Speeding up Semantic Segmentation for Autonomous Driving

Conference Paper · December 2016

CITATIONS
277

READS
8,217

12 authors, including:

Jose Antonio Arjona-Medina
Dynatrace
20 PUBLICATIONS **524** CITATIONS
SEE PROFILE

Thomas Unterthiner
Google Inc.
36 PUBLICATIONS **19,416** CITATIONS
SEE PROFILE

Andreas Mayr
Johannes Kepler University Linz
47 PUBLICATIONS **4,117** CITATIONS
SEE PROFILE

Martin Heusel
Zalando SE
9 PUBLICATIONS **9,908** CITATIONS
SEE PROFILE

Speeding up Semantic Segmentation for Autonomous Driving

Michael Treml [*1], **José Arjona-Medina**[*1], **Thomas Unterthiner**[*1],
Rupesh Durgesh[2], **Felix Friedmann**[2], **Peter Schuberth**[2],
Andreas Mayr[1], **Martin Heusel**[1], **Markus Hofmarcher**[1], **Michael Widrich**[1],
Ulrich Bodenhofer[1], **Bernhard Nessler**[1], **Sepp Hochreiter**[1]
[1] Institute of Bioinformatics, Johannes Kepler University Linz, Austria
[2] Audi Electronics Venture GmbH, Germany

{treml, arjona, unterthiner, nessler, hochreit}@bioinf.jku.at
{rupesh.durgesh, felix.friedmann, peter.schuberth}@audi.de

Abstract

Deep learning has considerably improved semantic image segmentation. However, its high accuracy is traded against larger computational costs which makes it unsuitable for embedded devices in self-driving cars. We propose a novel deep network architecture for image segmentation that keeps the high accuracy while being efficient enough for embedded devices. The architecture consists of ELU activation functions, a SqueezeNet-like encoder, followed by parallel dilated convolutions, and a decoder with SharpMask-like refinement modules. On the Cityscapes dataset, the new network achieves higher segmentation accuracy than other networks that are tailored to embedded devices. Simultaneously the frame-rate is still sufficiently high for the deployment in autonomous vehicles.

1 Introduction

Modern camera systems can produce high quality images at high rates at very low costs, allowing them to be placed in many commodity items ranging from mobile phones to surveillance systems and automotive vehicles. This increases the demand for systems that are able to understand this data. The advent of deep learning enabled great strides towards better visual understanding [19, 25]. Deep neural networks have achieved super-human performance on tasks like image classification [24, 9] or traffic sign recognition [3]. However, this performance boost was accomplished by increasing the network size [10], which means that deep networks incur large computational costs. These large networks are infeasible or at least difficult to employ them in embedded devices as found in self-driving cars.

The first step in many autonomous driving systems based on visual inputs is object recognition, object localization, and, in particular, semantic segmentation. The goal of semantic segmentation is to label each pixel in an image as belonging to a given semantic class. In a typical urban scene, these classes could be street, traffic signs, street markings, cars, pedestrians, or sidewalks. When employing deep learning for semantic segmentation, the recognition of major objects in the image such as persons or vehicles is realized at higher levels of a neural network. By design, these layers work on a coarser scale and are translation invariant (e.g. imposed via pooling operations), such that minor variations on a pixel level do not influence the recognition. Furthermore, semantic segmentation requires pixel-exact classification of small features, which are typically only found in lower layers

[*]These authors contributed equally to this work.

29th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

of a network. This trade-off in resolution is typically solved by using skip-connections from lower layers to the output which increase the resolution at layers close to the output. Skip-connection were introduced by the The Fully Convolutional Network (FCN) [21], which still serves as a blue-print for most modern approaches. These approaches only differ in how they encode the object level information and how they decode these classifications to pixel-exact labels. The original FCN employed the VGG network [26] that was pre-trained on the LSVRC image classification task. Then FCN added information to higher layers coming from lower layers which is upscale through a transposed convolution. The FCN architecture was improved by alternative ways to connect to the lower layers, e.g. by accessing the lower-layer pooling layers [2], by using enhanced methods to integrate lower level information [23] or forgoing pooling operations for dilated convolutions [20, 27]. As a post-processing step, many recent systems apply CRF-based refinement on the output produced by the neural network [20, 28]. CRF increases the accuracy of the segmentation at the cost of additional computation.

