State-of-the-art systems for semantic image segmentation utilize feed-forward pipelines with fixed computational costs. Building an image segmentation system that works across a range of computational budgets is challenging and time-intensive as new architectures must be designed and trained for every computational setting. To address this problem we develop a recurrent neural network that successively improves prediction quality with each iteration. Importantly, the RNN may be employed across a range of computational budgets by merely running the model for varying numbers of iterations. The RNN achieves results comparable to state-of-the-art systems in image segmentation on PASCAL VOC 2012 and Cityscapes segmentation datasets, however the RNN performs this task with a fraction of the computational budget. Finally, we demonstrate how one may exploit the properties of the RNN to efficiently perform video segmentation with half the computational cost of a comparable, state-of-the-art image segmentation method.

1. Introduction

Advances in the design of neural networks have driven the field of computer vision in the last few years [42, 18, 45]. State-of-the-art artificial vision systems in image recognition [42, 64, 62], object detection [58, 25, 26, 21, 65], depth estimation [20], semantic segmentation [8, 76, 26, 52, 23, 59], instance segmentation [57, 4] and many other image processing tasks [67, 46] are built on top of novel network architectures.

Designing neural networks is challenging and time-intensive (but see [78]). Additionally, deploying a trained vision system based on a neural network is computationally expensive often requiring dedicated, highly parallelized numerical linear algebra implementations [1, 11, 10] on custom SIMD hardware architectures [41, 42, 11, 68]. In addition to these challenges, applications of vision systems span a wide range of computational budgets—from severe time or energy budgets [7, 39, 69, 40] to unconstrained domains (e.g. medical image diagnosis [59]).

Semantic image segmentation [8, 76, 26, 52, 23, 59] is one such task that is commonly employed across problems with vastly different computational demands—from mobile applications [47, 5, 6] to medical diagnosis [59].

As larger networks often lead to improved performance [70, 78], a critical aspect of network design is the ability to build systems that operate well across a range of computational budgets—including and especially, extremely small computational budgets [39, 75].

One may design and train a distinct network architecture for each computational budget [78], however this approach is time- and labor-intensive and may require the calibration of a cascade-based system [2]. Another approach is to build a fully convolutional network that operates on images of arbitrary size [8]. This approach leads to state-of-the-art models, but comes at the expense of building models that operate on images downsampled in spatial resolution.

We instead draw inspiration from the image compression literature, where a single network is often trained and deployed across a large range of computational demands [67]. A natural way to achieve this goal is through a recurrent network architecture [29], which may be iterated a number of times in proportion to a desired accuracy. Although convolutional networks are far more common in architectures for vision problems, recurrent networks have been applied across tasks such as multiple object recognition [3], fine-grain image recognition [61], instance segmentation [57], and image generation [31].

In this work we propose a recurrent network architecture with an intermediate human-interpretable canvas for semantic image segmentation (Figure 2 and 3). The network updates past predictions stored in the canvas to iteratively refine the segmentation. We demonstrate the performance of the network on the PASCAL VOC 2012 [22] and Cityscapes segmentation datasets, however the RNN performs this task with a fraction of the computational budget. Finally, we demonstrate how one may exploit the properties of the RNN to efficiently perform video segmentation with half the computational cost of a comparable, state-of-the-art image segmentation method.
Figure 1. Leveraging a recurrent network to efficiently segment videos. Seeding a recurrent network with the segmentation from a previous frame notably reduces the computational budget required to segment a video. Segmentations from a recurrent network across 180 × 270 and 1080 × 608 resolution videos (top row) require half the computational cost of a state-of-the-art feed-forward architecture [9] (bottom row), without sacrificing quality.

Figure 2. Recurrent segmentation iteratively improves segmentation predictions across a range of computational budgets. Images from the PASCAL VOC 2012 [22] validation dataset at 513 × 513 resolution (left column) are segmented with a recurrent network for iterations one through six (columns two through seven), using single-scale evaluation (see Section 3). The multi-scale RNN segmentation at the sixth of iteration (column eight) and the ground truth segmentation (column nine) are shown for comparison. Floating point operations per second (FLOPS) measures the cumulative computational cost of each RNN iteration.

2. Related Work

Early work on semantic image segmentation learned single layer representations built on top of hand-crafted features (see [8] for review). The resurgence of neural networks enabled learning rich image features for classifying individual pixels, taking into account context and multi-
Figure 3. Recurrent segmentation iteratively improves Cityscapes segmentation masks [13]. Subsequent iterations add finer spatial details. High resolution images (1028 × 2048) from the Cityscapes semantic segmentation dataset (left column) are segmented by one or seven iterations of a recurrent network (columns two and three). By the seventh iteration, the RNN has added finer spatial segmentations (e.g., pedestrians, bicyclists, and sign posts) to the previous segmentation of large-scale structures. Ground truth segmentation masks (right column) are provided for comparison.

scale information [33, 15, 19]. Broadly speaking, many of these systems employ convolutional, feed-forward networks to extract deep features at multiple levels (and spatial resolutions) in order to densely predict the label [51, 8, 54]. Two particular features of the feed-forward architectures are the usage of skip connections [35] to bring lower level features into later processing stages [59] and atrous or dilated convolutions [38] to efficiently increase the receptive field size of activation maps without sacrificing spatial resolution [60, 8, 55, 27].

