The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation

Article · November 2016

CITATIONS
519

READS
1,050

5 authors, including:

Michal Drozdzal
University of Barcelona
48 PUBLICATIONS **2,965** CITATIONS
SEE PROFILE

David Vázquez
Autonomous University of Barcelona
100 PUBLICATIONS **4,544** CITATIONS
SEE PROFILE

Adriana Romero
Université de Montréal
37 PUBLICATIONS **5,174** CITATIONS
SEE PROFILE

Y. Bengio
Université de Montréal
816 PUBLICATIONS **275,944** CITATIONS
SEE PROFILE

The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation

Simon Jégou[1] Michal Drozdzal[2,3] David Vazquez[1,4] Adriana Romero[1] Yoshua Bengio[1]
[1]Montreal Institute for Learning Algorithms [2]École Polytechnique de Montréal
[3]Imagia Inc., Montréal, [4]Computer Vision Center, Barcelona
simon.jegou@gmail.com, michal@imagia.com, dvazquez@cvc.uab.es,
adriana.romero.soriano@umontreal.ca, yoshua.umontreal@gmail.com

Abstract

*State-of-the-art approaches for semantic image segmentation are built on Convolutional Neural Networks (CNNs). The typical segmentation architecture is composed of (a) a downsampling path responsible for extracting coarse semantic features, followed by (b) an upsampling path trained to recover the input image resolution at the output of the model and, optionally, (c) a post-processing module (*e.g. *Conditional Random Fields) to refine the model predictions.*

Recently, a new CNN architecture, Densely Connected Convolutional Networks (DenseNets), has shown excellent results on image classification tasks. The idea of DenseNets is based on the observation that if each layer is directly connected to every other layer in a feed-forward fashion then the network will be more accurate and easier to train.

In this paper, we extend DenseNets to deal with the problem of semantic segmentation. We achieve state-of-the-art results on urban scene benchmark datasets such as CamVid and Gatech, without any further post-processing module nor pretraining. Moreover, due to smart construction of the model, our approach has much less parameters than currently published best entries for these datasets. Code to reproduce the experiments is publicly available here : `https://github.com/SimJeg/FC-DenseNet`

1. Introduction

Convolutional Neural Networks (CNNs) are driving major advances in many computer vision tasks, such as image classification [29], object detection [25, 24] and semantic image segmentation [20]. The last few years have witnessed outstanding improvements on CNN-based models. *Very deep* architectures [29, 11, 31] have shown impressive results on standard benchmarks such as ImageNet [6] or MSCOCO [19]. State-of-the-art CNNs heavily reduce the input resolution through successive pooling layers and,

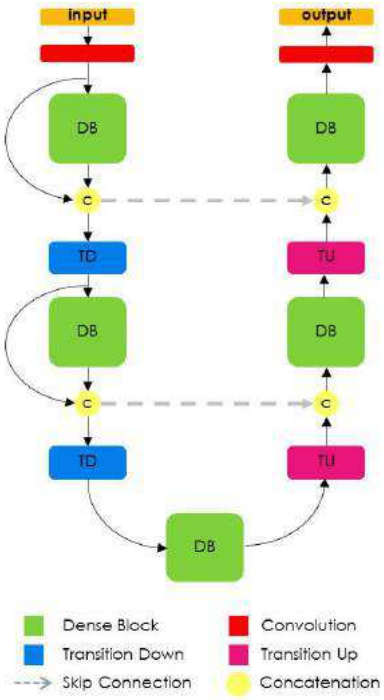

Figure 1. Diagram of our architecture for semantic segmentation. Our architecture is built from dense blocks. The diagram is composed of a downsampling path with 2 Transitions Down (TD) and an upsampling path with 2 Transitions Up (TU). A circle represents concatenation and arrows represent connectivity patterns in the network. Gray horizontal arrows represent skip connections, the feature maps from the downsampling path are concatenated with the corresponding feature maps in the upsampling path. Note that the connectivity pattern in the upsampling and the downsampling paths are different. In the downsampling path, the input to a dense block is concatenated with its output, leading to a linear growth of the number of feature maps, whereas in the upsampling path, it is not.

thus, are well suited for applications where a single prediction per input image is expected (*e.g.* image classification task).

