grid defined by kernel size and dilation [36], and it can also be a non-regular grid [7].

As is shown in Fig. 1, pixel A and pixel B have different semantic labels and different depths while they are not separable in RGB space. On the other hand, pixel A and pixel C have the same labels and similar depths. To exploit the depth correlation between pixels, depth-aware convolution simply adds a depth similarity term, resulting in two sets of weights in convolution: (1) the learnable convolution kernel $\mathbf{w}$; (2) depth similarity $F_{\mathbf{D}}$ between two pixels. Consequently, Eq. 1 becomes

$$\mathbf{y}(\mathbf{p}_0) = \sum_{\mathbf{p}_n \in \mathcal{R}} \mathbf{w}(\mathbf{p}_n) \cdot F_{\mathbf{D}}(\mathbf{p}_0, \mathbf{p}_0 + \mathbf{p}_n) \cdot \mathbf{x}(\mathbf{p}_0 + \mathbf{p}_n). \tag{2}$$

And $F_{\mathbf{D}}(\mathbf{p}_i, \mathbf{p}_j)$ is defined as

$$F_{\mathbf{D}}(\mathbf{p}_i, \mathbf{p}_j) = \exp(-\alpha|\mathbf{D}(\mathbf{p}_i) - \mathbf{D}(\mathbf{p}_j)|), \tag{3}$$

where α is a constant. The selection of $F_{\mathbf{D}}$ is based on the intuition that pixels with similar depths should have more impact on each other. We will study the effect of different α and different $F_{\mathbf{D}}$ in Sect. 4.2.

The gradients for $\mathbf{x}$ and $\mathbf{w}$ are simply multiplied by $F_{\mathbf{D}}$. Note that the $F_{\mathbf{D}}$ part does not require gradient during back-propagation, therefore, Eq. 2 does not integrate any parameters by the depth similarity term.

Figure 2(a) illustrates this process. Pixels which have similar depths with the convolving center will have more impact on the output during convolution.

3.2 Depth-Aware Average Pooling

The conventional average pooling computes the mean of a grid $\mathcal{R}$ over $\mathbf{x}$. It is defined as

$$\mathbf{y}(\mathbf{p}_0) = \frac{1}{|\mathcal{R}|} \sum_{\mathbf{p}_n \in \mathcal{R}} \mathbf{x}(\mathbf{p}_0 + \mathbf{p}_n). \tag{4}$$

It treats every pixel equally and will make the object boundary blurry. Geometric information is useful to address this issue.

Similar to as in depth-aware convolution, we take advantage of the depth similarity $F_{\mathbf{D}}$ to force pixels with more consistent geometry to make more contribution on the corresponding output. For each pixel location $\boldsymbol{p_0}$, the depth-aware average pooling operation then becomes

$$\mathbf{y}(\mathbf{p}_0) = \frac{1}{\sum_{\mathbf{p}_n \in \mathcal{R}} F_{\mathbf{D}}(\mathbf{p}_0, \mathbf{p}_0 + \mathbf{p}_n)} \sum_{\mathbf{p}_n \in \mathcal{R}} F_{\mathbf{D}}(\mathbf{p}_0, \mathbf{p}_0 + \mathbf{p}_n) \cdot \mathbf{x}(\mathbf{p}_0 + \mathbf{p}_n). \tag{5}$$

The gradient should be multiplied by $\frac{F_{\mathbf{D}}}{\sum_{\mathbf{p}_n \in \mathcal{R}} F_{\mathbf{D}}(\mathbf{p}_0, \mathbf{p}_0 + \mathbf{p}_n)}$ during back propagation. As illustrated in Fig. 2(b), this operation prevent suffering from the fixed geometric structure of standard pooling.

3.3 Understanding Depth-Aware CNN

A major advantage of CNN is its capability of using GPU to perform parallel computing and accelerate the computation. This acceleration mainly stems from unrolling convolution operation with the grid computation structure. However, this limits the ability of CNN to model geometric variations. Researchers in 3D deep learning have focused on modeling geometry in deep neural networks in the last few years. As the volumetric representation [30,31] is of high memory and computation cost, point clouds are considered as a more proper representation. However, deep learning frameworks [25,26] on point cloud are based on building kNN. This not only suffers from high computation complexity, but also breaks the pixel-wise correspondence between RGB and depth, which makes the framework is not able to leverage the efficiency of CNN's grid computation structure. Instead of operating on 3D data, we exploit the raw depth input. By augmenting the convolution kernel with a depth similarity term, depth-aware CNN captures geometry with transformable receptive field.