Reducing the computational burden of semantic segmentation is essential to make it feasible for embedded systems and autonomous driving. Neural networks are trained on servers or workstations with powerful GPUs, and these GPU systems are subsequently used for inference on new data. However, these commodities do not exist in self-driving cars. A self-driving car needs to react to new events instantly to guarantee the safety of passengers and other traffic participants, while it is often acceptable if the borders of objects are not recognized perfectly down to a pixel resolution. To segment an image in real-time is a strong requirement in self-driving applications. Thus, it is critical that any convolutional neural network deployed in these systems fulfills strict requirements in execution speed.

There has been a vast amount of research in reducing the computation required for deep learning. SqueezeNet [15] showed that it was possible to reproduce the image classification accuracy of powerful CNNs such as AlexNet [17] using 50x less parameters by employing a more efficient architecture. ENet [22] followed the same path and showed that semantic segmentation is feasible on embedded devices in real-time. Another line of research increases the efficiency of existing networks by deriving smaller networks from larger counterparts [1, 11], by pruning or quantizing weights [6, 8] or tweaking the network for execution on specific hardware designs [7]. These methods can be applied on top of new architectures to speed up execution.

In this work, we propose novel architectures that are able to do inference in real-time on embedded hardware, while achieving greater accuracy than competitors like ENet. The new architecture is benchmarked and tested on the Cityscapes dataset for semantic urban scene understanding [5].

2 Methods

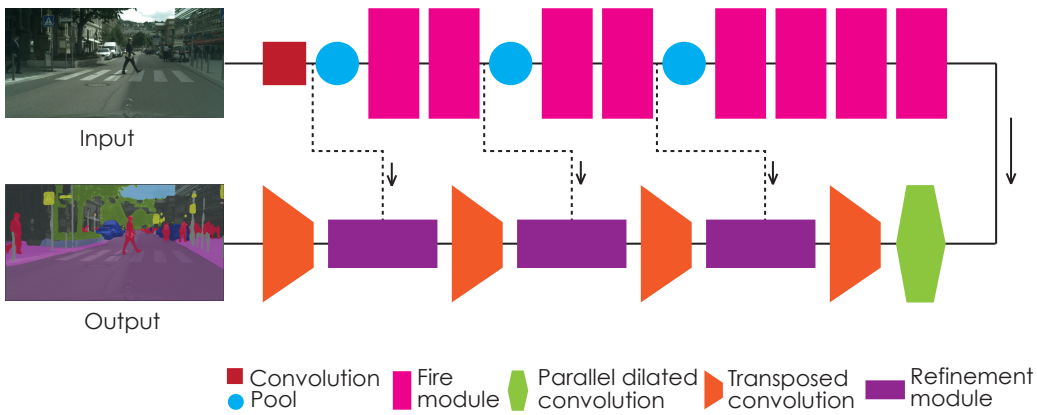

Figure 1: Architecture of the proposed network for semantic segmentation.

2.1 Overview

In order to achieve semantic segmentation in real time, we have to trade execution speed against achievable segmentation accuracy. Like most successful segmentation networks, our network is structured as an encoder-decoder pair. An encoder CNN detects higher-level objects such as cars or pedestrians in the input image. A decoder takes this information and enriches it with information from the lower layers of the encoder, supplying a prediction for each pixel in the original input. Figure 1 depicts the architecture.

2.2 Encoder

The encoder is a modified SqueezeNet 1.1 architecture [15], which was designed as a low-latency network for image recognition while retaining AlexNet [17] like accuracy. The main computational modules of SqueezeNet are the so-called "fire" modules consisting of three convolutional operations, depicted in Figure 2a. The encoder consists of eight "fire" modules, interspersed with a total of three max-pooling layers for downsampling. All rectified linear units (ReLUs) of the original architecture are substituted with exponential linear units (ELUs) [4], which make more efficient use of parameters by also conveying information in the negative part of the activation.