Fully Convolutional Networks (FCNs) [20, 27] were introduced in the literature as a natural extension of CNNs to tackle per pixel prediction problems such as semantic image segmentation. FCNs add upsampling layers to standard CNNs to recover the spatial resolution of the input at the output layer. As a consequence, FCNs can process images of arbitrary size. In order to compensate for the resolution loss induced by pooling layers, FCNs introduce skip connections between their downsampling and upsampling paths. Skip connections help the upsampling path recover fine-grained information from the downsampling layers.

Among CNN architectures extended as FCNs for semantic segmentation purposes, Residual Networks (ResNets) [11] make an interesting case. ResNets are designed to ease the training of *very deep* networks (of hundreds of layers) by introducing a residual block that sums two signals: a non-linear transformation of the input and its identity mapping. The identity mapping is implemented by means of a shortcut connection. ResNets have been extended to work as FCNs [4, 8] yielding very good results in different segmentation benchmarks. ResNets incorporate additional paths to FCN (shortcut paths) and, thus, increase the number of connections within a segmentation network. This additional shortcut paths have been shown not only to improve the segmentation accuracy but also to help the network optimization process, resulting in faster convergence of the training [8].

Recently, a new CNN architecture, called *DenseNet*, was introduced in [13]. DenseNets are built from *dense blocks* and pooling operations, where each dense block is an iterative concatenation of previous feature maps. This architecture can be seen as an extension of ResNets [11], which performs iterative summation of previous feature maps. However, this small modification has some interesting implications: (1) parameter efficiency, DenseNets are more efficient in the parameter usage; (2) implicit deep supervision, DenseNets perform deep supervision thanks to short paths to all feature maps in the architecture (similar to Deeply Supervised Networks [18]); and (3) feature reuse, all layers can easily access their preceding layers making it easy to reuse the information from previously computed feature maps. The characteristics of DenseNets make them a *very good fit* for semantic segmentation as they naturally induce skip connections and multi-scale supervision.

In this paper, we extend DenseNets to work as FCNs by adding an upsampling path to recover the full input resolution. Naively building an upsampling path would result in a *computationally intractable* number of feature maps with very high resolution prior to the softmax layer. This is because one would multiply the high resolution feature maps with a large number of input filters (from all the layers below), resulting in both very large amount of computation and number of parameters. In order to mitigate this effect, we *only* upsample the feature maps created by the preceding dense block. Doing so allows to have a number of dense blocks at each resolution of the upsampling path independent of the number of pooling layers. Moreover, given the network architecture, the upsampled dense block combines the information contained in the other dense blocks of the same resolution. The higher resolution information is passed by means of a standard skip connection between the downsampling and the upsampling paths. The details of the proposed architecture are shown in Figure 1. We evaluate our model on two challenging benchmarks for urban scene understanding, Camvid [2] and Gatech [22], and confirm the potential of DenseNets for semantic segmentation by improving the state-of-the-art.

Thus, the contributions of the paper can be summarized as follows:

- We carefully extend the DenseNet architecture [13] to fully convolutional networks for semantic segmentation, while mitigating the feature map explosion.
- We highlight that the proposed upsampling path, built from dense blocks, performs better than upsampling path with more standard operations, such as the ones in [27].
- We show that such a network can outperform current state-of-the-art results on standard benchmarks for urban scene understanding without neither using pretrained parameters nor any further post-processing.

2. Related Work

Recent advances in semantic segmentation have been devoted to improve architectural designs by (1) improving the upsampling path and increasing the connectivity within FCNs [27, 1, 21, 8]; (2) introducing modules to account for broader context understanding [36, 5, 37]; and/or (3) endowing FCN architectures with the ability to provide structured outputs [16, 5, 38].

First, different alternatives have been proposed in the literature to address the resolution recovery in FCN's upsampling path; from simple bilinear interpolation [10, 20, 1] to more sophisticated operators such as unpooling [1, 21] or transposed convolutions [20]. Skip connections from the downsampling to the upsampling path have also been adopted to allow for a finer information recovery [27]. More recently, [8] presented a thorough analysis on the combination of identity mapping [11] and long skip connections [27] for semantic segmentation.