Instead of relying on a strictly feed-forward architecture, our work focuses on the application of a recurrent network architecture to semantic segmentation. This work draws inspiration from early work applying semantic segmentation to images in small recurrent neural networks [56]. Pinheiro and Collobert [56] built convolutional recurrent networks and trained the resulting system on small datasets labeled for pixel-wise segmentation (e.g., Stanford Background, SIFT Flow). In this work, we scale up these networks substantially in depth and overall size. Notably, we employ the idea of a human-interpretable canvas that the network continuously (additively) updates during inference as previously used in generative models of images [31, 17, 63]. A human-interpretable canvas allows the network to make a viable prediction at each successive step of the recurrence.

Segmentation systems often employ Conditional Random Fields (CRFs) [43] as a post-processing step that uses image priors to improve the spatial consistency of segmentation masks [8]. Zheng et al. 2015 [77] reformulated CRFs as a constrained recurrent neural network (CRF-RNN) that performs approximate variational inference, and found that end-to-end learning of a CRF-RNN on top of a feed-forward convolutional network improved upon previous state-of-the-art results. We generalize the recurrent architecture and add a human interpretable canvas that allows for trading off speed and accuracy.

3. Methods

The goal of our architecture design is to build an end-to-end fully convolutional network that may operate on arbitrarily sized input images, where a feed-forward network provides features to a recurrent network. The output of the recurrent network iteratively updates a canvas of semantic segmentation, which is then fed back to the first recurrent layer at the next time step (Figure 4). We provide a brief summary of the major features below but save the details for the Appendix.

We compute generic image features as the output for conv4x (block4) from a ResNet-V1-101 network [36]. For a 513 × 513 image, the image features are of size 65 × 65 across 2048 feature channels (red box in Figure 4). This first portion of the network comprises 42.5M learned parameters and requires 54.4 gigaFLOPS (GFLOPS) to process a single 513 × 513 image. Although we experimented with training the convolutional features from scratch, we found that pre-trained features suffice and permit us to accelerate experimentation, hence all experiments presented employ ResNet features pre-trained on ImageNet image classification [16].

Many options exist for a recurrent network architecture, however we selected the LSTM [37] for ease of training [12]. In particular, we focused on a convolutional LSTM [73] (for definition see Appendix 5). All of the network weights are shared across spatial locations within an activation map. We used a 1 × 1 kernel size, which combines information only across activation maps. Preliminary experiments found that RNNs with larger than 1 × 1 kernels–either dense or dilated–underperformed RNNs with 1 × 1 kernels. In addition to learning the weights and biases during training, we also experimented with learning the initial state of the network. While these experiments produced
lower train error, they demonstrated poorer generalization; in the results presented here, we initialize the network state as zeros.

Previous work with sequence models [71] indicate that stacking multiple LSTM layers is a powerful method for capturing dependencies at multiple scales. In preliminary experiments we found that 3 layers provided a reasonable trade off between representational ability and training time. Each subsequent convolutional LSTM maintained the same spatial resolution while decreasing the output depth from the number of image feature channels to the number of semantic classes.

The final prediction of the network are the logits of an additive canvas [31]. Each iteration of the recurrent network provides additive adjustments to the logits for each class at each pixel location. The logits from each iteration are concatenated to the ResNet features to provide input on subsequent iterations of the recurrent network (Figure 4).

We used a softmax cross-entropy loss across the labels for each pixel. The loss was applied to the final canvas after $N$ iterations, where $N$ is a free parameter discussed below. In preliminary experiments we determined that weighted versions of the loss at earlier iterations of the canvas resulted in similar training performance, but changed the shape of the speed/accuracy curve. In particular, applying the loss to iteration three or earlier improved the prediction quality at earlier iterations, but decreased the quality of the best segmentation and produced a speed-accuracy trade-off curve that did not monotonically increase. All results shown here are for networks trained with a single loss applied at iteration six.

We trained the resulting architecture on the PASCAL VOC 2012 dataset [22] which contains 11,530 images of variable size, 6,929 of which have segmentation masks assigning each pixel into one of 20 classes (excluding the background class). We also trained models on the Cityscapes semantic segmentation dataset [13], which consists of 20,000 coarse training images, 3,475 fine validation images and 1,525 test images.

During training, optimization was performed using stochastic gradient descent with momentum, where the learning rate was reduced monotonically with a polynomial schedule (see Appendix for details).