Many works have studied spatial transformable receptive field of CNN. Dilated convolution [4,36] has demonstrated that increasing receptive field boost the performance of networks. In deformable CNN [7], Dai et al. demonstrate that learning receptive field adaptively can help CNN achieve better results. They also show that pixels within the same object in a receptive field contribute more to the output unit than pixels with different labels. We observe that semantic labels and depths have high correlations. Table 1 reports the statistics of pixel depth variance within the same class and across different classes on NYUv2 [22] dataset. Even the pixel depth variances of large objects such as wall and floor are much smaller than the variance of a whole scene. This indicates that pixels with the same semantic labels tend to have similar depths. This pattern is integrated in Eqs. 2 and 5 with $F_{\mathbf{D}}$. Without introducing any parameter, depth-aware convolution and depth-aware average pooling are able to enhance the localization ability of CNN. We evaluate the impact on performance of different depth similarity functions $F_{\mathbf{D}}$ in Sect. 4.2.

Table 1. Mean depth variance of different categories on NYUv2 dataset. "All" denotes the mean variance of all categories. For every image, pixel-wise variance of depth for each category is calculated. Averaged variance is then computed over all images. For "All", all pixels in a image are considered to calculate the depth variance. Mean variance over all images is further computed.

	Wall	Floor	Bed	Chair	Table	All
Variance	0.57	0.65	0.12	0.23	0.34	1.20

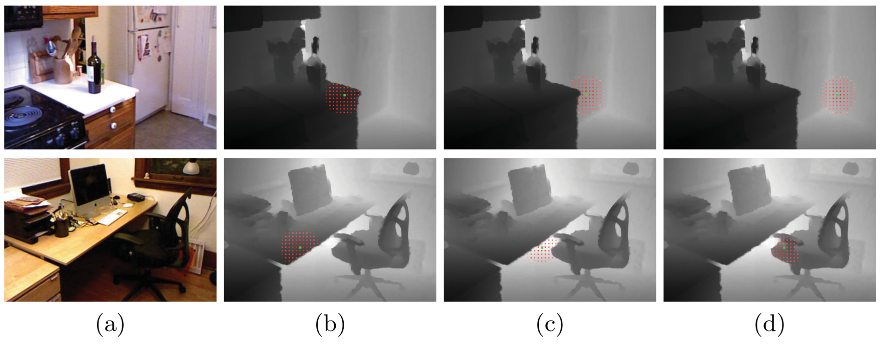


Fig. 3. Illustration of effective receptive field of Depth-aware CNN. (a) is the input RGB images. (b), (c) and (d) are depth images. For (b), (c) and (d), we show the sampling locations (red dots) in three levels of 3×3 depth-aware convolutions for the activation unit (green dot). (Color figure online)

To get a better understanding of how depth-aware CNN captures geometry with depth, Fig. 3 shows the effective receptive field of the given input neuron. In conventional CNN, the receptive fields and sampling locations are fixed across feature map. With the depth-aware term incorporated, they are adjusted by the geometric variance. For example, in the second row of Fig. 3(d), the green point is labeled as chair and the effective receptive field of the green point are essentially chair points. This indicates that the effective receptive field mostly have the same semantic label as the center. This pattern increases CNN's performance on semantic segmentation.

3.4 Depth-Aware CNN for RGB-D Semantic Segmentation

In this paper, we focus on RGB-D semantic segmentation with depth-aware CNN. Given an RGB image along with depth, our goal is to produce a semantic mask indicating the label of each pixel. Both depth-aware convolution and average pooling easily replace their counterpart in standard CNN.

Table 2. Network architecture. DeepLab is our baseline with a modified version of VGG-16 as the encoder. The convolution layer parameters are denoted as "C[kernel size]-[number of channels]-[dilation]". "DC" and "Davgpool" represent depth-aware convolution and depth-aware average pooling respectively.

Layer name	conv1_x	conv2_x	conv3_x	conv4_x	conv5_x	conv6 & conv7
Baseline DeepLab	C3-64-1 C3-64-1 maxpool	C3-128-1 C3-128-1 Maxpool	C3-256-1 C3-256-1 C3-256-1 maxpool	C3-512-1 C3-512-1 C3-512-1 maxpool	C3-512-2 C3-512-2 C3-512-2 Avgpool	C3-1024-12 C1-1024-0 globalpool+concat
D-CNN	DC3-64-1 C3-64-1 maxpool	DC3-128-1 C3-128-1 maxpool	DC3-256-1 C3-256-1 C3-256-1 maxpool	DC3-512-1 C3-512-1 C3-512-1 maxpool	DC3-512-2 C3-512-2 C3-512-2 Davgpool	DC3-1024-12 C1-1024-0 globalpool+concat

DeepLab [4] is a state-of-the-art method for semantic segmentation. We adopt DeepLab as our baseline for semantic segmentation and a modified VGG-16 network is used as the encoder. We replace layers in this network with depth-aware operations. The network configurations of the baseline and depth-aware CNN are outlined in Table 2. Suppose *conv*7 has C channels. Following [26], global pooling is used to compute a C-dim vector from *conv*7. This vector is then appended to all spatial positions and results in a $2C$-channel feature map. This feature map is followed by a 1×1 conv layer and produce the segmentation probability map.