Second, approaches that introduce larger context to semantic segmentation networks include [10, 36, 5, 37]. In

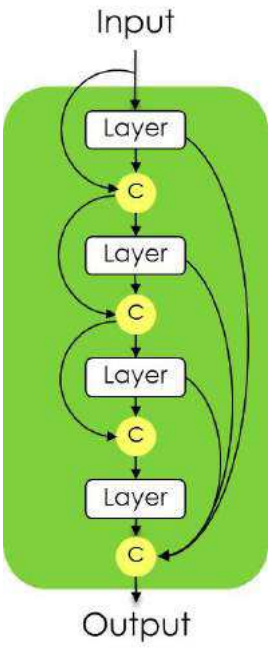

Figure 2. Diagram of a dense block of 4 layers. A first layer is applied to the input to create k feature maps, which are concatenated to the input. A second layer is then applied to create another k features maps, which are again concatenated to the previous feature maps. The operation is repeated 4 times. The output of the block is the concatenation of the outputs of the 4 layers, and thus contains $4 * k$ feature maps

[10], an unsupervised global image descriptor is computed added to the feature maps for each pixel. In [36], Recurrent Neural Networks (RNNs) are used to retrieve contextual information by sweeping the image horizontally and vertically in both directions. In [5], dilated convolutions are introduced as an alternative to late CNN pooling layers to capture larger context without reducing the image resolution. Following the same spirit, [37] propose to provide FCNs with a context module built as a stack of dilated convolutional layers to enlarge the field of view of the network.

Third, Conditional Random Fields (CRF) have long been a popular choice to enforce structure consistency to segmentation outputs. More recently, fully connected CRFs [16] have been used to include structural properties of the output of FCNs [5]. Interestingly, in [38], RNN have been introduced to approximate mean-field iterations of CRF optimization, allowing for an end-to-end training of both the FCN and the RNN.

Finally, it is worth noting that current state-of-the-art FCN architectures for semantic segmentation often rely on pre-trained models (*e.g.* VGG [29] or ResNet101 [11]) to improve their segmentation results [20, 1, 4].

3. Fully Convolutional DenseNets

As mentioned in Section 1, FCNs are built from a downsampling path, an upsampling path and skip connections. Skip connections help the upsampling path recover spatially detailed information from the downsampling path, by reusing features maps. The goal of our model is to further exploit the feature reuse by extending the more sophisticated DenseNet architecture, while avoiding the feature explosion at the upsampling path of the network.

In this section, we detail the proposed model for semantic segmentation. First, we review the recently proposed DenseNet architecture. Second, we introduce the construction of the novel upsampling path and discuss its advantages w.r.t. a naive DenseNet extension and more classical architectures. Finally, we wrap up with the details of the main architecture used in Section 4.

3.1. Review of DenseNets

Let x_ℓ be the output of the ℓ^{th} layer. In a standard CNN, x_ℓ is computed by applying a non-linear transformation H_ℓ to the output of the previous layer $x_{\ell-1}$

$$x_\ell = H_\ell(x_{\ell-1}), \tag{1}$$

where H is commonly defined as a convolution followed by a rectifier non-linearity (ReLU) and often dropout [30].

In order to ease the training of very deep networks, ResNets [11] introduce a residual block that sums the identity mapping of the input to the output of a layer. The resulting output x_ℓ becomes

$$x_\ell = H_\ell(x_{\ell-1}) + x_{\ell-1}, \tag{2}$$

allowing for the reuse of features and permitting the gradient to flow directly to earlier layers. In this case, H is defined as the repetition (2 or 3 times) of a block composed of Batch Normalization (BN) [14], followed by ReLU and a convolution.

Pushing this idea further, DenseNets [13] design a more sophisticated connectivity pattern that iteratively concatenates all feature outputs in a feedforward fashion. Thus, the output of the ℓ^{th} layer is defined as

$$x_\ell = H_\ell([x_{\ell-1}, x_{\ell-2}, ..., x_0]), \tag{3}$$

where $[...]$ represents the concatenation operation. In this case, H is defined as BN, followed by ReLU, a convolution and dropout. Such connectivity pattern strongly encourages the reuse of features and makes all layers in the architecture receive direct supervision signal. The output dimension of each layer ℓ has k feature maps where k, hereafter referred as to *growth rate* parameter, is typically set to a small value (*e.g.* $k = 12$). Thus, the number of feature maps in DenseNets grows linearly with the depth (*e.g.* after ℓ layers, the input $[x_{\ell-1}, x_{\ell-2}, ..., x_0]$ will have $\ell \times k$ feature maps).

A *transition down* is introduced to reduce the spatial dimensionality of the feature maps. Such transformation is composed of a 1×1 convolution (which conserves the number of feature maps) followed by a 2×2 pooling operation.