We explore the RNN model and competing architectures across single-scale and multi-scale evaluation schemes. In the single-scale scheme, we perform a single inference pass at a coarse ResNet output stride (16). This evaluation scheme is generally steered towards mobile platforms with limited computational budgets. Additionally, we test the model in a less computationally restricted setting by performing multiple inference passes per image: resampling the image at six different spatial scales, performing left-right flips, and averaging the logits, as described in [9]. The multi-scale setting achieves the best quantitative per-
4. Results

We present results measuring the performance of the RNN architecture on image segmentation tasks. In addition, we highlight experiments indicating how the architecture may be applied to video segmentation in a computationally efficient manner. Finally, we examine the errors of the recurrent network to highlight how the system operates and suggest opportunities for improvement.

4.1. Image segmentation with variable computational budgets

We trained the recurrent architecture on the PASCAL VOC 2012 dataset [22] to segment images into twenty semantic classes. This dataset is a popular test bed for advancements in image segmentation [8, 76, 23]. The proposed RNN was unrolled for six iterations and a cross entropy loss was applied to the final canvas state (Figure 4). The final network is fully convolutional and may operate on arbitrarily-sized images, however we focus our analysis at two operation regimes.

First, we analyze the network at a $513 \times 513$ spatial resolution with a single inference pass and a ResNet output stride of 16. A single inference pass of one image at a single iteration results in 61.1 GFLOPs of computation (Figure 2 second column, Figure 6 single-scale, Table 3). In contrast, state-of-the-art architectures for image segmentation, such as [76] and [9], require over 100 GFLOPS at the same output stride, indicating that our model is less computationally demanding (Table 3). Interestingly, because of the structure of the RNN, the model may produce intermediate predictions (i.e. by calculating the argmax of the canvas, see Figure 4). Specifically, at inference time, one may continue to iterate the RNN, producing an improved segmentation (Figure 5).

We note that fully convolutional networks also have the ability to exchange accuracy for compute, e.g. by decreasing the image resolution or increasing the output stride of the feature extraction step [9]. When we compare the speed-accuracy trade-offs of changing output stride versus running recurrent segmentation for a variable number of iterations on a $513 \times 513$ image, we find that the RNN achieves a higher quality segmentation at lower computational cost (Figure 6).

We quantified performance on the PASCAL VOC 2012 test dataset in terms of the mean intersection-over-union
Figure 6. Trade-off between computational cost and segmentation accuracy. We plot the computational cost (FLOPS) versus model accuracy (mIOU) on the PASCAL VOC 2012 validation dataset for single and multi-scale evaluation (see Methods). Increasing the number of iterations (“iter”) in the RNN improves model accuracy (blue curves). Additionally, a state-of-the-art fully convolutional network [9] is evaluated across a range of output strides (8 to 32). Importantly, recurrent segmentation achieves a higher quality segmentation at lower computational cost than [9] across a range of computational budgets.

(mIOU) between the ground truth labels and network predictions in Table 1. When run for many iterations, the RNN performs comparably to recently published, state-of-the-art networks [23, 72, 8, 76, 9]. The RNN outperforms on a subset of classes, but performs notably worse on specific classes that have high spatial frequency features, such as bicycle and chair. We discuss reasons for this behavior in Section 4.4 and the Discussion.

We additionally examined how the RNN performs with an increasingly strict computational budget. In this case, we examined the predictive performance of the network in terms of mIOU versus the number of unrolling steps in the RNN (Figure 6). The first iteration of the RNN leads to notable performance gain and these gains accrue as one iterates the network. Interestingly, if one runs the network for more steps than it was originally trained, the RNN predictions continue to improve (Figure 6, blue curves) as the canvas saturates.

We tested the same network on the Cityscapes [13] image segmentation dataset (Table 2). The performance of the network, trained only on the Cityscapes coarse dataset, was assessed using the mIOU as before – in addition, we calculated the instance-level intersection-over-union (iIOU), where the contribution of each pixel to the metric is weighted by the ratio of the class’ average instance size to the size of the ground truth instance. The RNN provided competitive results with respect to state-of-the-art methods in spite of the reduced computational budget.

Table 3. Computational budgets for RNN and state-of-the-art segmentation methods for single-pass and multi-pass evaluation for 513x513 images. Fully convolutional models [76, 9] may be evaluated across multiple output strides to reduce computational cost (but with reduced predictive precision). The RNN may be evaluated with an increasing number of iterations to improve predictive performance. Note that the RNN requires a consistently smaller computational budget.