4 Experiments

Evaluation is performed on three popular RGB-D datasets:

– NYUv2 [22]: NYUv2 contains of 1, 449 RGB-D images with pixel-wise labels. We follow the 40-class settings and the standard split with 795 training images and 654 testing images.
– SUN-RGBD [16,29]: This dataset have 37 categories of objects and consists of 10, 335 RGB-D images, with 5, 285 as training and 5050 as testing.

– Stanford Indoor Dataset (SID) [1]: SID contains $70,496$ RGB-D images with 13 object categories. We use Area $1, 2, 3, 4$ and 6 as training, and Area 5 as testing.

Four common metrics are used for evaluation: pixel accuracy (Acc), mean pixel accuracy of different categories (mAcc), mean Intersection-over-Union of different categories (mIoU), and frequency-weighted IoU (fwIoU). Suppose n_{ij} is the number of pixels with ground truth class i and predicted as class j, n_C is the number of classes and s_i is the number of pixels with ground truth class i, the total number of all pixels is $s = \sum_i s_i$. The four metrics are defined as follows: Acc $= \sum_i \frac{n_{ii}}{s}$, mAcc $= \frac{1}{n_C} \sum_i \frac{n_{ii}}{s_i}$, mIoU $= \frac{1}{n_C} \sum_i \frac{n_{ii}}{s_i + \sum_j n_{ji} - n_{ii}}$, fwIoU $= \frac{1}{s} \sum_i s_i \frac{n_{ii}}{s_i + \sum_j n_{ji} - n_{ii}}$.
Implementation Details. For most experiments, DeepLab with a modified VGG-16 encoder (c.f. Table 2) is the baseline. Depth-aware CNN based on DeepLab outlined in Table 2 is evaluated to validate the effectiveness of our approach and this is referred as "D-CNN" in the paper. We also conduct experiments with combining HHA encoding [9]. Following [8,26,28], two baseline networks consume RGB and HHA images separately and the predictions of both networks are summed up in the last layer. This two-stream network is dubbed as "HHA". To make fair comparison, we also build depth-aware CNN with this two-stream fashion and denote this as "D-CNN+HHA". In ablation study, we further replace VGG-16 with ResNet-50 [11] as the encoder to have a better understanding of the functionality of depth-aware operations.

We use SGD optimizer with initial learning rate 0.001, momentum 0.9 and batch size 1. The learning rate is multiplied by $(1 - \frac{iter}{max\_iter})^{0.9}$ for every 10 iterarions. α is set to 8.3. (The impact of α is studied in Sect. 4.2.) The dataset is augmented by randomly scaling, cropping, and color jittering. We use PyTorch deep learning framework. Both depth-aware convolution and depth-aware average pooling operators are implemented with CUDA acceleration. Code is available at github.com/laughtervv/DepthAwareCNN.

4.1 Main Results

Depth-aware CNN is compared with both its baseline and the state-of-the-art methods on NYUv2 and SUN-RGBD dataset. It is also compared with the baseline on SID dataset.
NYUv2. Table 3 shows quantitative comparison results between D-CNNs and baseline models. Since D-CNN and its baseline are in different function space, all networks are trained from scratch to make fair comparison in this experiment. Without introducing any parameters, D-CNN outperforms the baseline by incorporating geometric information in convolution operation. Moreover, the performance of D-CNN also exceeds "HHA" network by using only half of its parameters. This effectively validate the superior capability of D-CNN on handling geometry over "HHA".

