# Recurrent Scene Parsing with Perspective Understanding in the Loop

**Shu Kong Charless Fowlkes**
Department of Computer Science
University of California, Irvine
Irvine, CA 92697, USA
{skong2, fowlkes}@ics.uci.edu

## Abstract

Objects may appear at arbitrary scales in perspective images of a scene, posing a challenge for recognition systems that process an image at a fixed resolution. We propose a depth-aware gating module that adaptively chooses the pooling field size in a convolutional network architecture according to the object scale (inversely proportional to the depth) so that small details can be preserved for objects at distance and a larger receptive field can be used for objects nearer to the camera. The depth gating signal is provided from stereo disparity (when available) or estimated directly from a single image. We integrate this depth-aware gating into a recurrent convolutional neural network trained in an end-to-end fashion to perform semantic segmentation. Our recurrent module iteratively refines the segmentation results, leveraging the depth estimate and output prediction from the previous loop. Through extensive experiments on three popular large-scale RGB-D datasets, we demonstrate our approach achieves competitive semantic segmentation performance using more compact model than existing methods. Interestingly, we find segmentation performance improves when we estimate depth directly from the image rather than using "ground-truth" and the model produces state-of-the-art results for quantitative depth estimation from a single image.

## 1 Introduction

An intrinsic challenge of parsing rich scenes is understanding object layout relative to the camera. Roughly speaking, the scales of the objects in the image frame are inversely proportional to the distance to the camera. Humans easily recognize objects even when they range over many octaves of spatial resolution, e.g., the cars near the camera in urban scene can appear a dozen times larger than those at distance. The huge range and arbitrary scale at which objects appear poses difficulties for machine image understanding. Although individual local features (e.g., in a deep neural network) can exhibit some degree of scale-invariance, it is not obvious they handle the range scale variation that exists in images.

In this paper, we investigate how cues to perspective geometry conveyed by image content, estimated from stereo disparity, or measured directly via specialized sensors, might be exploited to improve recognition and scene understanding. We focus specifically on the tasks of semantic segmentation which seeks to produce per-pixel category labels.

One straightforward approach is to stack the depth map with RGB image as a four-channel input tensor which can then be processed using standard architectures. In practice, this RGB-D input has not proven successful and sometimes even results in worse performance [12, 24]. We conjecture including depth as a per-pixel input doesn't adequately address scale-invariance in learning; such models lack an explicit mechanism to generalize to depths not observed during training and hence

still require training examples with object instances at many different scales to learn a multiscale appearance model.

Instead, our approach takes inspiration from the work of Ladicky *et al.* [17], who propose using depth estimates to rescale local image patches to a pre-defined canonical depth prior to analysis. For patches contained within a fronto-parallel surface, this can provide true depth-invariance over a range of scales (limited by sensor resolution for small objects) while effectively augmenting the training data available for the canonical depth. Rather than rescaling the input image, we propose a depth gating module that adaptively selects pooling field sizes over higher-level feature activation layers in a convolutional neural network (CNN). Adaptive pooling works with a more abstract notion of scale than standard multiscale image pyramids that operate on input pixels. This mechanism allows spatially varying processing over the visual field which can capture context for semantic segmentation that is not too large or small, but "just right", maintaining details for objects at distance while simultaneously using much larger receptive fields for objects closer near the camera. The gating architecture is trained with a loss that encourages selection of target pooling scales derived from "ground-truth" stereo disparity but at test time makes accurate inferences about scene depth using only monocular cues.

Inspired by studies of human visual processing (e.g., [5]) that suggest dynamic allocation of computation depending on the task and image content (background clutter, occlusion, object scale), we propose embedding gated pooling inside a recurrent refinement module that takes initial estimates of high-level scene semantics as a top-down signal to reprocess feed-forward representations and refine the final scene segmentation (similar to the recurrent module proposed in [3] for human pose). This provides a simple implementation of "Biased Competition Theory" [2] which allows top-down feedback to suppress irrelevant stimuli or incorrect interpretations, an effect we observe qualitatively in our recurrent model near object boundaries and in cluttered regions with many small objects.

We train this recurrent adaptive pooling CNN architecture end-to-end and evaluate its performance on several scene parsing datasets. The monocular depth estimates produced by our gating channel yield state-of-the-art performance on the NYU-depth-v2 benchmark [27]. We find that using this gating signal to modulate pooling inside the recurrent refinement architecture results in improved semantic segmentation performance over fixed multiresolution pooling. The resulting system matches state-of-the-art segmentation performance on three large-scale datasets using a model which, thanks to recurrent computation, is substantially more compact than many existing approaches.

## 2 Related work

Starting from the "fully convolutional" architecture of [23], there has been a flurry of recent work exploring CNN architectures for semantic segmentation and other pixel-labeling tasks. The seminal DeepLab [4] model modifies the very deep residual neural network [13] for semantic segmentation using dilated or atrous convolution operators to maintain spatial resolution in high-level feature maps. To leverage features conveying finer granularity lower in the CNN hierarchy, it has proven useful to combine features across multiple layers (see e.g., FCN [23], LRR [10] and RefineNet [21]). To simultaneously cover larger fields-of-view and incorporate more contextual information, Zhao *et al.* [30] concatenates features pooled over different scales.

The role of perspective geometry and geometric context in object detection was emphasized by a line of work starting with [15] and others (e.g., [1]) and has played an increasingly important role, particularly for scene understanding in urban environments [9]. Starting from the work of [14, 26], estimating depth from (monocular) scene semantics has also been examined in a variety of indoor and indoor settings (see e.g., [18]). Ladicky *et al.* [17] showed reliable depth recovery from image patches (i.e., without vanishing point estimation) and that the resulting estimates can be used to estimate object scale and improve segmentation in turn. Accurate monocular depth estimation using a multiscale deep CNN architecture was shown by Eigen *et al.* [8] using a geometrically inspired regression loss. Follow-on work [7] showed that depth, surface orientation and semantic labeling predictions can benefit each other in a multi-task setting using a shared network model for feature extraction.