4.2. Observations of error correction with recurrent networks

Given that an RNN performs identical operations on every iteration, a natural question is to better characterize the computational properties of this generic operation. We addressed this question by artificially perturbing the segmentation mask stored in the RNN canvas and examining how the RNN responded to these perturbations. When we seed the initial canvas to the wrong class–segmenting a horse as a cow, for instance–we find that the RNN is nonetheless able to correct the semantic segmentation in just two iterations (Figure 7, bottom row). After a few more iterations, the segmentation mask from the perturbed RNN is of similar quality to the RNN initialized with a canvas of all zeros, highlighting the robustness of the method to an incorrect or poor initial segmentation (Figure 7, top row).

The RNN’s ability to iteratively improve partial segmentations (Figure 2) and correct segmentation errors (Figure 7) suggests a natural extension to segmenting video. In this setting, consecutive frames are highly correlated and segmentations from previous frames provide a good starting point for subsequent frames.

4.3. Leveraging recurrent networks for efficient video segmentation

Temporal correlations between video frames have been exploited in video compression [44] and the propagation of weak labels across video frames [66, 34]. In the latter case, optical flow and motion tracking algorithms may learn signals for propagating label information across frames [49]. In the case of semantic video segmentation, we ask whether the structure of the RNN provides a computation-
Figure 7. Error correction through RNN dynamics. When the canvas is initialized as zeros, the RNN correctly segments a horse after a single iteration (top row). When the canvas is incorrectly initialized as a cow, the RNN nonetheless corrects its prediction after two iterations (bottom row).

Figure 8. A brief overview and quantification of efficient video segmentation with RNN. Every twentieth frame from a 120 fps video (first row) is segmented using two or six iterations of the RNN (second and third rows, respectively). In the last row of images, the RNN canvases at Frames 20, 40, 60, and 80 are initialized to the logits from the sixth iteration of Frame 0. Bottom plot quantifies pixel accuracy from either initializing the logits to zero (orange) or from the logits of Frame 0 (blue), and running the RNN for two iterations. Pixel accuracy is relative to the segmentation mask at the sixth iteration.

Section 4.1 indicates that subsequent iterations of the RNN refine previous segmentations. A natural question to ask is if the RNN may improve the segmentation provided from a previous frame with only a few iterations, thereby saving significant computational resources. Figures 1 and 8 provide an example pipeline for segmenting recorded video from public domain videos and a mobile device recording, respectively. The first video frame is segmented with the RNN as described above. In subsequent video frames, we seed the RNN canvas with the segmentation from a previous video frame and merely run the RNN for a small number of iterations (two). The resulting segmentation looks comparable to having run the RNN for a full number of iterations (six), although the computational demand is significantly reduced as the RNN is only run for two iterations. In Figure 1 we demonstrate that this video segmentation technique works qualitatively comparable to [8] across an array of public domain videos in which ground truth is unknown, at half the cost (158 GFLOPS/frame for \( RNN_{iter=2} \) vs. 310 GFLOPS/frame for DeepLab V3 [9], on a 1080 \( \times \) 608 video).

We quantified how well this procedure worked relative to running the RNN for the full six iterations. We calculated the pixel accuracy of the second RNN iteration with respect to the segmentation mask produced at the sixth iteration. When initialized independently for each frame, the accuracy of the second iteration drops dramatically relative to the sixth iteration (Figure 8, orange line). However, when seeded with the segmentation from the first frame, the second iteration of the RNN is sufficient to maintain a highly accurate segmentation mask even 80 frames (667 ms) after the segmentation mask used for seeding (Figure 8, blue line). When seeding the RNN with a computationally expensive segmentation from a previous frame, the RNN is able to identify small features, like the boats in the background of Figure 8, much more rapidly. Using a seed for the recurrent segmentation also helps maintain the semantic consistency of an object. When flying hair occludes the face in frame 40 of Figure 8, the model incorrectly classifies the edges of the hair as a bird, perhaps due to its feather-like appearance. However when using a previous frame’s segmentation, the seeded RNN does not make this misclassification.
4.4. Diagnosing failures of the recurrent network

We found that despite setting a new state-of-the-art in several categories of the PASCAL VOC 2012 dataset (Table 1), the performance on several classes—in particular, classes with fine-grained features—was poor. To investigate the capacity of the recurrent neural network to learn fine-grained features, we initialized the RNN canvas to the ground truth label of a bicycle with the frame, rims, and wheel spokes all individually segmented (Figure 9, bottom row). When the RNN is allowed to successively iterate on the already correct canvas, the RNN nonetheless progressively fills in the empty areas of the segmentation mask with each iteration, covering a convex hull of the object. The final segmentation mask closely resembles the low spatial frequency mask discovered when the RNN evolves from a blank canvas (Figure 9, top row).

To quantify the predominance of low spatial frequencies in recurrent segmentation, we computed the spatial power spectral density of the RNN predictions, the ground truth labels, and the difference between the two (Figure 10). We find that the residuals between the RNN predictions and ground truth labels have accentuated higher frequencies relative to the ground truth and RNN power spectra (Figure 10, red line). This suggests that the mistakes of the RNN occur at high spatial frequencies.