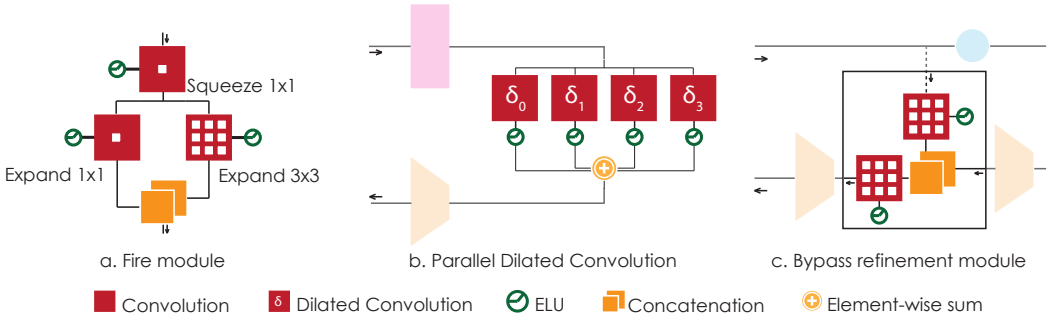

Figure 2: a) SqueezeNet Fire module; b) Parallel dilated convolution layer; c) Refinement module

2.3 Parallel Dilated Convolutions

The decoder is based on a parallel dilated convolution layer [20] as depicted in Figure 2b. This dilated layer combines the feature maps at the encoder output at different receptive field sizes by using four dilated convolutions of kernel size 3 with different dilation factors. This is equivalent to sampling the layer input with different rates. The contributions from the four dilated convolutions are then fused by an element-wise sum. As a result, the receptive field size is increased and multiscale spatial dependencies are taken into account without having to resort to fully connected layers which would be computationally infeasible. Thus, the decoder can be realized by considerably less parameters while the high performance is kept.

2.4 Decoder and Bypasses

Pooling layers in the encoder are used to ensure a degree of translational invariance when detecting the parts of an object. However, they in turn reduce the spatial resolution of the output. Transposed convolutions in the decoder are used to upsample the information back to its original size. To improve the upsampling, we don't just use the data that comes directly from the layer below the transposed convolution layer, but combine it with low-level knowledge from lower layers of the encoder. These layers are responsible for detecting finer structures at a higher resolution, which helps with classifying the contours of objects more exactly. Each refinement module combines two streams of information, one coming from the previous upsampling layer, the other one from the encoder. The two convolutional layers in the refinement module learn how to weigh these two streams before passing the information on to the next upsampling layer. We use refinement modules similar to the ones used in the SharpMask approach[23]. We again use ELUs instead of ReLU units (Figure 2c shows the implementation of the module).

Right before every pooling layer in the encoder, a bypass branches off to the refinement module. Once there, a convolution layer weights knowledge from lower layers. Then, it is concatenated with semantic object information from the previous upsampling layer. A second convolutional layer combines the concatenated feature maps from both branches into the class map.

2.5 Exponential Linear Units (ELU)

Our network makes extensive use of the exponential linear unit [4]. ELUs were designed to avoid the bias shift in neural network training. The ELU activation function is defined as

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\left(\exp(x) - 1\right) & \text{if } x \leq 0 \end{cases}, \quad f'(x) = \begin{cases} 1 & \text{if } x > 0 \\ f(x) + \alpha & \text{if } x \leq 0 \end{cases}. \tag{1}$$

The parameter α was set to its default value of 1 for all of our experiments. Similar to the often-used ReLU activation, ELUs have a linear positive part which helps to avoid the vanishing gradient [12, 13], thus it allows training very deep networks. However, in contrast to the ReLU, the saturating negative part converges to $-\alpha$. This allows the ELU to have a mean activation of 0, thereby avoiding any bias shift effect: In ReLU networks, units will typically have a non-zero mean activation, thus they will act as additional bias unit for units in the next layer. By enabling units to have zero mean, this bias shift effect is reduced, which makes it easier for units to focus solely on actual information processing. This could otherwise only be achieved by using batch normalization [16], which would increase the computational cost of the network by adding specific layers to perform this operation.

3 Experiments

3.1 Cityscapes dataset

We trained and evaluated the network on the Cityscapes dataset [5]. Cityscapes is a high quality dataset for semantic street scene understanding. The dataset is split in 2975, 500, and 1525 images of resolution 2048x1024 pixels for training, validation, and testing, respectively. It contains 35 annotated classes on pixel-level of which 19 are selected for evaluation in the challenge. Each class belongs to one category: flat, nature, object, sky, construction, human, vehicle. As performance measure the commonly used intersection over union metric is used, which is evaluated for individual classes and categories. Notice that our results were achieved without CRFs as post-processing because that would increase inference time dramatically.