In the remainder of the article, we will call *dense block* the concatenation of the *new* feature maps created at a given resolution. Figure 2 shows an example of dense block construction. Starting from an input x_0 (input image or output of a transition down) with m feature maps, the first layer of the block generates an output x_1 of dimension k by applying $H_1(x_0)$. These k feature maps are then stacked to the previous m feature maps by concatenation ($[x_1, x_0]$) and used as input to the second layer. The same operation is repeated n times, leading to a new dense block with $n \times k$ feature maps.

3.2. From DenseNets to Fully Convolutional DenseNets

The DenseNet architecture described in Subsection 3.1 constitutes the downsampling path of our Fully Convolutional DenseNet (FC-DenseNet). Note that, in the downsampling path, the linear growth in the number of features is compensated by the reduction in spatial resolution of each feature map after the pooling operation. The last layer of the downsampling path is referred to as *bottleneck*.

In order to recover the input spatial resolution, FCNs introduce an upsampling path composed of convolution, upsampling operations (transposed convolutions or unpooling operations) and skip connections. In FC-DenseNets, we substitute the convolution operation by a dense block and an upsampling operation referred to as *transition up*. Transition up modules consist of a transposed convolution that upsamples the previous feature maps. The upsampled feature maps are then concatenated to the ones coming from the skip connection to form the input of a new dense block. Since the upsampling path increases the feature maps spatial resolution, the linear growth in the number of features would be too memory demanding, especially for the full resolution features in the pre-softmax layer.

In order to overcome this limitation, the input of a dense block is not concatenated with its output. Thus, the transposed convolution is applied only to the feature maps obtained by *the last dense block and not to all feature maps concatenated so far*. The last dense block summarizes the information contained in all the previous dense blocks at the same resolution. Note that some information from earlier dense blocks is lost in the transition down due to the pooling operation. Nevertheless, this information is available in the downsampling path of the network and can be passed via skip connections. Hence, the dense blocks of the upsampling path are computed using all the available feature maps at a given resolution. Figure 1 illustrates this idea in detail.

Therefore, our upsampling path approach allows us to build very deep FC-DenseNets without a feature map explosion. An alternative way of implementing the upsampling path would be to perform consecutive transposed convolutions and complement them with skip connections from the downsampling path in a U-Net [27] or FCN-like [20] fashion. This will be further discussed in Section 4

3.3. Semantic Segmentation Architecture

In this subsection, we detail the main architecture, *FC-DenseNet103*, used in Section 4.

First, in Table 1, we define the dense block layer, transition down and transition up of the architecture. Dense block layers are composed of BN, followed by ReLU, a 3×3 same convolution (no resolution loss) and dropout with probability $p = 0.2$. The growth rate of the layer is set to $k = 16$. Transition down is composed of BN, followed by ReLU, a 1×1 convolution, dropout with $p = 0.2$ and a non-overlapping max pooling of size 2×2. Transition up is composed of a 3×3 transposed convolution with stride 2 to compensate for the pooling operation.

Second, in Table 2, we summarize all Dense103 layers. This architecture is built from 103 convolutional layers : a first one on the input, 38 in the downsampling path, 15 in the bottleneck and 38 in the upsampling path. We use 5 Transition Down (TD), each one containing one extra convolution, and 5 Transition Up (TU), each one containing a transposed convolution. The final layer in the network is a 1×1 convolution followed by a softmax non-linearity to provide the per class distribution at each pixel.

It is worth noting that, as discussed in Subsection 3.2, the proposed upsampling path properly mitigates the DenseNet feature map explosion, leading to reasonable pre-softmax feature map number of 256.

Finally, the model is trained by minimizing the pixel-wise cross-entropy loss.

4. Experiments

We evaluate our method on two urban scene understanding datasets: CamVid [2], and Gatech [22]. We trained our models *from scratch without using any extra-data nor post-processing module*. We report the results using the Intersection over Union (IoU) metric and the global accuracy (pixel-wise accuracy on the dataset). For a given class c, predictions (o_i) and targets (y_i), the IoU is defined by

$$IoU(c) = \frac{\sum_i (o_i == c \wedge y_i == c)}{\sum_i (o_i == c \vee y_i == c)}, \tag{4}$$

where $\wedge$ is a logical *and* operation, while $\vee$ is a logical *or* operation. We compute IoU by summing over all the pixels i of the dataset.