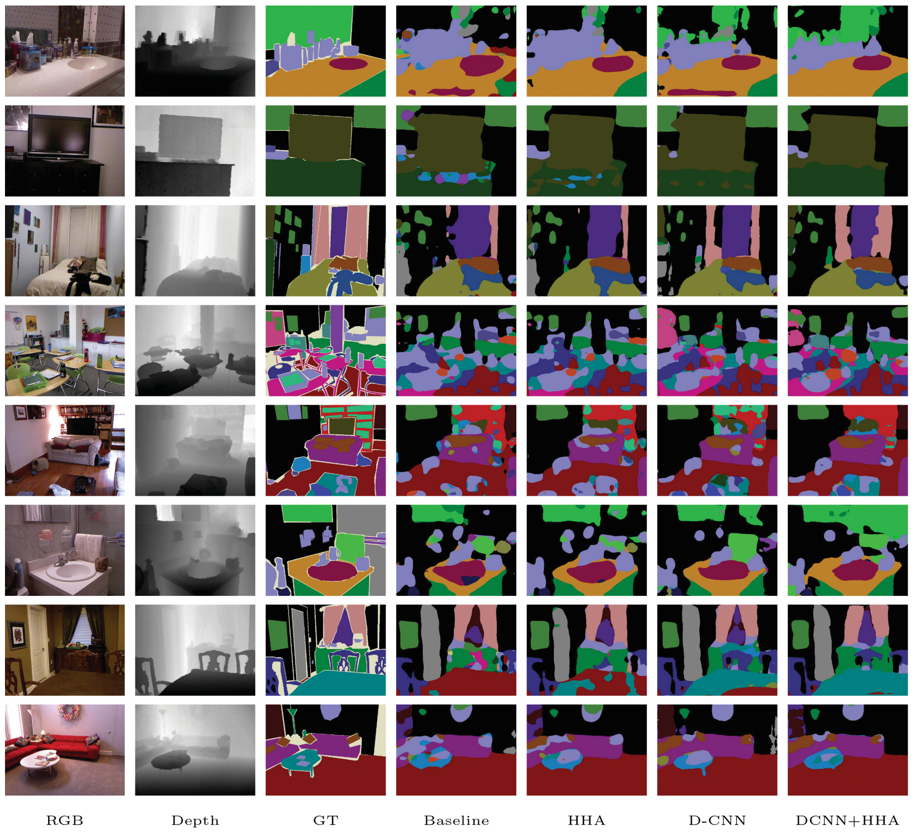


Fig. 4. Segmentation results on NYUv2 test dataset. "GT" denotes ground truth. The white regions in "GT" are the ignoring category. Networks are trained from pre-trained models.

Table 3. Comparison with baseline CNNs on NYUv2 test set. Networks are trained from scratch.

	Baseline	HHA	D-CNN	D-CNN+HHA
Acc (%)	50.1	59.1	60.3	**61.4**
mAcc (%)	23.9	30.8	**39.3**	35.6
mIoU (%)	15.9	21.9	**27.8**	26.2
fwIoU (%)	34.2	43.0	44.9	**45.7**

Table 4. Comparison with the state-of-the-arts on NYUv2 test set. Networks are trained from pre-trained models.

	[28]	[8]	[12]	[26]	HHA	D-CNN	D-CNN+HHA	DM-CNN+HHA	[20]	D-ResNet-152
mAcc (%)	46.1	45.1	53.8	55.2	51.1	53.6	56.3	**58.4**	58.9	**61.1**
mIoU (%)	34.0	34.1	40.1	42.0	40.4	41.0	43.9	**44.7**	46.5	**48.4**

We also compare our results with the state-of-the-art methods. Table 4 illustrates the good performance of D-CNN. In this experiment, the networks are initialized with the pre-trained parameters in [4]. Long et al. [28] and Eigen et al. [8] both use the two-stream network with HHA/depth encoding. Yang et al. [12] compute optical flows and superpixels to augment the performance with spatial-temporal information. D-CNN with only one VGG network is superior to their methods. Qi et al. [26] built a 3D graph on the top of VGG encoder and use RNN to update the graph, which introduces more network parameters and higher computation complexity. By replacing max-pooling layers in `Conv1`, `Conv2`, `Conv3` as depth-aware max-pooling (defined as $\mathbf{y}(\mathbf{p}_0) = \max_{\mathbf{p}_n \in \mathcal{R}} F_{\mathbf{D}}(\mathbf{p}_0, \mathbf{p}_0 + \mathbf{p}_n) \cdot \mathbf{x}(\mathbf{p}_0 + \mathbf{p}_n))$, we can get further performance improvement, and this experiment is referred as DM-CNN-HHA in Table 4. We also replace the baseline VGG with ResNet-152 (pre-trained with [20]) and compare with its baseline [20] in Table 4. As is shown in Table 4, D-CNN is already comparable with these state-of-the-art methods. By incorporating HHA encoding, our method achieves the state-of-the-art on this dataset. Figure 4 visualizes qualitative comparison results on NYUv2 test set.

SUN-RGBD. The comparison results between D-CNN and its baseline are listed in Table 5. The networks in this table are trained from scratch. D-CNN outperforms baseline by a large margin. Substituting the baseline with the two-stream "HHA" network is able to further improve the performance.