Finally, there have been a number of proposals to carry out high-level recognition tasks such as human pose estimation [19, 3] and semantic segmentation [25] using recurrent or iterative processing. As pixel-wise labelling tasks are essentially a structured prediction problem, there has also been a

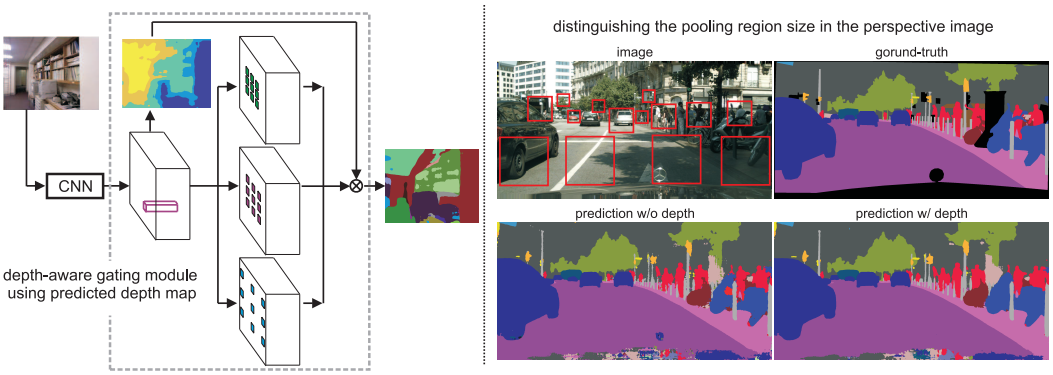


Figure 1: Left: depth-aware gating module using the predicted depth map. Right: an example image with ground-truth, segmentation result with and without the depth gating module. Rectangles overlayed on the image indicate pooling field sizes which are adapted based on the local depth estimate. We quantize the depth map into five discrete scales in our experiments. Using depth-gated pooling yields more accurate segment label predictions by avoiding pooling across small multiple distant objects while simultaneously allowing using sufficiently large pooling fields for nearby objects.

related line of work that aims to embed unrolled conditional random fields into differentiable CNN architectures to allow for more tractable learning and inference (e.g., [31]).

## 3 Depth-aware Gating Module

Our depth-aware gating module utilizes estimated depth at each image location as a proxy for object scale in order to select the appropriate spatial extent over which to pool features. Informally speaking, for a given object category (e.g., cars) the size of an object in the image is inversely proportional to the distance from the camera. Thus, if a region of an image has a larger depth values, the windows over which features are pooled (pooling field size) should be smaller in order to avoid pooling responses over many small objects and capture details needed to precisely segment small objects. For regions with small depth values, the same object will appear much larger and the pooling field size should be scaled up in a covariant manner to capture sufficient contextual appearance information in the vicinity of the object.

This depth-aware gating can readily utilize depth maps derived from stereo disparity or specialized time-of-flight sensors. Raw depth maps typically contain missing data and measurement noise due to oblique view angle, reflective surface and occlusion boundary. While these estimates can be improved using more extensive off-line processing (e.g., [28]), in our experiments we use these "raw" measurements. When such depth measurements are not available, the depth-aware gating can alternately exploit depth estimated directly from monocular cues. The left panel of Figure 6, illustrates the architecture of our depth-aware gating module using monocular depth predictions derived from the same front-end feature extractor.

Regardless of the source of the depth map, we quantize the depth into a discrete set of predicted scales (5 in our experiments). The scale prediction at each image location is then used to multiplicatively gate between a set of feature maps computed with corresponding pooling regions and summed to produce the final feature representation for classification. In the depth gating module, we use atrous convolution with different dilate rates to produce the desired pooling field size on each branch.

When training a monocular depth prediction branch, we quantize the ground-truth depth and treat it as a five-way classification using a softmax loss. For the purpose of quantitatively evaluating the accuracy of such monocular depth prediction, we also train a depth regressor over the input feature of the module using a simple Euclidean loss for the depth map $\mathbf{D}$ in log-space:

$$\ell_{depthReg}(\mathbf{D}, \mathbf{D}^*) = \frac{1}{|M|} \sum_{(i,j) \in M} \|\mathbf{D}_{ij} - \mathbf{D}^*_{ij}\|_2^2, \tag{1}$$

where $\mathbf{D}^*$ is the ground-truth depth. Since our "ground-truth" depth may have missing entries, we only compute the loss over pixels inside a mask $M$ which indicates locations with valid ground-truth

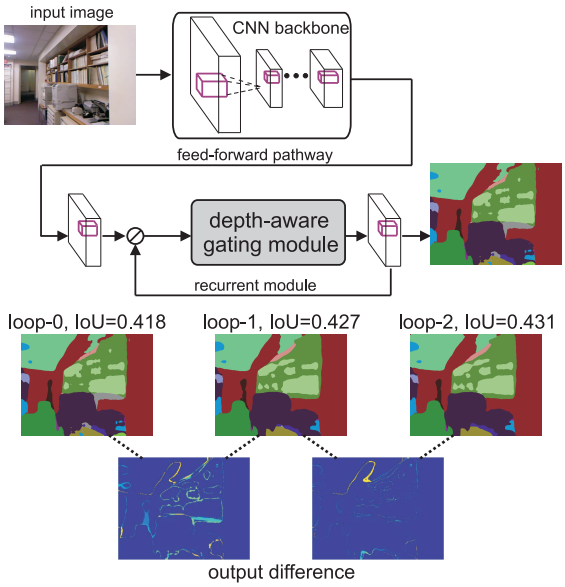


Figure 2: The input to our recurrent module is the concatenation (denoted by $\oslash$) of the feature map from an intermediate layer of the feed-forward pathway with the prior recurrent prediction. Our recurrent module utilizes depth-aware gating which carries out both depth regression and quantized prediction. Updated depth predictions at each iteration gate pooling fields used for semantic segmentation. This recurrent update of depth estimation increases the flexibility and representation power of our system yielding improved segmentation. We illustrate the prediction prior to, and after two recurrent iterations for a particular image. We also visualize the difference in predictions between consecutive iterations and note the gains in accuracy at each iteration as measured by average intersection-over-union (IoU) benchmark performance.

depth. For benchmarking we convert the log-depth predictions back to depths using an element-wise exponential of the predicted depth map, i.e. $\exp(\mathbf{D})$. Although more specific depth-oriented losses have been explored [8, 7], we show in experiment that this simplistic Euclidean loss on log-depth achieves state-of-the-art monocular depth estimation when combined with our architecture for semantic segmentation.