3.2 Training

The SqueezeNet encoder was initialized using publicly available ImageNet pre-trained weights and fine-tuned for semantic segmentation on Cityscapes. The rest of the weights were initialized using the MSRA scheme [9] and the network was trained end-to-end at once with full resolution images. The loss function was the sum of cross entropy terms for all classes, equally weighted in the overall loss function. It was optimized with Stochastic Gradient Descent, using a fixed learning rate of 10^{-8}, momentum of 0.9 and a weight decay factor of 0.0002. The architecture was implemented using the Caffe framework. Total training time was around 22 hours using 2 Titan X Maxwell GPUs with batch size 3 each at full resolution images. In our experiments we trained for the pixel-wise segmentation task using only the fine annotations without any additional training data. Augmentations were not applied in order to establish a baseline for performance.

4 Results

Our network compares favourably against the ENet segmentation network [22], which to the best of our knowledge is the only other network designed for semantic segmentation for low-latency devices.

4.1 Segmentation performance

We evaluated the network on the test set using the official Cityscapes evaluation server. We achieve 59.8 per-class mean IoU and 84.3 per-category mean IoU. Hence the network architecture is able to

outperform both ENet as well as SegNet [2] as it can be seen in Table 1. We improved on per-class IoU for all but 5 classes (wall, truck, bus, train, motorcycle) compared to ENet. Detailed per-class classification results are presented in Table 2. Visual inspection of the predictions of the network show satisfying results on typical urban street scene images (Figure 3). Object contours are segmented very sharply. We believe that this is due to the enhanced ability to integrate pixel-level information from early layers in the encoder into upsampling layers in the decoder by using refinement modules in the bypasses.

Table 1: Mean IoU over the 19 individual classes and the 7 categories on the Cityscapes testset. ENet and SegNet results are taken from Paszke et al. [22].

	Class IoU	Category IoU
Ours	**59.8**	**84.3**
ENet	58.3	80.4
SegNet	56.1	79.8

Table 2: Per-class IoU on the Cityscapes testset. We improved the ENet results on all but 5 classes (wall, truck, bus, train, motorcycle). ENet results are taken from Paszke et al. [22].

	road	sidewalk	building	wall	fence	pole	trafficlight	trafficsign	vegetation	terrain
Ours	**96.9**	**75.4**	**87.9**	31.6	**35.7**	**50.9**	**52.0**	**61.7**	**90.9**	**65.8**
ENet	96.3	74.2	85.0	**32.2**	33.2	43.5	34.1	44.0	88.6	61.4

	sky	person	rider	car	truck	bus	train	motorcycle	bicycle
Ours	**93.0**	**73.8**	**42.6**	**91.5**	18.8	41.2	33.3	34.0	**59.9**
ENet	90.6	65.5	38.4	90.6	**36.9**	**50.5**	**48.1**	**38.8**	55.4

4.2 Inference Run-time Performance

Similar to ENet, we are able to surpass the 10 fps design goal on the Nvidia TX1 board on a resolution of 640x360, which is a sensible lower limit for enabling self-driving car applications. See Table 3 for a comparison of run-times between the different architectures. The run-times were achieved using CUDA 7.5 and cuDNN 4.0, however we expect that timings will improve significantly by switching to newer software versions that support Winograd convolutions [18].

Table 3: Comparison of inference times on Nvidia Tegra X1. Timings for ENet were taken from the original publication [22].

	480x320		640x360		1280x720	
	ms	fps	ms	fps	ms	fps
Ours	**60**	**16.7**	**86**	**11.6**	**389**	**2.6**
ENet	47	21.1	69	14.6	262	3.8

5 Conclusion

Deep Neural Networks are the current state of the art systems in semantic image segmentation. However, these deep networks incur a high computational cost, which makes them unsuitable for deployment in self-driving cars. In this work we investigated how neural networks can be made small enough to run in embedded devices used in autonomous vehicles. Our preliminary results

are based on an encoder-decoder approach that combines global object information obtained from SqueezeNet architecture with local high-resolution information using SharpMask-like refinement modules. This network is capable of running with 10+ fps on a Nvidia Jetson Tegra X1, while still yielding segmentation performance that is sufficient for self-driving car applications.