By comparing with the state-of-the-art methods in Table 6, we can further see the effectiveness of D-CNN. Similarly as in NYUv2, the networks are initialized with pre-trained model in this experiment. Figure 5 illustrates the qualitative comparison results on SUN-RGBD test set. Our network achieves comparable performance with the state-of-the-art method [26], while their method is more time-consuming. We will further compare the runtime and numbers of model parameters in Sect. 4.3.

Table 5. Comparison with baseline CNNs on SUN-RGBD test set. Networks are trained from scratch.

	Baseline	HHA	D-CNN	D-CNN+HHA
Acc (%)	66.6	72.6	72.4	**72.9**
mAcc (%)	31.5	37.9	38.6	**41.2**
mIoU (%)	22.8	28.8	29.7	**31.3**
fwIoU (%)	51.4	58.5	58.2	**59.3**

Table 6. Comparison with the state-of-the-arts on SUN-RGBD test set. Networks are trained from pre-trained models.

	[18]	[26]	HHA	D-CNN	D-CNN+HHA
mAcc (%)	48.1	**55.2**	50.5	51.2	53.5
mIoU (%)	-	**42.0**	40.2	41.5	**42.0**

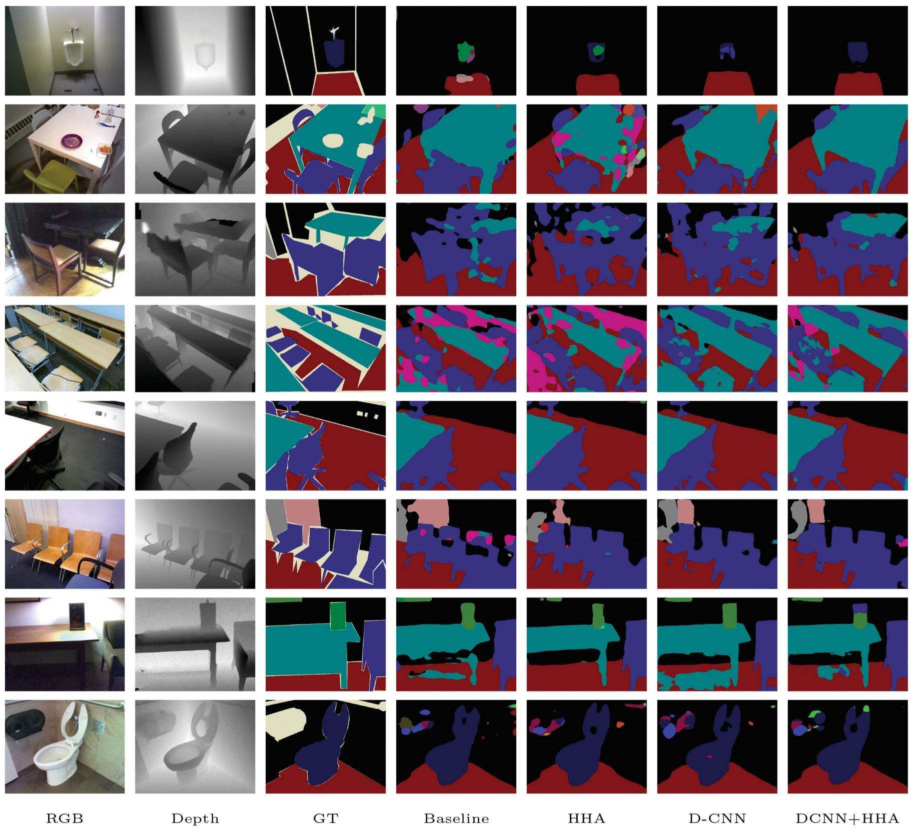


Fig. 5. Segmentation results on SUN-RGBD test dataset. "GT" denotes ground truth. The white regions in "GT" are the ignoring category. Networks are trained from pre-trained models.

SID. The comparison results on SID between D-CNN with its baseline are reported in Table 7. Networks are trained from scratch. Using depth images, D-CNN is able to achieve 4% IoU over CNN while preserving the same number of parameters and computation complexity.

Table 7. Comparison with baseline CNNs on SID Area 5. Networks are trained from scratch.

	Baseline	D-CNN
Acc (%)	64.3	**65.4**
mAcc (%)	46.7	**55.5**
mIoU (%)	35.5	**39.5**
fwIoU (%)	48.5	**49.9**

4.2 Ablation Study

In this section, we conduct ablation studies on NYUv2 dataset to validate efficiency and efficacy of our approach. Testing results on NYUv2 test set are reported.