In our experiments, we evaluate models based on RGB-D images (where the depth channel is used for gating) and on RGB images using the monocular depth estimation branch. We find that using predicted (monocular) depth to gate segmentation feature maps yields better performance than models using the ground-truth depth. This is a surprising, but desirable outcome, as it avoids the need for extra sensor hardware and/or additional computation for refining depth estimates from multiple video frames (e.g., [28]).

## 4 Recurrent Refinement Module

Our recurrent refinement module takes as input feature maps extracted from a feed-forward CNN model along with current segmentation predictions available from previous iterations of the recurrent module. These are concatenated into a single feature map. This allows the recurrent module to provide an anytime segmentation prediction which can be further dynamically refined in future iterations. The recurrent refinement module has multiple convolution layers, each of which is followed by a ReLU and batch normalization layers. We also insert the depth-aware gating module in the recurrent module, allowing the depth output to serve as a top-down signal for use in refining the segmentation (as shown in experiments below). Figure 2 depicts our final recurrent architecture using the depth-aware gating module inside.

For a semantic segmentation problem with $K$ semantic classes, we use a $K$-way softmax classifier on individual pixels to train our network. With the depth-aware gating module, our objective function utilizes multiple losses weighted by hyperparameters:

$$\ell = \sum_{l=0}^{L} (\ell^l_{segCls} + \lambda_r \ell^l_{depthReg} + \lambda_c \ell^l_{depthCls}), \tag{2}$$

where $L$ means we unroll the recurrent module into $L$ loops and $l = 0$ denotes the prediction from the feed-forward pathway. The three losses $\ell^l_{segCls}$, $\ell^l_{depthReg}$ and $\ell^l_{depthCls}$ correspond to the semantic segmentation, depth regression and quantized depth classification at loop $l$, respectively. We use $\lambda_r$ and $\lambda_c$ to weight the importance of the depth losses. Throughout our experiments, we set them small value, $\lambda_r = 0.02$ and $\lambda_c = 0.1$, as our main task is improving semantic segmentation. As elaborated

in the next section, we train the modules in a stage-wise procedure, culminating in end-to-end training using the full objective.

# 5 Implementation

We implement our model with the MatConvNet toolbox [29] and train using SGD on a single Titan X GPU. We use the pre-trained ResNet50 and ResNet101 models [13] as the backbone of our models. To increase the output resolution of ResNet, like [4], we remove the top global $7 \times 7$ pooling layer and the last two $2 \times 2$ pooling layers. Instead we apply atrous convolution with dilation rate 2 and 4, respectively to maintain a spatial sampling rate which is of $1/8$ resolution to the original image size (rather than $1/32$ resolution if all pooling layers are kept). To obtain a final full resolution segmentation prediction, we simply apply bilinear interpolation on the softmax class scores to upsample the output by a factor of eight.

We train our models in a stage-wise procedure. First, we train a feed-forward baseline model. The feed-forward module is similar to DeepLab [4], but we add two additional $3 \times 3$-kernel layers (without atrous convolution) on top of the ResNet backbone. Starting from this baseline, we replace the second $3 \times 3$-kernel layer with our depth prediction and depth-aware gating module. We train the recurrent refinement module (containing the depth-aware gating), unrolling one layer at a time, and fine-tune the whole system using the objective function of Eq. 2.

We augment the training set using random data transforms. Specifically, we use random scaling by $s \in [0.5, 2]$, in-plate rotation by degrees in $[-10^\circ, 10^\circ]$, random left-right flip with 0.5 probability, random crop with sizes around $700 \times 700$ divisible by 8, and color jittering. Note that when scaling the image by $s$, we also divide the depth values by $s$. All these data transforms can be performed in-place with minimal computational cost.

When training the model, we fix the batch normalization in ResNet backbone, using the same constant global moments in both training and testing. Throughout training, we set batch size to one where the batch is a single input image (or a crop of a very high-resolution image). As we do not update batch normalization layers in the ResNet backbone, we do not have problems in training even with batch size as one. We use the "poly" learning rate policy [4] with a base learning rate of $2.5e-4$ scaled as a function of iteration by $(1 - \frac{iter}{maxiter})^{0.9}$. The code and trained models will be be made public[1].

# 6 Experiments

To show the effectiveness of our approach, we carry out comprehensive experiments on three large-scale RGB-D datasets (introduced below). We start with a quantitative evaluation of our monocular depth predictions which achieves state-of-the-art performance. We the describe ablation experiments to determine whether our depth-aware gating module improves semantic segmentation, validate the benefit of our recurrent module, and compare ground-truth and predicted depth-gating. Finally, we compare our complete model with the state-of-the-art methods on the semantic segmentation.

## 6.1 Datasets and Benchmarks

For our primary task of semantic segmentation, we use the standard Intersection-over-Union (IoU) criteria to measure the performance. As is common practice, we also report the per-pixel prediction accuracy for the first two datasets to allow comparison to existing approaches.

**NYUD-depth-v2** [27] consists of 1,449 RGB-D indoor scene images of the resolution $640 \times 480$ which include color and pixel-wise depth obtained by a Kinect sensor. We use the ground-truth segmentation into 40 classes provided in [11] and a standard train/test split into 795 and 654 images respectively.

**SUN-RGBD** [28] is an extension of NYUD-depth-v2 dataset [27], containing 5,285 training images and 5,050 testing images. It provides pixel labelling masks for 37 classes, and depth maps using different depth cameras. While this dataset provides refined depth maps (exploiting depth from the

[1]For code and models, please see project page `http://www.ics.uci.edu/~skong2/recurrentDepthSeg`.