Layer
Batch Normalization
ReLU
3×3 Convolution
Dropout $p = 0.2$

Transition Down (TD)
Batch Normalization
ReLU
1×1 Convolution
Dropout $p = 0.2$
2×2 Max Pooling

Transition Up (TU)
3×3 Transposed Convolution $stride = 2$

Table 1. Building blocks of fully convolutional DenseNets. From left to right: layer used in the model, Transition Down (TD) and Transition Up (TU). See text for details.

4.1. Architecture and training details

We initialize our models using HeUniform [12] and train them with RMSprop [33], with an initial learning rate of $1e-3$ and an exponential decay of 0.995 after each epoch. All models are trained on data augmented with random crops and vertical flips. For all experiments, we finetune our models with full size images and learning rate of $1e-4$. We use validation set to earlystop the training and the finetuning. We monitor mean IoU or mean accuracy and use patience of 100 (50 during finetuning).

We regularized our models with a weight decay of $1e-4$ and a dropout rate of 0.2. For batch normalization, we use current batch statistics at training, validation and test time.

Architecture
Input, $m = 3$
3×3 Convolution, $m = 48$
DB (4 layers) + TD, $m = 112$
DB (5 layers) + TD, $m = 192$
DB (7 layers) + TD, $m = 304$
DB (10 layers) + TD, $m = 464$
DB (12 layers) + TD, $m = 656$
DB (15 layers), $m = 896$
TU + DB (12 layers), $m = 1088$
TU + DB (10 layers), $m = 816$
TU + DB (7 layers), $m = 578$
TU + DB (5 layers), $m = 384$
TU + DB (4 layers), $m = 256$
1×1 Convolution, $m = c$
Softmax

Table 2. Architecture details of FC-DenseNet103 model used in our experiments. This model is built from 103 convolutional layers. In the Table we use following notations: DB stands for Dense Block, TD stands for Transition Down, TU stands for Transition Up, BN stands for Batch Normalization and m corresponds to the total number of feature maps at the end of a block. c stands for the number of classes.

4.2. CamVid dataset

CamVid[1] [2] is a dataset of fully segmented videos for urban scene understanding. We used the split and image resolution from [1], which consists of 367 frames for training, 101 frames for validation and 233 frames for test. Each frame has a size 360×480 and its pixels are labeled with 11 semantic classes. Our models were trained with crops of 224×224 and batch size 3. At the end, the model is finetuned with full size images.

In Table 3, we report our results for three networks with respectively (1) 56 layers (*FC-DenseNet56*), with 4 layers per dense block and a growth rate of 12; (2) 67 layers (*FC-DenseNet67*) with 5 layers per dense block and a growth rate of 16; and (3) 103 layers (*FC-DenseNet103*) with a growth rate $k = 16$ (see Table 2 for details). We also trained an architecture using standard convolutions in the upsampling path instead of dense blocks (*Classic Upsampling*). In the latter architecture, we used 3 convolutions per resolution level with respectively 512, 256, 128, 128 and 64 filters, as in [27]. Results show clear superiority of the proposed upsampling path w.r.t. the classic one, consistently improving the IoU significantly for all classes. Particularly, we observe that unrepresented classes benefit notably from the FC-DenseNet architecture, namely *sign*, *pedestrian*, *fence*, *cyclist* experience a crucial boost in performance (ranging from 15% to 25%).

As expected, when comparing FC-DenseNet56 or FC-DenseNet67 to FC-DenseNet103, we see that the model benefits from having more depth as well as more parameters.

When compared to other methods, we show that FC-DenseNet architectures achieve state-of-the-art, improving upon models with 10 times more parameters. It is worth mentioning that our small model FC-DenseNet56 already outperforms popular architectures with at least 100 times more parameters.

It is worth noting that images in CamVid correspond to video frames and, thus, the dataset contains temporal information. Some state-of-the-art methods such as [17] incorporate long range spatio-temporal regularization to the out-

[1] http://mi.eng.cam.ac.uk/research/projects/VideoRec/CamVid/

put of a FCN to boost their performance. Our model is able to outperform such state-of-the-art model, without requiring any temporal smoothing. However, any post-processing temporal regularization is complementary to our approach and could bring additional improvements.

Unlike most of the current state-of-the-art methods, FC-DenseNets have not been pretrained on large datasets such as ImageNet [6] and could most likely benefit from such pretraining. More recently, it has been shown that deep networks can also boost their performance when pretrained on data other than natural images, such as video games [26, 28] or clipart [3], and this an interesting direction to explore.