5. Discussion

We propose a recurrent neural network architecture for semantic segmentation that enables variable quality segmentation across computational budgets. This yields a natural framework for segmenting videos, resulting in a high quality segmentation of video frames at a fraction of the computational cost previously feasible.

The RNN model achieves accuracy comparable to state-of-the-art models but with reduced computational demand. We note however that RNN dynamics tend to fill in the interior of objects with a single semantic label. To address this, it would be possible to target the fine-grained features in these difficult images by bootstrapping [9] or by explicitly modifying the loss function to penalize object boundaries.

Due to memory constraints during training, the RNN processes a relatively coarse spatial resolution due to the output stride applied to the image features. While it is remarkable that the RNN achieves good performance with such coarse-grained features, we anticipate gains in GPU hardware memory will permit increased spatial resolution in convolutional RNN architectures and thus allow for more capacity in the network. In the interim, switching from an LSTM to other RNN architectures with fewer parameters may alleviate some of these memory limitations during training [29]. Another direction may be to relax the strict requirement that every iteration of the RNN must perform an identical computation. One method for approaching this is to build a hyper-network [32] whereby a second network predicts the network weights for each iteration. Another approach is to instead build an RNN architecture that adaptively updates the number of iterations [30] in order to achieve a given accuracy level.

The intersection of computational constraints, predictive power, and memory limitations necessitate a diversity of architectural approaches for image and video segmentation. Here we provide one such method, a recurrent architecture that trades off speed and accuracy for semantic segmentation. Further developing novel architectures for classic computer vision problems remains fertile ground for future research.
Acknowledgements

We thank Kevin Murphy and George Papandreou for comments on the manuscript, Liang-Chieh Chen for helpful comments about running Deeplab V3, and discussions with George Toderici and Ben Poole.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467, 2015.

[2] A. Angelova, A. Krizhevsky, and V. Vanhoucke. Pedestrian detection with a large-field-of-view deep network. In Robotics and Automation (ICRA), 2015 IEEE International Conference on, pages 704–711. IEEE, 2015.

[3] J. Ba, V. Mnih, and K. Kavukcuoglu. Multiple object recognition with visual attention. In International Conference of Learned Representations (ICLR), 2015.

[4] M. Bai and R. Urtasun. Deep watershed transform for instance segmentation. In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[5] J. T. Barron, A. Adams, Y. Shih, and C. Hernández. Fast bilateral-space stereo for synthetic defocus. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4466–4474, 2015.

[6] J. T. Barron and B. Poole. The fast bilateral solver. In European Conference on Computer Vision, pages 617–632. Springer, 2016.

[7] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firmer, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba. End to end learning for self-driving cars. CoRR, abs/1604.07316, 2016.

[8] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected cnns. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017.

[9] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Re-thinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587, 2017.

[10] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. arXiv preprint arXiv:1512.01274, 2015.

[11] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer. cudnn: Efficient primitives for deep learning. arXiv preprint arXiv:1410.0759, 2014.

[12] J. Collins, J. Sohl-Dickstein, and D. Sussillo. Capacity and trainability in recurrent neural networks. In International Conference of Learned Representations (ICLR), 2016.

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

[14] J. Dai, K. He, and J. Sun. Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In Proceedings of the IEEE International Conference on Computer Vision, pages 1635–1643, 2015.

[15] J. Dai, K. He, and J. Sun. Convolutional feature masking for joint object and stuff segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3992–4000, 2015.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2009.

[17] E. L. Denton, S. Chintala, R. Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In Advances in neural information processing systems, pages 1486–1494, 2015.

[18] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In International Conference on Machine Learning, volume 32, pages 647–655, 2014.

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

[20] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 27, pages 2366–2374. Curran Associates, Inc., 2014.

[21] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2014.

[22] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. International Journal of Computer Vision, 88(2):303–338, June 2010.

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

[24] G. Ghiassi and C. C. Fowlkes. Laplacian pyramid reconstruction and refinement for semantic segmentation. In European Conference on Computer Vision, pages 519–534. Springer, 2016.

[25] R. Girshick. Fast r-cnn. In The IEEE International Conference on Computer Vision (ICCV), December 2015.
segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 1

[27] A. Giusti, D. C. Ciresan, J. Masci, L. M. Gambardella, and J. Schmidhuber. Fast image scanning with deep max-pooling convolutional neural networks. In *Image Processing (ICIP), 2013 20th IEEE International Conference on*, pages 4034–4038. IEEE, 2013. 3

[28] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010. 12

[29] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016. 1, 8

[30] A. Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016. 8

[31] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra. Draw: A recurrent neural network for image generation. In *International Conference on Machine Learning*, 2015. 1, 3, 4

[32] D. Ha, A. Dai, and Q. V. Le. Hypernetworks. In *International Conference on Learning Representations*, 2017. 8