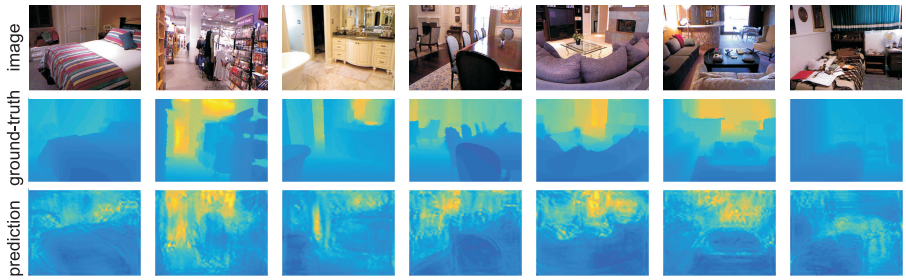


Figure 3: Examples of depth results. First row: the input RGB image; second row: ground-truth; third row: our result. In our visualizations, all depth maps use the same fixed (absolute) colormap to represent metric depth.

neighborhood video frames), the ground-truth depth maps still have significant noisy/mislabeled depth (examples can be found in our supplemental material).

**Cityscapes** [6] contains high quality pixel-level annotations of images collected in street scenes from 50 different cities. The training, validation, and test sets contain 2,975, 500, and 1,525 images respectively labeled for 19 semantic classes. The images of Cityscapes are of high resolution ($1024 \times 2048$), which makes training challenging due to limited GPU memory. We randomly crop out sub-images of $800 \times 800$ resolution for training.

### 6.2 Depth Prediction

In developing our approach, accurate depth prediction was not the primary focus but rather an intermediate quantized signal used to select the pooling field size. However, to validate our depth prediction, we also trained a depth regressor and compare the resulting predictions with previous work. We evaluated our model on NYU-depth-v2 dataset, on which a variety of depth prediction methods have been tested. Results on SUN-RGBD dataset can be found in the appendix. We report the performance by the widely used threshold metrics, i.e., the percentage of predicted depth pixels $d_i$ s.t. $\max(\frac{d_i}{d_i^*}, \frac{d_i^*}{d_i}) = \delta < threshold$, where $threshold$ is set to $1.25$, $1.25^2$ and $1.25^3$.

Table 1 provides a quantitative comparison of our predictions with several published methods. We can see our model trained with the Euclidean loss is quite competitive and achieves significantly better performance in the $\delta < 1.25$ metric. This simplistic loss compares well to, e.g., Eigen and Fergus [7] who develop a scale-invariant loss and use first-order matching term which compares image gradients of the prediction with the ground-truth.

In Figure 3, we visualize our estimated depth maps on the NYU-depth-v2 dataset[2]. Visually, we can see our predicted depth maps tend to be noticeably less smooth than true depth maps. Inspired by Eigen and Fergus [7] who advocate modeling smoothness in the local prediction, we apply Gaussian smoothing on our predicted depth map. This simple post-process is sufficient to outperform the state-of-the-art. We attribute the success of our depth estimator to several factors. First, we train our depth prediction branch on the features used for semantic segmentation. This is essentially a multi-task problem and the semantic segmentation may understandably help depth prediction. Second, we use a deeper architecture than that in [7] which has generally been shown to improve performance on a variety vision tasks as reported in literature.

### 6.3 Semantic Segmentation

To validate the proposed depth-aware gating module and the recurrent refinement module, we evaluate variants of our model with and without these components. We list the performance details in the first six rows in Table 2. The results are consistent across models trained independently on the three datasets. Adding depth maps for gating feature pooling brings noticeable boost in segmentation

[2]We also evaluate our depth prediction on SUN-RGBD dataset, and achieve 0.754, 0.899 and 0.961 by the three threshold metrics. As SUN-RGBD is an extension of NYU-depth-v2 dataset, it has similar data statistics resulting in similar prediction performance. Examples of depth prediction on SUN-RGBD dataset can be found in the appendix.

Table 1: Depth prediction on NYU-depth-v2 dataset.

| | Ladicky [17] | Liu [22] | Eigen [8] | Eigen [7] | Ours | Ours-Blur |
|---|---|---|---|---|---|---|
| $\delta < 1.25$ | 0.542 | 0.614 | 0.614 | 0.769 | 0.809 | 0.816 |
| $\delta < 1.25^2$ | 0.829 | 0.883 | 0.888 | 0.950 | 0.945 | 0.950 |
| $\delta < 1.25^3$ | 0.940 | 0.971 | 0.972 | 0.988 | 0.986 | 0.989 |

Table 2: Performance of semantic segmentation on different datasets. Results marked by $^*$ are evaluated by the dataset server on test set. Note that we train our models based on ResNet50 architecture on NYU-depth-v2 and SUN-RGBD datasets, and ResNet101 on the Cityscapes dataset.

| | NYU-depth-v2 [27] | | SUN-RGBD [27] | | Cityscapes [6] |
|---|---|---|---|---|---|
| | IoU | pixel acc. | IoU | pixel acc. | IoU |
| baseline | 0.406 | 0.703 | 0.402 | 0.776 | 0.738 |
| w/ gt-depth | 0.413 | 0.708 | 0.422 | 0.787 | 0.753 |
| w/ pred-depth | 0.418 | 0.711 | 0.423 | 0.789 | 0.759 |
| loop1 w/o depth | 0.419 | 0.706 | 0.432 | 0.793 | 0.762 |
| loop1 w/ gt-depth | 0.425 | 0.711 | 0.439 | 0.798 | 0.769 |
| loop1 w/ pred-depth | 0.427 | 0.712 | 0.440 | 0.798 | 0.772 |
| loop2 | 0.431 | 0.713 | 0.443 | 0.799 | 0.776 |
| loop2 (test-aug) | 0.445 | 0.721 | 0.451 | 0.803 | 0.791 / $0.782^*$ |
| DeepLab [4] | - | - | - | - | 0.704 / $0.704^*$ |
| LRR [10] | - | - | - | - | 0.700 / $0.697^*$ |
| Context [20] | 0.406 | 0.700 | 0.423 | 0.784 | - / $0.716^*$ |
| PSPNet [30] | - | - | - | - | - / $0.784^*$ |
| RefineNet-Res50 [21] | 0.438 | - | - | - | - / - |
| RefineNet-Res101 [21] | 0.447 | - | 0.457 | 0.804 | - / $0.736^*$ |
| RefineNet-Res152 [21] | 0.465 | 0.736 | 0.459 | 0.806 | - / - |

performance, with greatest improvements on the large-scale datasets (SUN-RGBD and Cityscapes) compared to the smaller NYU-depth-v2 dataset.