Figure 3 shows some qualitative segmentation results on the CamVid dataset. Qualitative results are well aligned with the quantitative ones, showing sharp segmentations that account for a lot of details. For example, trees, column poles, sidewalk and pedestrians appear very well sketched. Among common errors, we find that thin details found in trees can be confused with column poles (see fifth row), buses and trucks can be confused with buildings (fourth row), and shop signs can be confused with road signs (second row).

4.3. Gatech dataset

Gatech[2] [23] is a geometric scene understanding dataset, which consists of 63 videos for training/validation and 38 for testing. Each video has between 50 and 300 frames (with an average of 190). A pixel-wise segmentation map is provided for each frame. There are 8 classes in the dataset: *sky*, *ground*, *buildings*, *porous* (mainly trees), *humans*, *cars*, *vertical mix* and *main mix*. The dataset was originally built to learn 3D geometric structure of outdoor video scenes and the standard metric for this dataset is mean global accuracy.

We used the FC-DenseNet103 model pretrained on CamVid, removed the softmax layer, and finetuned it for 10 epochs with crops of 224×224 and batch size 5. Given the high redundancy in Gatech frames, we used *only* one out of 10 frames to train the model and tested it on all full resolution test set frames.

In Table 4, we report the obtained results. We compare the results to the recently proposed method for video segmentation of [34], which reports results of their architecture with 2D and 3D convolutions. Frame-based 2D convolutions do not have temporal information. As it can be seen in Table 4, our method gives an impressive improvement of 23.7% in global accuracy with respect to previously published state-of-the-art with 2D convolutions. Moreover, our model (trained with only 2D convolutions) also achieves a significant improvement over state-of-the-art models based on spatio-temporal 3D convolutions (3.4% improvement).

[2] http://www.cc.gatech.edu/cpl/projects/videogeometriccontext/

5. Discussion

Our fully convolutional DenseNet implicitly inherits the advantages of DenseNets, namely: (1) parameter efficiency, as our network has substantially less parameters than other segmentation architectures published for the Camvid dataset; (2) implicit deep supervision, we tried including additional levels of supervision to different layers of our network without noticeable change in performance; and (3) feature reuse, as all layers can easily access their preceding layers not only due to the iterative concatenation of feature maps in a dense block but also thanks to skip connections that enforce connectivity between downsampling and upsampling path.

Recent evidence suggest that ResNets behave like ensemble of relatively shallow networks [35]: "Residual networks avoid the vanishing gradient problem by introducing short paths which can carry gradient throughout the extent of very deep networks". It would be interesting to revisit this finding in the context of fully convolutional DenseNets. Due to iterative feature map concatenation in the dense block, the gradients are forced to be passed through networks of different depth (with different numbers of non-linearities). Thus, thanks to the smart connectivity patterns, FC-DenseNets might represent an ensemble of variable depth networks. This particular ensemble behavior would be very interesting for semantic segmentation models, where the ensemble of different paths throughout the model would capture the multi-scale appearance of objects in urban scene.

6. Conclusion

In this paper, we have extended DenseNets and made them fully convolutional to tackle the problem semantic image segmentation. The main idea behind DenseNets is captured in dense blocks that perform iterative concatenation of feature maps. We designed an upsampling path mitigating the linear growth of feature maps that would appear in a naive extension of DenseNets.

The resulting network is *very deep* (from 56 to 103 layers) and has *very few* parameters, about 10 fold reduction w.r.t. state-of-the-art models. Moreover, it improves state-of-the-art performance on challenging urban scene understanding datasets (CamVid and Gatech), without neither additional post-processing, pretraining, nor including temporal information.

Aknowledgements

The authors would like to thank the developers of Theano [32] and Lasagne [7]. Special thanks to Frédéric Bastien for his work assessing the compilation issues. Thanks to Francesco Visin for his well designed dataloader [9], as well as Harm de Vries for his support