Depth-Aware CNN. To verify the functionality of both depth-aware convolution and depth-aware average pooling, the following experiments are conducted.

- VGG-1: `Conv1_1`, `Conv2_1`, `Conv3_1`, `Conv4_1`, `Conv5_1` and `Conv6` in VGG-16 are replaced with depth-aware convolution. This is the same as in Table 2.
- VGG-2: `Conv4_1`, `Conv5_1` and `Conv6` in VGG-16 are replaced with depth-aware convolution. Other layers remain the same as in Table 2.
- VGG-3: The depth-aware average pooling layer listed in Table 2 is replaced with regular pooling. Other layers remain the same as in Table 2.
- VGG-4: Only `Conv1_1`, `Conv2_1`, `Conv3_1` are replaced with depth-aware convolution.

Results are shown in Table 8. Compared to VGG-2, VGG-1 adds depth-aware convolution in bottom layers. This helps the network propagate more fine-grained features with geometric relationships and increase segmentation performance by 6% in IoU. VGG-1 also outperforms VGG-4. Top layers conv4, 5 have more contextual information, and applying D-CNN on these layers still benefits the prediction. As is shown in [24], not all contextual information is useful. D-CNN helps to capture more effective contextual information. The depth-aware average pooling operation is able to further promote the accuracy.

Table 8. Results of using depth-aware operations in different layers. Experiments are conducted on NYUv2 test set. Networks are trained from scratch.

	Baseline	HHA	VGG-1	VGG-2	VGG-3	VGG-4
Acc (%)	50.1	59.1	**60.3**	56.0	59.3	59.5
mAcc (%)	23.9	30.8	**39.3**	32.2	39.2	37.3
mIoU (%)	15.9	21.9	**27.8**	22.4	27.4	26.6
fwIoU (%)	34.2	43.0	**44.9**	40.2	44.0	43.8

Table 9. Results of using depth-aware operations in ResNet-50. Networks are trained from pre-trained models.

	VGG-1	ResNet-50	D-ResNet-50
Acc (%)	69.4	68.9	**69.6**
mAcc (%)	**53.6**	50.2	53.3
mIoU (%)	41.0	38.8	**41.5**
fwIoU (%)	**54.5**	54.4	54.4

We also replace VGG-16 to ResNet as the encoder. We test depth-aware operations on ResNet. The `Conv3_1`, `Conv4_1`, and `Conv5_1` in ResNet-50 are replaced with depth-aware convolution. ResNet-50 is initialized with parameters pre-trained on ADE20K [37]. Detailed architecture and training details for ResNet can be found in Supplementary Materials. Results are listed in Table 9.

Depth Similarity Function. We modify α and $F_{\mathbf{D}}$ to further validate the effect of different choices of depth similarity function on performance. We conduct the following experiments:

- $\alpha_{8.3}$: α is set to 8.3. The network architecture is the same as Table 2.
- α_{20}: α is set to 20. The network architecture is the same as Table 2.
- $\alpha_{2.5}$: α is set to 2.5. The network architecture is the same as Table 2.
- clip$F_{\mathbf{D}}$: The network architecture is the same as Table 2. $F_{\mathbf{D}}$ is defined as

$$F_{\mathbf{D}}(\mathbf{p}_i, \mathbf{p}_j) = \begin{cases} 0, & |\mathbf{D}(\mathbf{p}_i) - \mathbf{D}(\mathbf{p}_j)| \geq 1 \\ 1, & \text{otherwise} \end{cases} \tag{6}$$

Table 10 reports the test performances with different depth similarity functions. Though the performance varies with different α, they are all superior to baseline and even "HHA". The result of clip$F_{\mathbf{D}}$ is also comparable with "HHA". This validate that the effectiveness of using a depth-sensitive term to weight the contributions of neurons.

Table 10. Results of using different α and $F_{\mathbf{D}}$. Experiments are conducted on NYUv2 test set. Networks are trained from scratch.

	Baseline	HHA	$\alpha_{8.3}$	α_{20}	$\alpha_{2.5}$	clip$F_{\mathbf{D}}$
Acc (%)	50.1	59.1	**60.3**	58.5	58.5	53.0
mAcc (%)	23.9	30.8	**39.3**	35.2	35.9	29.8
mIoU (%)	15.9	21.9	**27.8**	24.9	25.3	20.1
fwIoU (%)	34.2	43.0	**44.9**	42.6	42.9	37.5

Performance Analysis. To have a better understanding of how depth-aware CNN outperforms the baseline, we visualize the improvement of IoU for each semantic class in Fig. 6(a). The statics shows that D-CNN outperform baseline on most object categories, especially these large objects such as ceilings and curtain. Moreover, we observe depth-aware CNN has a faster convergence than baseline, especially trained from scratch. Figure 6(b) shows the training loss evolution with respect to training steps. Our network gains lower loss values than baseline. Depth similarity helps preserving edge details, however, when depth values vary in a single object, depth-aware CNN may lose contextual information. Some failure cases can be found in supplemental material.