Interestingly, we achieve slightly better performance using the predicted depth map rather than the provided ground-truth depth. We attribute this to three explanations. Firstly, the predicted depth is smooth without holes or invalid entries. When using ground-truth depth on Cityscapes[3], we assign equal weight on the missing entries so that the gating actually average the information at different scales. This average pooling might be harmful in some cases such as a very small object at a distance. Secondly, the predicted depth maps show some object-aware patterns (e.g., car region shown in the visualization in Figure 5), which might be helpful for class-specific segmentation. Thirdly, the model is trained end-to-end so co-adaption of the depth prediction and segmentation branches may increase the overall representation power and flexibility of the whole model, benefiting the final predictions.

Table 2 also shows the benefit of the recurrent refinement module as seen in the improved performance from baseline to loop1 and loop2. Equipped with depth in the recurrent module, the improvement is more notable. As with the pure feed-forward model, using predicted depth maps in the recurrent module yields slight gains over the ground-truth depth. We observe that performance improves using a depth 2 unrolling (third group of rows in Table 2) but saturates/converges after two iterations.

In comparing with state-of-the-art methods, we follow common practice of augmenting images at test time by running the model on flipped and rescaled variants and average the class scores to produce the final segmentation output (compare loop2 and loop2 (test-aug)). We can see our model performs on par or better than recently published results listed in Table 2.

Note that for NYU-depth-v2 and SUN-RGBD, our backbone architecture is ResNet50, whereas RefineNet reports the results using a much deeper models (ResNet101 and ResNet152) which typically outperform shallower networks in vision tasks. For the Cityscapes, we also submitted our final result for held-out benchmark images which were evaluated by the Cityscapes benchmark server. Our model achieves IoU $0.782$, on par with the best reported result, IoU $0.784$, by PSPNet. We did

[3]NYU-depth-v2 and SUN-RGBD datasets provide improved depth maps without invalid entries.

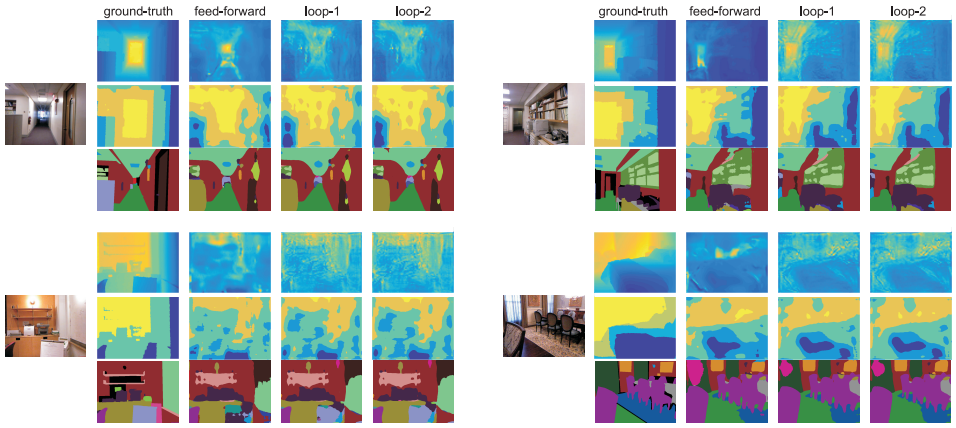


Figure 4: Visualization of the output on NYU-depth-v2. We show four randomly selected testing images with ground-truth and predicted disparity (first row), quantized disparity (second row) and segmentation (third row) at each iteration of the recurrent computation.

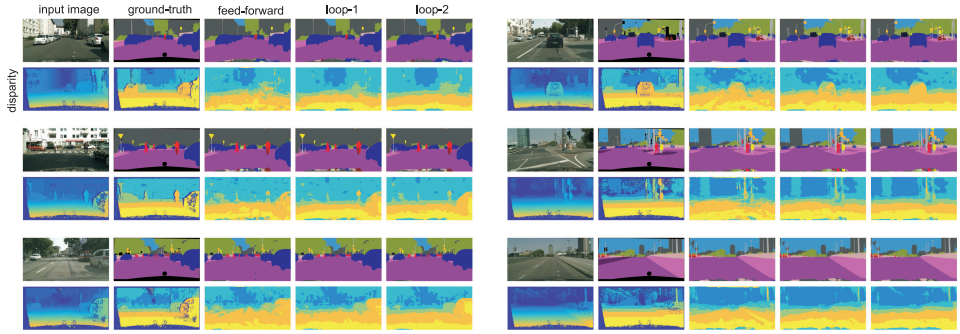


Figure 5: Visualization of six randomly selected validation images from Cityscapes with the segmentation output and the predicted quantized disparity at each iteration of the recurrent loop. We depict "ground-truth" continuous and quantized disparity beneath the input image. Our monocular disparity estimate makes predictions for reflective surfaces where stereo fails and recurrent iteration further improves estimates, particularly for featureless areas such as the pavement.

not perform any extensive performance tuning and only utilized the fine-annotation training images for training (without the twenty thousand coarse-annotation images and the validation set). We also didn't utilize any post-processing (such as the widely used fully-connected CRF [16] which typical yields additional performance increments).