Model	Pretrained	# parameters (M)	Building	Tree	Sky	Car	Sign	Road	Pedestrian	Fence	Pole	Sidewalk	Cyclist	Mean IoU	Global accuracy
SegNet [1]	✓	29.5	68.7	52.0	87.0	58.5	13.4	86.2	25.3	17.9	16.0	60.5	24.8	46.4	62.5
Bayesian SegNet [15]	✓	29.5	n/a											63.1	86.9
DeconvNet [21]	✓	252	n/a											48.9	85.9
Visin et al. [36]	✓	32.3	n/a											58.8	88.7
FCN8 [20]	✓	134.5	77.8	71.0	88.7	76.1	32.7	91.2	41.7	24.4	19.9	72.7	31.0	57.0	88.0
DeepLab-LFOV [5]	✓	37.3	81.5	74.6	89.0	82.2	42.3	92.2	48.4	27.2	14.3	75.4	50.1	61.6	–
Dilation8 [37]	✓	140.8	82.6	76.2	89.0	84.0	46.9	92.2	56.3	35.8	23.4	75.3	55.5	65.3	79.0
Dilation8 + FSO [17]	✓	140.8	**84.0**	77.2	91.3	**85.6**	**49.9**	92.5	59.1	**37.6**	16.9	76.0	**57.2**	66.1	88.3
Classic Upsampling	✗	20	73.5	72.2	92.4	66.2	26.9	90.0	37.7	22.7	30.8	69.6	25.1	55.2	86.8
FC-DenseNet56 (k=12)	✗	1.5	77.6	72.0	92.4	73.2	31.8	92.8	37.9	26.2	32.6	79.9	31.1	58.9	88.9
FC-DenseNet67 (k=16)	✗	3.5	80.2	75.4	93.0	78.2	40.9	**94.7**	58.4	30.7	**38.4**	81.9	52.1	65.8	90.8
FC-DenseNet103 (k=16)	✗	9.4	83.0	**77.3**	**93.0**	77.3	43.9	94.5	**59.6**	37.1	37.8	**82.2**	50.5	**66.9**	**91.5**

Table 3. Results on CamVid dataset. Note that we trained our own pretrained FCN8 model

Model	Acc.
2D models (no time)	
2D-V2V-from scratch [34]	55.7
FC-DenseNet103	**79.4**
3D models (incorporate time)	
3D-V2V-from scratch [34]	66.7
3D-V2V-pretrained [34]	76.0

Table 4. Results on Gatech dataset

in network parallelization, and Tristan Sylvain. We acknowledge the support of the following agencies for research funding and computing support: Imagia Inc., Spanish projects TRA2014-57088-C2-1-R & 2014-SGR-1506, TECNIOspring-FP7-ACCI grant.

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. Shotton, J. Fauqueur, and R. Cipolla. Segmentation and recognition using structure from motion point clouds. In *European Conference on Computer Vision (ECCV)*, 2008.

[3] L. Castrejon, Y. Aytar, C. Vondrick, H. Pirsiavash, and A. Torralba. Learning aligned cross-modal representations from weakly aligned data. *CoRR*, abs/1607.07295, 2016.

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

[6] 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 (CVPR)*, 2009.

[7] S. Dieleman, J. Schlter, C. Raffel, E. Olson, and et al. Lasagne: First release., Aug. 2015.

[8] M. Drozdzal, E. Vorontsov, G. Chartrand, S. Kadoury, and C. Pal. The importance of skip connections in biomedical image segmentation. *CoRR*, abs/1608.04117, 2016.

[9] A. R. F. Visin. Dataset loaders: a python library to load and preprocess datasets. `https://github.com/fvisin/dataset_loaders`, 2017.

[10] C. Gatta, A. Romero, and J. van de Weijer. Unrolling loopy top-down semantic feedback in convolutional deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) workshop*, 2014.

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

[12] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015.

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

[14] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[15] A. Kendall, V. Badrinarayanan, and R. Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *CoRR*, abs/1511.02680, 2015.

[16] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in Neural Information Processing Systems (NIPS)*. 2011.

