Improving Dense Convolutional Neural Network Image Semantic Segmentation

Tony Peng
MIT 6.819
tonypeng@mit.edu

Michael Shumikhin
MIT 6.819
shn@mit.edu

Abstract

We apply the power of deep dense convolutional neural networks (DenseNets) to the problem of semantic image segmentation. DenseNets have been shown to achieve state-of-the-art performance on image segmentation. DenseNets consist of dense blocks, which consist of consecutive layers, each passing its output to all further layers in a feed-forward fashion. Empirically, this results in more efficient learning and higher accuracies.

In image segmentation, we attempt to classify each pixel in an image as one of a set of semantic categories. Semantic segmentation has a wide array of applications, including self-driving cars and medical imaging.

We first evaluate the Tiramisu DenseNet architecture, a model which has shown to produce state-of-the-art results on segmentation tasks. The Tiramisu architecture consists of a downsampling path which extracts features from the input followed by an upsampling path which then constructs the output image at the original resolution. Next, we apply incremental improvements to the baseline architecture to significantly boost the accuracy of the model. We train a model on the CamVid and ADE20K datasets with limited compute resources and report comparable accuracies. Extrapolating the data, we predict higher accuracies for equally-long-trained models. We provide code to reproduce our results here: `http://github.com/tonypeng/ez-image-segmentation`

1. Introduction

In recent years convolutional neural networks (CNNs) have greatly outperformed traditional computer vision algorithms on image classification tasks. In the past decade, the literature has largely trended towards deeper (more layers) networks to achieve greater accuracy and representational power. This follows from the the observation that neurons in further layers have increased receptive fields and the intuition that each layer learns more abstract features based on prior layers, and thus deeper networks can learn more abstract classifiers. A typical classification CNN architecture consists of multiple *convolutional* and *pooling* layers ending in a classification output [11]. Convolutional filters apply learned filters to the input image, and pooling layers reduce the spatial dimensionality of the intermediate feature maps, reducing parameter count and resolution.

We now consider the problem of *semantic image segmentation*, in which we seek to classify each pixel in an image as semantic category. Early approaches involved *patch-based per-pixel methods*, in which a individual patches of an image are passed into a CNN to classify the center pixel of the patch. As expected, this method is very slow, as one pass through the CNN is performed for each pixel of the image [6]. Instead, the literature has now moved on to *dense classification methods* in which a CNN classifies all the pixels of the input image in one pass, outputting an image map with per-pixel predictions.

One such method is the *encoder-decoder* architecture, in which case the output of a traditional, down-sampling CNN is fed into the input of reversed, up-sampling CNN. The *SegNet* is such an implementation, which is composed of two back-to-back VGG-16 models and has achieved considerable accuracy [1][13]. Unfortunately, such models can take a long time to train, on the order of weeks.

2. Related Work

The use of residual connections has enabled the training of much deeper models by introducing shortcut connections between layers [8]. These short-cuts are added together at the inputs of each layer to the activations that have progressed through the network with no shortcuts. These shortcuts encourage learning shared features between the layers and improve the supervision signal in earlier layers. Dilated (Atrous) convolutions have been used to increase the field of veiw of features without increasing the size of the network [15]. DeepLab is one such architecture that leverages these convolutions and it has reached state of the art accuracy on the Pascal VOC12 segmentation dataset several years in a row [3]. The latest DeepLab iterations (v2, v3) uses conditional random field postprocessing to smooth segmentations [4]. Pyramidal pooling is also used in several models to aggregate local image context and improve the expressiveness of a network[5] [16]. Many of these models rely on pre-trained parameters.

3. Approach

We introduce our model, The Huge Image-segmentation Convolutional Classifier (THICC), as an iterative improvement on the state-of-the-art *Tiramisu* model. Specifically, we replace ReLU activations with ELU activations, convolutional layers in dense blocks with atrous convolutions, max-pooling layers with 2-strided convolutions, and transpose convolution layers with resize-convolution layers.