One key advantage of recurrent refinement is that it allows richer computation (and better performance) without additional model parameters. Our ResNet50 model (used on the NYU-depth-v2 dataset) is relatively compact (221MB) compared to RefineNet-Res101 which achieves similar performance but is nearly double the size (426MB). Our model architecture is similar to DeepLab which also adopts pyramid atrous convolution (but simply averages output feature maps without any depth-guided adaptive pooling). However, the final DeepLab model utilizes an ensemble which yields a much larger model (530MB). PSPNet concatenates the intermediate features into 4,096 dimension before classification while our model operates on small 512-dimension feature maps.

### 6.4 Visualization

In Figure 4 and 5, we visualize several randomly selected examples from the test set of NYU-depth-v2 and Cityscapes. In the figures, we show both the segmentation results and the depth maps being updated in the recurrent loops. Interestingly, the depth maps on Cityscapes change more noticeably than those on NYU-depth-v2 dataset. Regions in the predicted depth map corresponding to objects, such as the car, are grouped together and disparity estimates on texture-less regions such as the street surface improve across iterations. Gains for the NYU-depth-v2 data are less apparent. We conjecture

this is because images in NYU-depth-v2 are more varied in overall layout and often have less texture and fewer objects from which the model can infer semantics and subsequently depth. In both figures, we can see that our model is able to exploit recurrence to to correct misclassified regions/pixels "in the loop", visually demonstrating the effectiveness of the recurrent refinement module.

## 7 Conclusion and discussion

In this paper, we have proposed a depth-aware gating module that uses depth estimates to adaptively modify the pooling field size at a high level layer of neural network for better segmentation performance. The adaptive pooling can use large pooling fields to include more contextual information to segment large nearby objects while maintaining fine-scale detail for objects further from the camera. While our model can utilize stereo disparity when available, we find that using such data to train a depth predictor which is subsequently used at test-time in place of stereo ultimately yields better performance. Additionally, we also demonstrate the utility of performing recurrent refinement of both semantic segmentation and depth predictions, which yields improved prediction accuracy without adding additional model parameters.

We envision that the recurrent refinement module can capture object shape priors, contour smoothness and region continuity. However, our current approach converges after a few iterations and performance saturates. This leaves open future work in exploring other training objectives that might push the recurrent computation towards producing more varied outputs. This might be further enriched in the setting of video where the recurrent component could be extended to incorporate memory of previous frames.

### Acknowledgments

This project is supported by NSF grants IIS-1618806, IIS-1253538, DBI-1262547 and a hardware donation from NVIDIA.

## References

[1] Bao, S. Y., Sun, M., and Savarese, S. (2011). Toward coherent object detection and scene layout understanding. *Image and Vision Computing*, **29**(9), 569–579.

[2] Beck, D. M. and Kastner, S. (2009). Top-down and bottom-up mechanisms in biasing competition in the human brain. *Vision research*, **49**(10), 1154–1165.

[3] Belagiannis, V. and Zisserman, A. (2016). Recurrent human pose estimation. *arXiv:1605.02914*.

[4] Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2016). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*.

[5] Cichy, R. M., Pantazis, D., and Oliva, A. (2014). Resolving human object recognition in space and time. *Nature neuroscience*, **17**(3), 455–462.

[6] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[7] Eigen, D. and Fergus, R. (2015). Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658.

[8] Eigen, D., Puhrsch, C., and Fergus, R. (2014). Depth map prediction from a single image using a multi-scale deep network. In *NIPS*.

[9] Geiger, A., Lauer, M., Wojek, C., Stiller, C., and Urtasun, R. (2014). 3d traffic scene understanding from movable platforms. *IEEE transactions on pattern analysis and machine intelligence*, **36**(5), 1012–1025.

[10] Ghiasi, G. and Fowlkes, C. C. (2016). Laplacian pyramid reconstruction and refinement for semantic segmentation. In *ECCV*.

[11] Gupta, S., Arbelaez, P., and Malik, J. (2013). Perceptual organization and recognition of indoor scenes from rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[12] Hazirbas, C., Ma, L., Domokos, C., and Cremers, D. (2016). Fusenet: Incorporating depth into semantic segmentation via fusion-based cnn architecture. In *ACCV*.

[13] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[14] Hoiem, D., Efros, A. A., and Hebert, M. (2005). Geometric context from a single image. In *Proceedings of the IEEE International Conference on Computer Vision*.

[15] Hoiem, D., Efros, A. A., and Hebert, M. (2008). Putting objects in perspective. *International Journal of Computer Vision*, **80**(1), 3–15.

[16] Krahenbuhl, P. and Koltun, V. (2011). Efficient inference in fully connected crfs with gaussian edge potentials. *NIPS*.

[17] Ladicky, L., Shi, J., and Pollefeys, M. (2014). Pulling things out of perspective. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[18] Lee, D. C., Hebert, M., and Kanade, T. (2009). Geometric reasoning for single image structure recovery. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2136–2143. IEEE.

[19] Li, K., Hariharan, B., and Malik, J. (2016). Iterative instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[20] Lin, G., Shen, C., van den Hengel, A., and Reid, I. (2016). Efficient piecewise training of deep structured models for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[21] Lin, G., Milan, A., Shen, C., and Reid, I. (2017). Refinenet: Multi-path refinement networks with identity mappings for high-resolution semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[22] Liu, F., Shen, C., and Lin, G. (2015). Deep convolutional neural fields for depth estimation from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[23] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[24] Luo, Z., Peng, B., Huang, D.-A., Alahi, A., and Fei-Fei, L. (2017). Unsupervised learning of long-term motion dynamics for videos. *arXiv:1701.01821*.

[25] Romera-Paredes, B. and Torr, P. H. S. (2016). Recurrent instance segmentation. In *ECCV*.

[26] Saxena, A., Chung, S. H., and Ng, A. Y. (2005). Learning depth from single monocular images. In *NIPS*.

[27] Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. (2012). Indoor segmentation and support inference from rgbd images. *ECCV*.

[28] Song, S., Lichtenberg, S. P., and Xiao, J. (2015). Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[29] Vedaldi, A. and Lenc, K. (2015). Matconvnet: Convolutional neural networks for matlab. In *Proceedings of the 23rd ACM international conference on Multimedia*.

[30] Zhao, H., Shi, J., Qi, X., Wang, X., and Jia, J. (2017). Pyramid scene parsing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[31] Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P. H. (2015). Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1529–1537.