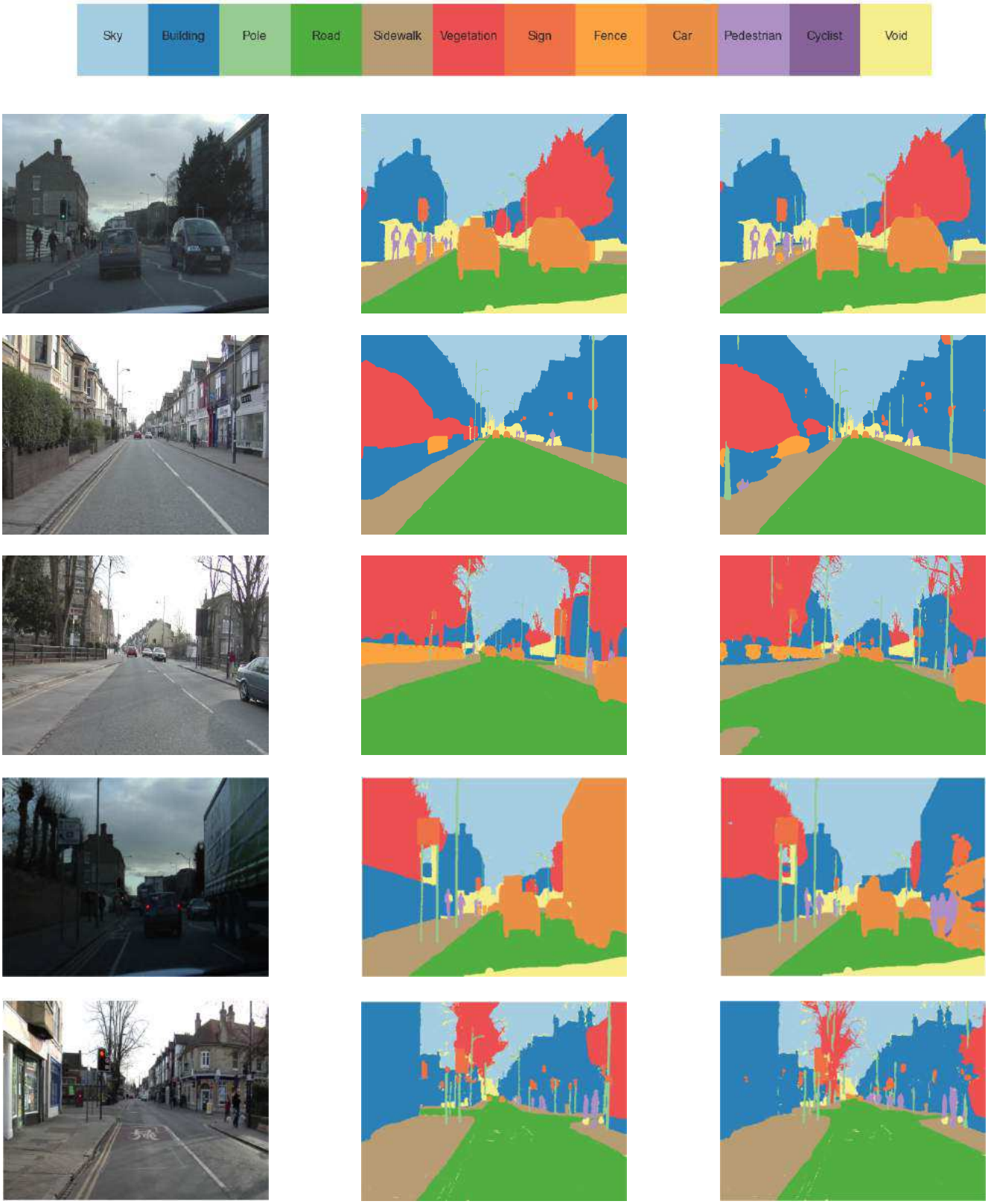

Figure 3. Qualitative results on the CamVid test set. Pixels labeled in yellow are void class. Each row represents (from left to right): original image, original annotation (ground truth) and prediction of our model.

[17] A. Kundu, V. Vineet, and V. Koltun. Feature space optimization for semantic video segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[18] C. Lee, S. Xie, P. W. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2015.

[19] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollr, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014.

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

[21] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. *arXiv preprint arXiv:1505.04366*, 2015.

[22] S. H. Raza, M. Grundmann, and I. Essa. Geometric context from video. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[23] S. H. Raza, M. Grundmann, and I. Essa. Geometric context from video. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[24] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[25] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[26] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. In *European Conference on Computer Vision (ECCV)*, 2016.

[27] 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 (MICAI)*, 2015.

[28] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

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

[30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[32] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[33] T. Tieleman and G. Hinton. rmsprop adaptive learning. In *COURSERA: Neural Networks for Machine Learning*, 2012.

[34] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Deep end2end voxel2voxel prediction. *CoRR*, abs/1511.06681, 2015.

[35] A. Veit, M. J. Wilber, and S. J. Belongie. Residual networks are exponential ensembles of relatively shallow networks. *CoRR*, abs/1605.06431, 2016.

[36] F. Visin, M. Ciccone, A. Romero, K. Kastner, K. Cho, Y. Bengio, M. Matteucci, and A. Courville. Reseg: A recurrent neural network-based model for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) workshop*, 2016.

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

[38] 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.