3.1. DenseNets

As mentioned in section 1, DenseNets are composed of dense blocks. Each dense block contains several convolutional layers, each of which passes its output not just to the immediately next convolutional layer, but also to all further convolutional layers in a feed-forward fashion [9]. Let x_l denote the output of convolutional layer l and $F_l(x)$ denote the transformation operation at layer l (such as a convolution, then ReLU) in a dense block of size n (has n convolutional layers). Let x_0 be the input to the dense block, and x_n be the output of the dense block. Then,

$$x_l = F_l([x_{l-1}, x_{l-2}, ...x_0])$$

Here, $[x_{l-1}, x_{l-2}, ...x_0]$ denotes concatenation of the prior feature maps.

3.2. Tiramisu Model Architecture

We used the architecture as described in [10] for our baseline model and as a starting point for our modifications. From a high level, the model is composed of dense blocks (each block is as described in the rior section), transition-down blocks, and transition-up blocks, arranged in downsampling, bottleneck, and upsampling paths. The model is composed of a 3x3 convolution, followed by N dense-block-concatenate-transition-down groups, followed by a bottleneck dense-block, followed by N transition-up-concatenate-dense-block groups, followed by a 1x1 convolution outputting C feature maps, where C is the number of semantic classes, and finally a softmax layer to compute a probability distribution. Shortcut connections are added between corresponding dense blocks of the same resolution on downsampling and upsampling paths to preserve fine-grain detail in the upsampling stage. Readers looking for specific technical discussion and motivation should refer to [10].

3.3. ELU Activation

We replaced all activation functions in the network with *exponential linear units*, or ELU. The ELU activation is similar to the ReLU activation in that it is an identity function for positive inputs, but with the distinction of outputting an exponential for negative inputs. Concretely, ELU is defined as:

$$ELU(x) = \begin{cases} x, & \text{if } x < 0. \\ \alpha(\exp(x) - 1), & \text{otherwise.} \end{cases}$$

ELU provides numerous advantages over the ReLU [7]. Namely, ELU avoids the ”dying ReLU problem” in which ReLU units cannot recover when the input x becomes less than 0. As the derivative of the ELU is non-zero, the ELU unit can recover. Further, the exponential helps bias activations closer to 0, which has empirically been shown to lead to faster learning. ELU also empirically provides slightly higher accuracies, although in general convergence is similar to ReLU.

3.4. Atrous Convolutions

We then replaced all convolutional layers in dense blocks with atrous convolutions (also known as *dilated* convolutions) [15]. Atrous convolutions can be though of as ”convolution with holes.” That is, we insert zeros between consecutive elements along spatial dimensions in the filter and then apply the augmented filter to the image as a normal

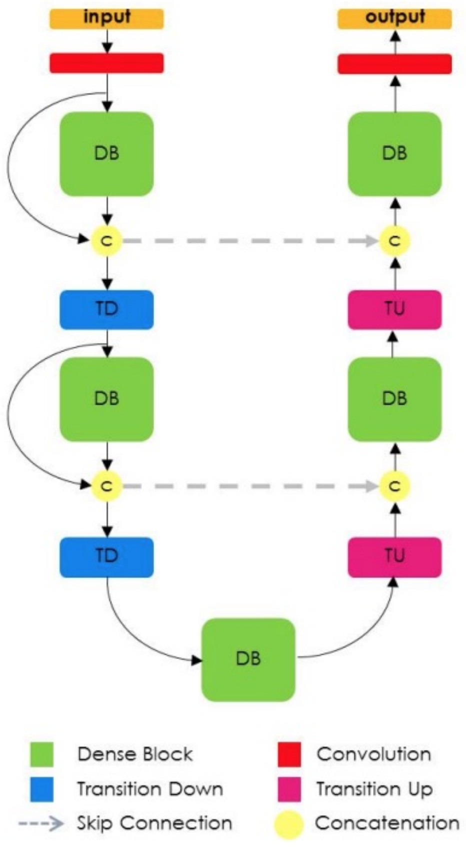

Figure 1. Tiramisu DenseNet architecture [10]