## Appendix: Analysis of Depth-aware Gating Module

In this section, we analyze the proposed depth-aware gating module with detailed results in Table 3. We perform the ablation study on the Cityscapes dataset [6]. Specifically, we train the following models in order.

1. "baseline" is our DeepLab-like baseline model by training two convolutional (with $3 \times 3$ kernels) layers above the ResNet101 backbone.
2. "tied, avg." is the model we train based on "baseline" by using the same $3 \times 3$ kernel but different dilate rates equal to $\{1, 2, 4, 8, 16\}$, respectively. So there are five branches and each of them has the same kernels which are tied to make processing scale-invariant. We average the feature maps for the final output prior to classification.
3. "gt-depth, tied, gating" is the model using the ground-truth depth map to select the branch; the pooling window size is determined according to the ground-truth depth value.
4. "gt-depth, untied, gating" is the model based on "gt-depth, tied, gating" by unleashing the tied kernels in the five branches. These untied kernels improve the flexibility and representation power of the network. Figure 6 (a) depicts this model.
5. "pred-depth, untied, gating" is our final model in which we learn a quantized depth predictor to gate the five branches. This model determines the size of pooling window based on its predicted depth map. Figure 6 (b) shows the architecture of this model.

Through Table 3, we can see that increasing the dilate rate with our model "tied, avg." improves the performance noticeably. This is consistent with the observation in [4], in which the large view-of-field version of DeepLab performs better. The benefit can be explained by the large dilation rate increasing the size of the receptive field, allowing more contextual information to be captured at higher levels of the network. With the gating mechanism, either using ground-truth depth map or the predicted one, the performance is improved further over non-adaptive pooling. The depth-aware gating module helps determine the pooling window size wisely, which is better than averaging all branches equally as in our "tied, avg." model and DeepLab. Moreover, by unleashing the tied kernels, the "gt-depth untied, gating" improves over "gt-depth, tied, gating" remarkably. We conjecture that this is because the untied kernels provide more flexibility to distinguish features at different scales and allow selection of the appropriate non-invariant features from lower in the network. Interestingly, using the predicted depth map achieves the best among all these compared models. We attribute this to three reasons. Firstly, the predicted depth is smooth without holes or invalid entries. When using ground-truth depth on Cityscapes dataset, we assign equal weight on the missing entries so that the gating actually averages the information at different scales. This average pooling might be harmful in some cases such as very small object at distance. This can be taken as complementary evidence that the blindly averaging all branches achieves inferior performance to using the depth-aware gating. Secondly, the predicted depth maps have some object-aware pattern structure, which might be helpful for segmentation. From the visualization in the paper (not shown here), we can observe such patterns, e.g. for cars. Thirdly, the depth prediction branch generally increases the representation power and flexibility of the whole model; this can be beneficial for segmentation.

## Appendix: Results on the SUN-RGBD dataset

In Figure 7, we show the depth prediction results of several images randomly picked from the test set of SUN-RGBD dataset. Note that the there are unnatural regions in the ground-truth depth maps, which are the result of refined depth completion by the algorithm in [28]. Visually, these regions do not always make sense and constitute bad depth completions. In contrast, our predicted depth maps are much smoother. We also evaluate our depth prediction on SUN-RGBD dataset, and achieve 0.754, 0.899 and 0.961 by the three threshold metrics respectively. As SUN-RGBD is an extension of NYU-depth-v2 dataset, it has similar data statistics resulting in similar prediction performance.

In Figure 8, we randomly show fourteen images and their segmentation results at loops of the recurrent refining module. Visually, we can see that the our recurrent module refines the segmentation result in the loops.

Table 3: Result of different depth-aware gating module deployments on Cityscapes dataset. IoU is short for intersection over union averaged over all classes, and nIoU is the weighted IoU through the pre-defined class weights provided by the benchmark.

| | baseline | | tied, avg. | | gt-depth, tied, gating | | gt-depth, untied, gating | | pred-depth, untied, gating | |
|---|---|---|---|---|---|---|---|---|---|---|
| | IoU | nIoU | IoU | nIoU | IoU | nIoU | IoU | nIoU | IoU | nIoU |
| **Score Avg.** | **0.738** | **0.547** | **0.747** | **0.554** | **0.748** | **0.556** | **0.753** | **0.561** | **0.759** | **0.571** |
| road | 0.980 | – | 0.981 | – | 0.981 | – | 0.982 | – | 0.982 | – |
| sidewalk | 0.849 | – | 0.847 | – | 0.849 | – | 0.982 | – | 0.857 | – |
| building | 0.916 | – | 0.917 | – | 0.918 | – | 0.919 | – | 0.920 | – |
| wall | 0.475 | – | 0.499 | – | 0.506 | – | 0.511 | – | 0.512 | – |
| fence | 0.596 | – | 0.605 | – | 0.605 | – | 0.611 | – | 0.614 | – |
| pole | 0.598 | – | 0.599 | – | 0.604 | – | 0.616 | – | 0.624 | – |
| traffic light | 0.684 | – | 0.674 | – | 0.678 | – | 0.692 | – | 0.699 | – |
| traffic sign | 0.780 | – | 0.776 | – | 0.775 | – | 0.782 | – | 0.790 | – |
| vegetation | 0.918 | – | 0.917 | – | 0.918 | – | 0.920 | – | 0.922 | – |
| terrain | 0.619 | – | 0.620 | – | 0.627 | – | 0.632 | – | 0.638 | – |
| sky | 0.941 | – | 0.937 | – | 0.940 | – | 0.942 | – | 0.944 | – |
| person | 0.803 | 0.635 | 0.803 | 0.631 | 0.804 | 0.639 | 0.808 | 0.648 | 0.814 | 0.659 |
| rider | 0.594 | 0.448 | 0.595 | 0.462 | 0.602 | 0.460 | 0.612 | 0.461 | 0.616 | 0.473 |
| car | 0.939 | 0.859 | 0.942 | 0.854 | 0.942 | 0.863 | 0.942 | 0.867 | 0.944 | 0.871 |
| truck | 0.631 | 0.398 | 0.666 | 0.421 | 0.679 | 0.407 | 0.679 | 0.417 | 0.674 | 0.421 |
| bus | 0.759 | 0.595 | 0.802 | 0.607 | 0.787 | 0.612 | 0.786 | 0.609 | 0.799 | 0.615 |
| train | 0.621 | 0.467 | 0.683 | 0.494 | 0.656 | 0.489 | 0.655 | 0.487 | 0.687 | 0.507 |
| motorcycle | 0.562 | 0.396 | 0.587 | 0.387 | 0.591 | 0.398 | 0.602 | 0.410 | 0.610 | 0.425 |
| bicycle | 0.755 | 0.582 | 0.747 | 0.387 | 0.753 | 0.575 | 0.761 | 0.586 | 0.765 | 0.594 |