[33] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 447–456, 2015. 2

[34] G. Hartmann, M. Grundmann, J. Hoffman, D. Tsai, V. Kwastra, O. Madani, S. Vijayanarasimhan, I. Essa, J. Rehg, and R. Sukthankar. Weakly supervised learning of object segmentations from web-scale video. In *Computer Vision–ECCV 2012. Workshops and Demonstrations*, pages 198–208. Springer, 2012. 6

[35] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3, 4

[36] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 3

[37] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 3

[38] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian. A real-time algorithm for signal analysis with the help of the wavelet transform. In *Wavelets*, pages 286–297. Springer, 1990. 3

[39] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 1, 4

[40] M.-C. Hu, Y.-N. Tseng, J.-L. Wu, C.-H. Kuo, Y.-M. Hsiao, and K.-L. Hua. Real-time depth of field rendering with bokeh effect. In *Consumer Electronics (ISCE), 2013 IEEE 17th International Symposium on*, pages 99–100. IEEE, 2013. 1

[41] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, et al. In-datacenter performance analysis of a tensor processing unit. *arXiv preprint arXiv:1704.04760*, 2017. 1

[42] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing System*, 2012. 1

[43] J. Lafferty, A. McCallum, and F. C. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001. 3

[44] D. Le Gall. Mpeg: A video compression standard for multimedia applications. *Communications of the ACM*, 34(4):46–58, 1991. 6

[45] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. 1

[46] A. Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016. 8

[47] G. Lin, C. Shen, A. van den Hengel, and I. Reid. Efficient piecewise training of deep structured models for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3194–3203, 2016. 14

[48] C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):978–994, 2011. 6

[49] W. Liu, A. Rabinovich, and A. C. Berg. Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*, 2015. 14

[50] Z. Liu, X. Li, P. Luo, C.-C. Loy, and X. Tang. Semantic image segmentation via deep parsing network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1377–1385, 2015. 3

[51] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015. 1, 5, 14

[52] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich. Feedforward semantic segmentation with zoom-out features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3376–3385, 2015. 14

[53] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528, 2015. 3

[54] G. Papandreou, I. Kokkinos, and P.-A. Savalle. Modeling local and global deformations in deep learning: Epitomic convolution, multiple instance learning, and sliding window detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 390–399, 2015. 3

[55] P. Pinheiro and R. Collobert. Recurrent convolutional neural networks for scene labeling. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 82–90, Beijing, China, 22–24 Jun 2014. PMLR. 3
[57] M. Ren and R. S. Zemel. End-to-end instance segmentation and counting with recurrent attention. In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017. 1

[58] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In Advances in Neural Information Processing Systems, pages 91–99, 2015. 1

[59] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical Image Computing and Computer-Assisted Intervention, pages 234–241. Springer, 2015. 1, 3

[60] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229, 2013. 3

[61] P. Sermanet, A. Frome, and E. Real. Attention for fine-grained categorization. In International Conference of Learned Representations (ICLR), 2015. 1

[62] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In International Conference on Learning Representations, 2015. 1

[63] S. Sukhbaatar, J. Weston, R. Fergus, et al. End-to-end memory networks. In Advances in neural information processing systems, pages 2440–2448, 2015. 3

[64] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In IEEE Conference on Computer Vision and Pattern Recognition, 2015. 1

[65] C. Szegedy, A. Toshev, and D. Erhan. Deep neural networks for object detection. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 26, pages 2553–2561. Curran Associates, Inc., 2013. 1

[66] K. Tang, R. Sukthankar, J. Yagnik, and L. Fei-Fei. Discriminative segment annotation in weakly labeled video. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2483–2490, 2013. 6

[67] G. Toderici, S. M. O’Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar. Variable rate image compression with recurrent neural networks. In International Conference of Learned Representations (ICLR), 2016. 1

[68] A. Viebke and S. Plana. The potential of the intel (r) xeon phi for supervised deep learning. In High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), 2015 IEEE 12th International Conference on Embedded Software and Systems (ICCESS), 2015 IEEE 17th International Conference on, pages 758–765. IEEE, 2015. 1

[69] P. Warden. Running tensorflow graphs on microcontrollers, 2017. 1

[70] W. Williams, N. Prasad, D. Mrva, T. Ash, and T. Robinson. Scaling recurrent neural network language models. In Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on, pages 5391–5395. IEEE, 2015. 1

[71] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google’s neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144, 2016. 4

[72] Z. Wu, C. Shen, and A. v. d. Hengel. Bridging category-level and instance-level semantic image segmentation. arXiv preprint arXiv:1605.06885, 2016. 6, 14

[73] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In Advances in neural information processing systems, pages 802–810, 2015. 3, 13

[74] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122, 2015. 14

[75] X. Zhang, X. Zhou, L. Mengxiao, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. arXiv preprint arXiv:1707.01083, 2017. 1