convolution. Atrous convolutions then have an additional hyperparameter: the *rate*, which controls how many zeros are inserted between consecutive elements. Specifically, $rate - 1$ zeros are inserted. Concretely, we define the augmented, atrous convolutional weight matrix $\tilde{W}_{ij}$ from the weight matrix W_{ij} as:

$$\tilde{W}_{ij} = \begin{cases} W_{\frac{i}{rate}, \frac{j}{rate}}, & \text{if } (i,j) \mod rate = (0,0). \\ 0, & \text{otherwise.} \end{cases}$$

Here, for simplicity, $\tilde{W}_{ij}$ and W_{ij} are zero-indexed, W is $N \times N$ and $\tilde{W}$ is $rate{*}N \times rate{*}N$. $(i,j) \mod rate = (0,0)$ denotes that i and j are both multiples of $rate$.

With the same filter size, atrous convolutions increase the receptive field for a neuron. In general, by increasing the filter size, and thus the number of parameters, linearly the receptive field increases exponentially. Intuitively, we can imagine that increasing the receptive field is useful for semantic segmentation, as it allows the neuron to "see" parts of the original image farther away, which may be useful for deciding the current part of the image.

3.5. Strided Convolutional Downsampling

We also replaced all the max-pooling layers in the architecture with 3x3 2-strided convolutional layers. By using strided convolutional layers instead of imposing parameterless max-pooling, the network is able to learn the most effective way to downsample the image through the network [14]. Note that a potential disadvantage of using strided convolutional layers is an increase in the number of parameters in the model.

3.6. Resize Convolutional Upsampling

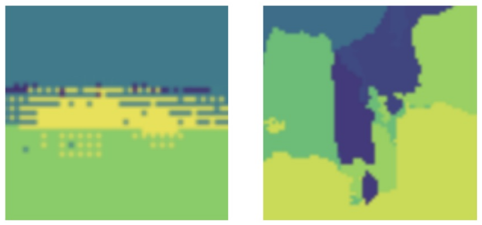

Figure 2. Tiramisu output (left). Thicc output with resize convolutions (right)

In the original Tiramisu model, *transpose convolutions* are used in to upsample and construct the output prediction map. However, Odena, et al. show in [12] that transpose convolutions are prone to visual checkerboard artifiacts. We implement resize convolution by conducting a nearest-neighbors resize followed by a strided normal convolution as describbed by Odena, et al. Note that we resize the image to twice the target size because of the following 2-strided convolution. The improvement of using resize convolutions in our model is highlighted in Figure 2.

Model	Global Mean Accuracy	Mean IoU
Tiramisu-67 Optimal	90.8%	65.8%
Tiramisu-103 Optimal	91.5%	66.9%
Thicc-67 16 hours	89.5%	64.6%
Thicc-103 16 hours	90.9%	65.6%

Table 1. Summary of global accuracies and mean intersection over union across the CamVid Dataset

4. Experimental Results

We trained our two models, Thicc-67 and Thicc-103, on Amazon EC2 P3 instances ($3 per hour) from scratch on the CamVid and ADE20K datasets, and report the global pixel-wise accuracies and Intersection over Union metric. For all datasets we used the same parameters for the RMSProp optimizer as in the original Tiramisu paper [10]. To regularize, we used dropout with $p_keep = 0.8$ and weight decay $1e-4$.

4.1. CamVid

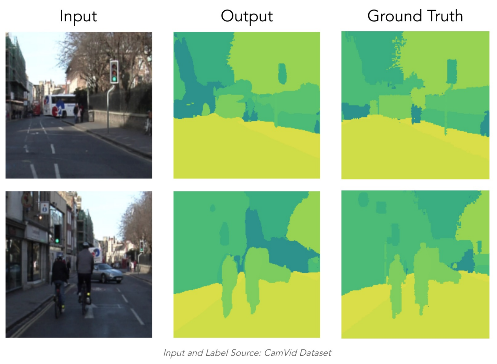

Figure 3. Thicc-103 on camVid dataset

We use the CamVid [2] dataset, which includes 367 training images and 11 semantic classes (and one void class), for urban scene understanding to benchmark our model against Tiramisu. We trained with a crop size of 224x224 and a batch size of 8. Our results on this dataset are summarized by table 1.