There are several possible ways to improve the current architecture at little to no additional run-time cost. For one, the encoder was currently initialized from a SqueezeNet that originally used ReLUs. As we replaced these with ELU units, the initialization is likely to be sub-optimal, as the weights implicitly assume an activation function that saturates at zero. Re-training the encoder from scratch is likely to improve our results without any effect on run-time. We furthermore have not yet looked into dataset augmentation, neither have we trained with the coarsely annotated images provided in the Cityscapes dataset. We also plan to add more residual connections between individual fire modules, which was shown to improve other architectures like ResNet [14]. With this change, we do expect to improve upon this architecture in the future. Furthermore it is still unclear if the network does need to be this deep, or if it is possible to remove further processing units, e.g. via Knowledge Distillation [11]. It would also be possible to replace the pooling units with dilated convolutions, which has been shown to improve segmentation performance in other designs [27]. We expect these improvements to allow us to be able to train better segmentation networks with even faster inference times in the future.

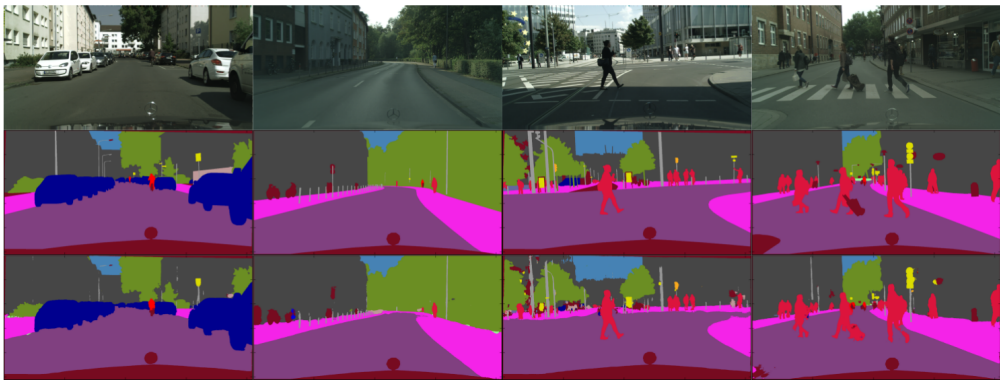

Figure 3: Example output (bottom) of the network with ground truth (center) on images from the Cityscapes validation set.

Acknowledgments

The authors gratefully acknowledge the Nvidia Corporation for supporting this research through hardware donations and thank Matthias Dorfer for the insightful discussions and comments.

References

[1] J. Ba and R. Caruana. Do deep nets really need to be deep? *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015.

[3] D. Ciresan, U. Meier, J. Masci, and J. Schmidhuber. Multi-column deep neural network for traffic sign classification. *Neural Networks*, 32:333–338, 2012.

[4] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). *International Conference on Learning Representations (ICLR)*, 2016.

[5] 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. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3213–3223, 2016.

[6] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1135–1143, 2015.

[7] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally. Eie: Efficient inference engine on compressed deep neural network. *International Conference on Computer Architecture (ISCA)*, 2016.

[8] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations (ICLR)*, 2016.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[11] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[12] S. Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. Master's thesis, Technische Universität München, Institut für Informatik, 1991.

[13] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In Kremer and Kolen, editors, *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press, 2001.

[14] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016.

[15] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size. *arXiv preprint arXiv:1602.07360*, 2016.

[16] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS) 25*, 2012.

[18] A. Lavin and S. Gray. Fast algorithms for convolutional neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

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

[20] C. Liang-Chieh, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *International Conference on Learning Representations (ICLR)*, 2015.

[21] 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 (CVPR)*, pages 3431–3440, 2015.

[22] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation. *arXiv preprint arXiv:1606.02147*, 2016.

[23] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[24] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[25] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. doi: 10.1016/j.neunet.2014.09.003.

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference of Learning Representations (ICLR)*, 2015.

[27] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *International Conference on Learning Representations (ICLR)*, 2016.

[28] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr. Conditional random fields as recurrent neural networks. In *International Conference on Computer Vision (ICCV)*, 2015.