[76] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. arXiv preprint arXiv:1612.01105, 2016. 1, 2, 5, 6, 13, 14

[77] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr. Conditional random fields as recurrent neural networks. In Proceedings of the IEEE International Conference on Computer Vision, pages 1529–1537, 2015. 3, 5, 14

[78] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. CoRR, abs/1707.07012, 2017. 1
## Appendix

### Hyperparameters

| Operation            | Kernel size | Stride | Feature maps | Padding |
|----------------------|-------------|--------|--------------|---------|
| Network – 513 × 513 × 3 input |             |        |              |         |
| ResNet-v1-101        |             |        |              |         |
| Convolutional LSTM 1 | 1           | 1      | 512          | SAME    |
| Convolutional LSTM 1 | 1           | 1      | 256          | SAME    |
| Convolutional LSTM 1 | 1           | 1      | |classes| | SAME |
| Canvas (add:0)       |             |        | |classes| |         |
| Bilinear upsampling  |             |        |              |         |
| Padding mode         | Zeros       |        |              |         |
| Normalization        | Batch normalization after every ResNet convolution | | | |
| Optimizer            | SGD with Momentum (momentum = 0.95) | | | |
| Parameter updates    | 30,000      |        |              |         |
| Learning rate schedule | $0.001 - 1e^{-6} \cdot (1 - \text{step}/1000)^{0.9} + 1e^{-6}$ | | | |
| Batch size           | 16          |        |              |         |
| Weight initialization | Glorot normal [28] | | | |

Table 4. Details of the recurrent network architecture for image segmentation. |classes| is 21 for the PASCAL VOC 2012 semantic segmentation dataset, and 19 for the Cityscapes dataset. The final block4 of the ResNet-v1-101 was augmented with dilation rates of (2, 4, 8) in the three units of block4, following [9].
Supplemental methods

Convolutional LSTM

For the recurrent network architecture, we employed stacked convolutional LSTM layers [73] defined as

\[ i_t = \sigma \left( W_{ih}^i \ast h_{t-1} + W_{ix}^i \ast x_t + b_i^i \right) \]  
\[ f_t = \sigma \left( W_{fh}^f \ast h_{t-1} + W_{fx}^f \ast x_t + b_f^f + b_{fg} \right) \]  
\[ c_{t}^{in} = \tanh \left( W_{ch}^c \ast h_{t-1} + W_{cx}^c \ast x_t + b_c^c \right) \]  
\[ c_t = f_t \cdot c_{t-1} + i_t \cdot c_{t}^{in} \]  
\[ o_t = \sigma \left( W_{oh}^o \ast h_{t-1} + W_{ox}^o \ast x_t + b_o^o \right) \]  
\[ h_t = o_t \cdot \tanh(c_t), \]

where \( \sigma(\cdot) \) is the logistic function, \( \ast \) is the convolution operator, where \( i, f, o \) represent the input, forget and output gates, respectively, and \( c \) and \( h \) are the state of the LSTM.

Forget gate offset bias \( b_{fg} = 1.0 \).

Training details

We found that the best performance was achieved by having a batch-size of 12 - 16 and a relatively large ResNet output stride of 16.

We also found that the crop-size had a significant effect on performance, with the highest performance achieved with keeping the crop-size of the input image as large as the native resolution - 513 × 513 for PASCAL VOC images and 1025 × 2049 for Cityscapes images. In each case the crop size is an integer divisible by 32, plus one, in order to avoid edge effects with the ResNet output stride.