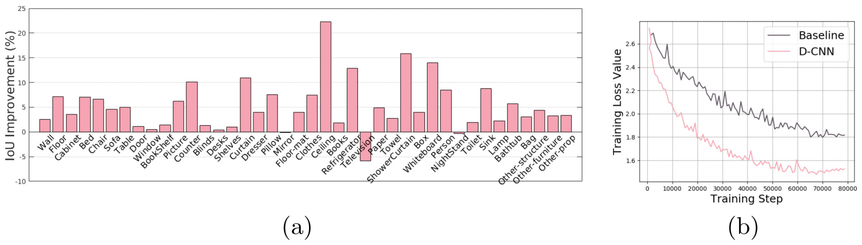


Fig. 6. Performance Analysis. (a) Per-class IoU improvement of D-CNN over baseline on NYUv2 test dataset. (b) Evolution of training loss on NYUv2 train dataset. Networks are trained from scratch.

4.3 Model Complexity and Runtime Analysis

Table 11 reports the model complexity and runtime of D-CNN and the state-of-the-art method [26]. In their method, kNN takes $O(kN)$ runtime at least, where N is the number of pixels. We leverage the grid structure of raw depth input. As is shown in Table 11, depth-aware operations do not incorporate any new parameters. The network forward time is only slightly greater than its baseline. Without increasing any model parameters, D-CNN is able to incorporate geometric information in CNN efficiently.

Table 11. Model complexity and runtime comparison. Runtime is tested on Nvidia 1080Ti, with input image size $425 \times 560 \times 3$.

	Baseline	HHA	[26]	D-CNN	D-CNN-HHA
net. forward (ms)	32.5	64.2	214	39.3	79.7
# of params	47.0M	92.0M	47.25M	47.0M	92.0M

5 Conclusion

We present a novel depth-aware CNN by introducing two operations: depth-aware convolution and depth-aware average pooling. Depth-aware CNN augments conventional CNN with a depth similarity term and encode geometric variance into basic convolution and pooling operations. By adapting effective receptive field, these depth-aware operations are able to incorporate geometry into CNN while preserving CNN's efficiency. Without introducing any parameters and computational complexity, this method is able to improve the performance on RGB-D segmentation over baseline by a large margin. Moreover, depth-aware CNN is flexible and easily replaces its plain counterpart in standard CNNs. Comparison with the state-of-the-art methods and extensive ablation studies on RGB-D semantic segmentation demonstrate the effectiveness and efficiency of depth-aware CNN.

Depth-aware CNN provides a general framework for vision tasks with RGB-D input. Moreover, depth-aware CNN takes the raw depth image as input and bridges the gap between 2D CNN and 3D geometry. In future works, we will apply depth-aware CNN on various tasks such as 3D detection, instance segmentation and we will perform depth-aware CNN on more challenging dataset. Apart from depth input, we will exploit more geometric input such as normal map.

Acknowledgements. We thank Ronald Yu, Yi Zhou and Qiangui Huang for the discussion and proofread. This research is supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/ Interior Business Center (DOI/IBC) contract number D17PC00288. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DOI/IBC, or the U.S. Government.

References

1. Armeni, I., Sax, A., Zamir, A.R., Savarese, S.: Joint 2D–3D-semantic data for indoor scene understanding. ArXiv e-prints (2017)
2. Barron, J.T., Poole, B.: The fast bilateral solver. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9907, pp. 617–632. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46487-9_38
3. Chen, J., Paris, S., Durand, F.: Real-time edge-aware image processing with the bilateral grid. ACM Trans. Graph. (TOG) **26**, 103 (2007)
4. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Semantic image segmentation with deep convolutional nets and fully connected CRFs. In: ICLR (2015)
5. Cheng, Y., Cai, R., Li, Z., Zhao, X., Huang, K.: Locality-sensitive deconvolution networks with gated fusion for RGB-D indoor semantic segmentation. In: CVPR (2017)
6. Couprie, C., Farabet, C., Najman, L., Lecun, Y.: Indoor semantic segmentation using depth information. In: ICLR (2013)
7. Dai, J., et al.: Deformable convolutional networks. In: ICCV (2017)
8. Eigen, D., Fergus, R.: Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In: ICCV (2015)
9. Gupta, S., Girshick, R., Arbeláez, P., Malik, J.: Learning rich features from RGB-D images for object detection and segmentation. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8695, pp. 345–360. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10584-0_23
10. Hazirbas, C., Ma, L., Domokos, C., Cremers, D.: FuseNet: incorporating depth into semantic segmentation via fusion-based CNN architecture. In: Lai, S.-H., Lepetit, V., Nishino, K., Sato, Y. (eds.) ACCV 2016. LNCS, vol. 10111, pp. 213–228. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54181-5_14
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)