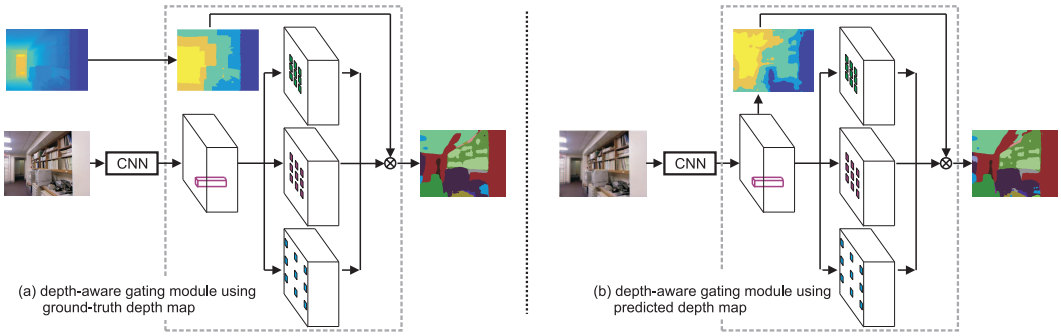


Figure 6: (a) Depth-aware gating module using the ground-truth depth map, and (b) depth-aware gating module using the predicted depth map. The grids within the feature map blocks distinguish different pooling field sizes. Here we depict three different pooling window sizes while in our actual experiments we quantize the depth map into five scale bins.

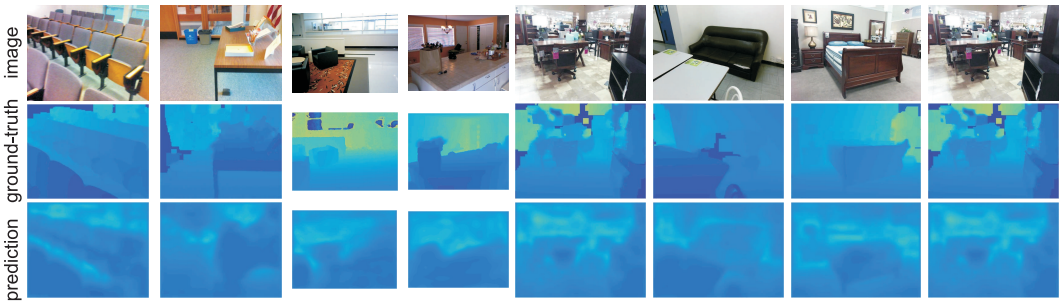


Figure 7: Visualization of images from SUN-RGBD dataset and their ground-truth depth and our predicted depth on the three rows, respectively. We scale all the depth maps into a fixed range of $[0, 10^5]$. In this sense, the color of the depth maps directly reflect the absolute physical depth. Note that there are unnatural regions in the ground-truth depth maps, which have been refined by the algorithm in [28]. Visually, these refined region do not always make sense and are incorrect depth completions. In contrast, our monocular predictions are quite smooth.

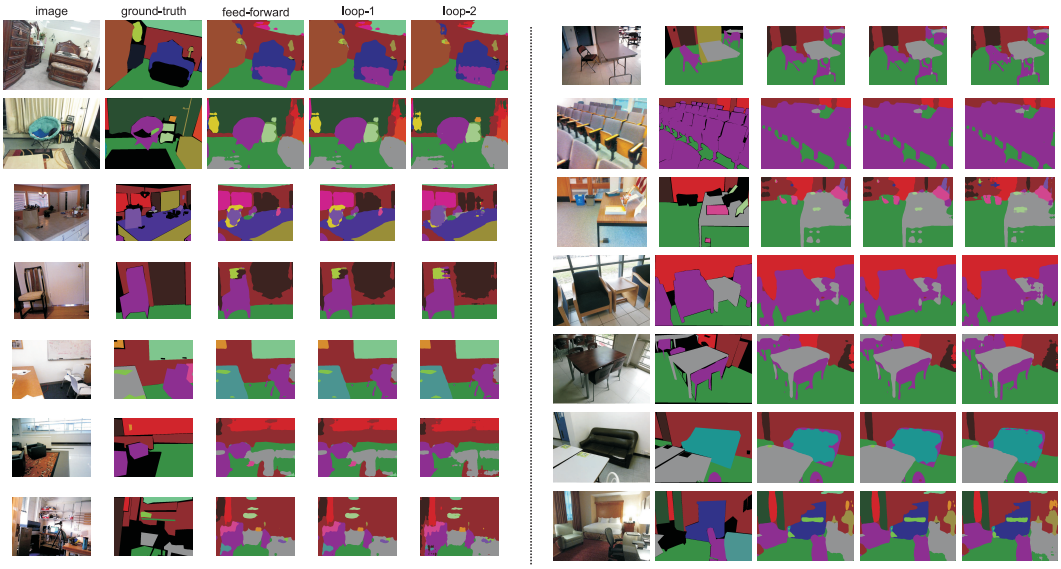


Figure 8: Visualization of the output on SUN-RGBD dataset. We randomly show fourteen images from validation set with their segmentation output from both feed-forward pathway and recurrent loops. In the ground-truth segmentation annotation, we can see that there are many regions (with black color) not annotated.