4.2. ADE20k (Benchmark)

We then evaluated the performance of our modifications as compared to the baseline Tiramisu model on the ADE20k dataset as a benchmark [17]. That is, we did not aim to train to completion, as we were constrained by compute and financial resources, but rather we wished to compare relative training performance between Tiramisu and Thicc. For the duration we trained on, training accuracy remained

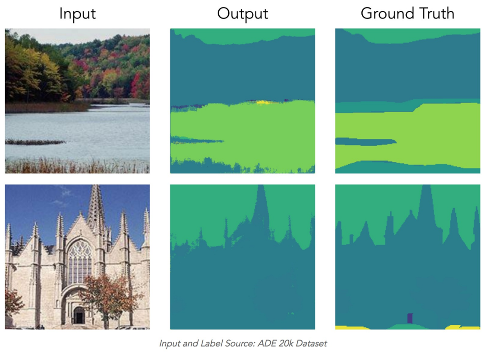

Figure 4. Thicc-103 on ADE20k dataset

the same, however validation accuracy was significantly improved (Figure 5).

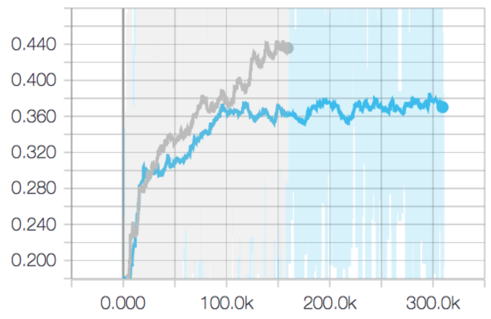

Figure 5. Validation accuracy Thicc-103 (gray) at 150k iterations vs Tiramisu-103 (light-blue) at 300k iterations

5. Conclusion

In this paper, we augmented Tiramisu-103, a state-of-the-art semantic image segmentation network, to integrate recent advances in computer vision to improve segmentation accuracy. We have shown that atrous convolutions, strided convolutional downsampling, resize convolutional upsampling, and ELU activations work well in tanem to to significantly improve relative performance. Although Thicc-103 did not exceed the global mean accuracy achieved by Tiramisu-103, it was not run until convergence (see attached Figures), suggesting that significant gains can be attained by training for more epochs. Strided convolutions improved validation accuracy. Atrous convolutions improved training accuracy. Resize convolutions reduced output artifacts. This is the first reported time a DenseNet architecture has been benchmarked against ADE20k.

References

[1] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, abs/1511.00561, 2015.

[2] G. J. Brostow, J. Fauqueur, and R. Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 2008.

[3] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *CoRR*, abs/1412.7062, 2014.

[4] 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 crfs. *CoRR*, abs/1606.00915, 2016.

[5] L. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *CoRR*, abs/1706.05587, 2017.

[6] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in Neural Information Processing Systems 25*, pages 2843–2851. Curran Associates, Inc., 2012.

[7] D. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *CoRR*, abs/1511.07289, 2015.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[9] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[10] S. Jégou, M. Drozdzal, D. Vázquez, A. Romero, and Y. Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. *CoRR*, abs/1611.09326, 2016.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. pages 1097–1105, 2012.

[12] A. Odena, V. Dumoulin, and C. Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.

[13] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[14] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller. Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806, 2014.

[15] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *CoRR*, abs/1511.07122, 2015.

[16] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. *CoRR*, abs/1612.01105, 2016.

[17] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Semantic understanding of scenes through the ade20k dataset. *arXiv preprint arXiv:1608.05442*, 2016.