12. He, Y., Chiu, W.C., Keuper, M., Fritz, M.: STD2P: RGBD semantic segmentation using spatio-temporal data-driven pooling. In: CVPR (2017)
13. Huang, Q., Wang, W., Neumann, U.: Recurrent slice networks for 3D segmentation on point clouds. In: CVPR (2018)
14. Huang, Q., Wang, W., Zhou, K., You, S., Neumann, U.: Scene labeling using gated recurrent units with explicit long range conditioning. arXiv preprint arXiv:1611.07485 (2016)
15. Jaderberg, M., Simonyan, K., Zisserman, A., kavukcuoglu, k.: Spatial transformer networks. In: NIPS (2015)
16. Janoch, A., et al.: A category-level 3-d object dataset: Putting the kinect to work. In: ICCV workshop (2011)
17. Khan, S.H., Bennamoun, M., Sohel, F., Togneri, R.: Geometry driven semantic labeling of indoor scenes. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8689, pp. 679–694. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10590-1_44
18. Li, Z., Gan, Y., Liang, X., Yu, Y., Cheng, H., Lin, L.: LSTM-CF: unifying context modeling and fusion with LSTMs for RGB-D scene labeling. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9906, pp. 541–557. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46475-6_34
19. Lin, D., Chen, G., Cohen-Or, D., Heng, P.A., Huang, H.: Cascaded feature network for semantic segmentation of RGB-D images. In: ICCV (2017)
20. Lin, G., Milan, A., Shen, C., Reid, I.: RefineNet: multi-path refinement networks for high-resolution semantic segmentation. In: CVPR (2017)
21. Ma, L., Stueckler, J., Kerl, C., Cremers, D.: Multi-view deep learning for consistent semantic mapping with RGB-D cameras. In: IROS (2017)
22. Silberman, N., Hoiem, D., Kohli, P., Fergus, R.: Indoor segmentation and support inference from RGBD images. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7576, pp. 746–760. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33715-4_54
23. Park, S.J., Hong, K.S., Lee, S.: RDFNet: RGB-D multi-level residual feature fusion for indoor semantic segmentation. In: ICCV (2017)
24. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: deep learning on point sets for 3d classification and segmentation. In: CVPR (2017)
25. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: deep hierarchical feature learning on point sets in a metric space. In: NIPS (2017)
26. Qi, X., Liao, R., Jia, J., Fidler, S., Urtasun, R.: 3d graph neural networks for RGBD semantic segmentation. In: ICCV (2017)
27. Ren, X., Bo, L., Fox, D.: RGB-(D) scene labeling: features and algorithms. In: CVPR (2012)
28. Shelhamer, E., Long, J., Darrell, T.: Fully convolutional networks for semantic segmentation. In: PAMI (2016)
29. Song, S., Lichtenberg, S.P., Xiao, J.: SUN RGB-D: A RGB-D scene understanding benchmark suite. In: CVPR (2015)
30. Song, S., Xiao, J.: Deep sliding shapes for amodal 3D object detection in RGB-D images. In: CVPR (2016)
31. Song, S., Yu, F., Zeng, A., Chang, A.X., Savva, M., Funkhouser, T.: Semantic scene completion from a single depth image. In: CVPR (2017)
32. Wang, J., Wang, Z., Tao, D., See, S., Wang, G.: Learning common and specific features for RGB-D semantic segmentation with deconvolutional networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9909, pp. 664–679. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46454-1_40

33. Wang, W., Huang, Q., You, S., Yang, C., Neumann, U.: Shape inpainting using 3d generative adversarial network and recurrent convolutional networks. In: ICCV (2017)
34. Wang, W., Wang, N., Wu, X., You, S., Yang, C., Neumann, U.: Self-paced cross-modality transfer learning for efficient road segmentation. In: ICRA (2017)
35. Wang, W., Yu, R., Huang, Q., Neumann, U.: SGPN: similarity group proposal network for 3d point cloud instance segmentation. In: CVPR (2018)
36. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. In: ICLR (2016)
37. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Scene parsing through ADE20K dataset. In: CVPR (2017)