Estimating computational cost

We used the Tensorflow profiler (tf.profiler.Profiler) to estimate FLOPS during evaluation of the models. We also constructed Tensorflow models of the Pyramid Scene Parsing network [76] and Deeplab V3 [9], following the methods reported as closely as possible. From these models we used Tensorflow profiling as before to estimate the FLOPS for these models. Note however that all performance numbers for both models are taken from the values reported in the original papers.
| Method          | aero | bike | bird | boat | bottle | bus | car | cat | chair | cow | table | dog | horse | mbike | person | plant | sheep | sofa | train | tv | mIOU |
|----------------|------|------|------|------|--------|-----|-----|-----|-------|-----|-------|-----|-------|-------|--------|-------|-------|------|-------|----|------|
| FCN [52]      | 76.8 | 34.2 | 68.9 | 49.4 | 60.3   | 75.3 | 74.7 | 77.6 | 21.4  | 62.5 | 46.8  | 71.8 | 63.9  | 76.5  | 73.9   | 45.2  | 72.4  | 37.4 | 70.9  | 55.1 | 62.2 |
| Zoom-out [53]  | 85.6 | 37.3 | 83.2 | 62.5 | 66.0   | 85.1 | 80.7 | 84.9 | 27.2  | 73.2 | 57.5  | 78.1 | 79.2  | 81.1  | 77.1   | 53.6 | 74.0  | 49.2 | 71.7  | 63.3 | 69.6 |
| CRF-RNN † [77] | 90.4 | 55.3 | 88.7 | 68.4 | 69.8   | 88.3 | 82.4 | 85.1 | 32.6  | 78.5 | 64.4  | 79.6 | 81.9  | 86.4  | 81.8   | 58.6 | 82.4  | 53.5 | 77.4  | 70.1 | 74.7 |
| BoxSup † [14]  | 89.8 | 38.0 | 89.2 | 68.9 | 68.0   | 89.6 | 83.0 | 87.7 | 34.4  | 83.6 | 67.1  | 81.5 | 83.7  | 85.2  | 83.5   | 58.6 | 84.9  | 55.8 | 81.2  | 70.7 | 75.2 |
| Dilatation8 † [74] | 91.7 | 39.6 | 87.8 | 63.1 | 71.8   | 89.7 | 82.9 | 89.8 | 37.2  | 84.0 | 63.0  | 83.3 | 89.0  | 83.8  | 85.1   | 56.8 | 87.6  | 56.0 | 80.2  | 64.7 | 75.3 |
| DPN † [50]     | 89.0 | 61.6 | 87.7 | 66.8 | 74.7   | 91.2 | 84.3 | 87.6 | 36.5  | 86.3 | 66.1  | 84.4 | 87.8  | 85.6  | 85.4   | 63.6 | 87.3  | 61.3 | 79.4  | 66.4 | 77.5 |
| Piecewise † [48] | 94.1 | 40.7 | 84.1 | 67.8 | 75.9   | 93.4 | 84.3 | 88.4 | 42.5  | 86.4 | 64.7  | 85.4 | 89.0  | 85.8  | 86.0   | 76.7 | 90.2  | 63.8 | 80.9  | 73.0 | 78.0 |
| FCRNs † [72]   | 91.9 | 48.1 | 93.4 | 69.3 | 75.5   | 94.2 | 87.5 | 92.8 | 36.7  | 86.9 | 65.2  | 89.1 | 90.2  | 86.5  | 87.2   | 64.6 | 90.1  | 59.7 | 85.5  | 72.7 | 79.1 |
| LRR † [24]     | 92.4 | 45.1 | 94.6 | 65.2 | 75.8   | 95.1 | 89.1 | 92.3 | 39.0  | 85.7 | 70.4  | 88.6 | 89.4  | 88.6  | 86.6   | 65.8 | 86.2  | 57.4 | 85.7  | 73.3 | 79.3 |
| DeepLab † [8]  | 92.6 | 60.4 | 91.6 | 63.4 | 76.3   | 95.0 | 88.4 | 92.6 | 32.7  | 88.5 | 67.6  | 89.6 | 92.1  | 87.0  | 87.4   | 63.3 | 88.3  | 60.0 | 86.8  | 74.5 | 79.7 |
| PSPNet † [76]  | 95.8 | 72.7 | 95.0 | 78.9 | 84.4   | 94.7 | 92.0 | 95.7 | 43.1  | 91.0 | 80.3  | 91.3 | 96.3  | 92.3  | 90.1   | 71.5 | 94.4  | 66.9 | 88.8  | 82.0 | 85.4 |
| DeepLabv3 † [9] | 96.4 | 76.6 | 92.7 | 77.8 | 87.6   | 96.7 | 90.2 | 95.4 | 47.5  | 93.4 | 76.3  | 91.4 | 97.2  | 91.0  | 92.1   | 71.3 | 90.9  | 68.9 | 90.8  | 79.3 | 85.7 |
| SDN+ † [23]    | 96.9 | 78.6 | 96.0 | 79.6 | 84.1   | 97.1 | 91.9 | 96.6 | 48.5  | 94.3 | 78.9  | 93.6 | 95.5  | 92.1  | 91.1   | 75.0 | 93.8  | 64.8 | 89.0  | 84.6 | 86.6 |
| RNN † [iteration 6] | 94.9 | 47.3 | 94.7 | 72.5 | 81.1   | 95.8 | 93.2 | 93.8 | 44.9  | 93.0 | 65.4  | 89.0 | 94.3  | 92.9  | 89.9   | 76.6 | 93.8  | 62.4 | 89.9  | 78.9 | 82.9 |

Table 5. Per-class intersection-over-union results on the PASCAL VOC 2012 test dataset. Methods using datasets in addition to PASCAL VOC 2012 are marked with ‘†’. Last row corresponds to the semantic segmentation at the sixth iteration of the recurrent neural network presented in this paper.

Figure 11. Qualitative comparison of recurrent semantic segmentation masks to feed-forward semantic segmentation models on the PASCAL VOC 2012 validation dataset. Here we compare the RNN segmentation on the sixth iteration with segmentation masks from the FCN [52], DPN [50], DeepLab [8], and PSPNet [76] architectures.