Ladder-style DenseNets for Semantic Segmentation of Large Natural Images

Ivan Krešo Siniša Šegvić
Faculty of Electrical Engineering and Computing
University of Zagreb, Croatia
name.surname@fer.hr

Josip Krapac
EyeEm Mobile GmbH
Berlin, Germany
josip@eyeem.com

Abstract

Recent progress of deep image classification models provides a large potential to improve state-of-the-art performance in related computer vision tasks. However, the transition to semantic segmentation is hampered by strict memory limitations of contemporary GPUs. The extent of feature map caching required by convolutional backprop poses significant challenges even for moderately sized PASCAL images, while requiring careful architectural considerations when the source resolution is in the megapixel range. To address these concerns, we propose a DenseNet-based ladder-style architecture which is able to deliver high modelling power with very lean representations at the original resolution. The resulting fully convolutional models have few parameters, allow training at megapixel resolution on commodity hardware and display fair semantic segmentation performance even without ImageNet pre-training. We present experiments on Cityscapes and Pascal VOC 2012 datasets and report competitive results.

1. Introduction

Semantic segmentation is a computer vision task in which the algorithm has to classify pixels into meaningful high-level categories. Due to being complementary to object localization, it represents an important step towards advanced future techniques for natural image understanding. Some of the most attractive application fields include autonomous control, intelligent transportation systems and automated analysis of photographic collections.

Early semantic segmentation approaches optimized a trade-off between multiple local classification cues (texture, color etc) and their global agreement across the image. Later work improved these ideas with nonlinear feature embeddings [3], multiscale analysis [5] and depth [16]. Global consistency has been improved by promoting agreement between pixels and semantic labels [15], as well as by learning asymetric pairwise semantic agreement potentials [20]. However, none of these approaches have been able to sustain an improvement rate comparable to simple local application of state-of-the-art deep convolutional models [17, 5].

Deep convolutional models have caused unprecedented growth of computer vision performance in the last five years. Depth of state-of-the-art models has been steadily increasing from 8 levels [17] to 19 [30], 22 [31], 152 [8], 201 [10], and more [8]. Much recent attention has been directed towards models with residual connections (also known as ResNets) [8, 9] which sum the output of a nonlinear mapping with its input. This construction introduces an auxiliary information path which allows a direct propagation across the layers, similarly to the way the state vector flows across LSTM cells. This improves the gradient flow towards the early model layers and allows successful backprop training through hundreds of convolutional layers. However, in contrast to the great depth of residual models, Veit et al [33] have empirically determined that most of their training occurs along relatively short paths. Hence, it has been conjectured [33] that a residual model acts as an exponentially large ensemble of moderately deep sub-models.

Recent approaches [18, 10] replicate and exceed the ResNet success with suitable skip-connections [21, 22] between early and later layers. This encourages feature sharing and reduces the number of parameters (especially when semantic classes have differing complexities), while also favouring the gradient flow towards early layers. Our work is based on the DenseNet architecture [10] in which the convolutions operate on a concatenation of all previous features at the current resolution. ImageNet experiments have shown that DenseNets succeed to match the ResNet performance with a three-fold parameter reduction [10].

Regardless of the particular architecture, early layers of a deep model are typically rich with detail and poor in semantics. On the other hand, the later, deeper layers are rich with semantics and poor in details. Thus, we see that blending features at different levels of abstraction has a potential to improve specialization, and lead to a better exploitation of parameters. Such blending can be conveniently achieved with a ladder network [32]. Suppose we have feature tensors F_t produced by a usual convolutional pipeline (F_{t+1} is at half resolution of F_t $\forall t$). Then, a blended representation $\hat{F}_u$ which combines feature tensors F_t for all $t \geq u$ can be recursively obtained by mixing F_t with $\hat{F}_{t+1}$.

$$\hat{F}_t = g_t(F_t, \hat{F}_{t+1}) \quad (1)$$

The resulting model first performs the usual succession from F_0 (being the input) to F_n = $\hat{F}_n$ (being the most subsampled representation), and then proceeds with backward-blending from $\hat{F}_n$ to $\hat{F}_0$ (being the result). This idea has been introduced in autoencoders [32], while recently it has also been used in segmentation [27] and detection [29].

In this paper we present an efficient and effective architecture for semantic segmentation which augments the DenseNet classifier with ladder-style skip-connections. The purpose of skip-connections is to blend semantic information from the deep layers with spatially accurate information from the early layers. Contrary to the related previous work [27, 6, 19, 11, 25], our work is based on the DenseNet architecture [10] and performs the blending on the feature level by one projection, one concatenation and one 3 × 3 convolution, instead of complex residual processing at the class-score level. Contrary to the related previous work [12], our models are specifically tailored to support operation on very large natural images, by having few feature maps at high spatial resolution and many feature maps in the most downsampled layers. Consequently, we are able to perform the training on entire images in the megapixel range without high-end hardware, while outperforming dilated and residual architectures. We present experiments on Cityscapes and Pascal VOC 2012 datasets and report competitive IoU performance, modest memory requirements and fair execution speed.

2. Related Work

Early convolutional models for semantic segmentation had only few pooling layers and trained from scratch [5]. Later work built on image classification models pretrained on ImageNet [30, 8, 10], which typically perform 5 downsamplings before aggregation. The resulting loss of spatial resolution requires special techniques for upsampling the features back to the resolution of the source image. Some researchers approached the problem by trained upsampling [21]. This idea has been further improved by taking into account cached switches from strided pooling layers [23, 13].

A simpler approach to recover some resolution is to set the output stride of some pooling layers to one pixel. The resulting feature tensor must be subsequently reshaped in a *space to batch* fashion in order to make subsequent convolutions equivalent to the case of strided pooling which was used during pretraining [28, 35]. Thus, a regular convolution on the reshaped tensor becomes equivalent to the dilated [36] convolution on the original tensor. Besides upsampling, dilated filtering has also been applied to increase the receptive field of pixel-level classification [1, 36]. In theory, this approach can completely recover the resolution loss due to strided pooling, without any compromises with respect to the pretrained classification model [28, 35]. However, due to reasons we shall revisit later, practical implementations recover only the last two strided poolings, which allows subsequent inference at 8× subsampled resolution [35, 37, 34]. Wu et al [35] recall the relative shortness of most residual network training paths [33] and argue that models with reduced depth and increased semantic dimensionality (i.e. more feature maps) lead to more efficient memory utilization. They show ResNet experiments in which a 38-layer model outperforms deeper models. Zhao et al [37] pool features in a pyramidal fashion in order to recover a set of representations with increasingly larger receptive fields. They also introduce an auxiliary loss in the middle of the 4th group of the ResNet architecture. Due to dilated convolution, both losses operate at the same resolution. Wang et al [34] recover the resolution loss due to subsampling by explicitly trading semantic dimensionality for spatial resolution. This is achieved by applying a standard convolutional layer with $8^2{\cdot}$C kernels to the ResNet feature tensor W/8×H/8×2048. The resulting tensor W/8×H/8×$8^2{\cdot}$C is reshaped to W×H×C and finally employed for pixel-level inferrence.

Despite the success of dilated filtering, we believe that there are two important reasons why this technique is not likely going to be a definitive solution for upsampling semantic segmentation maps. First, dilated filtering substantially increases memory requirements. For instance, a VGG-D architecture (16 trainable layers) would require around 33 GB to store all convolutional feature maps at the Cityscapes resolution, which would preclude experimentation on most available hardware. Recent architectures [8, 10] additionally increase the memory pressure due to greater depth and batchnorm regularization. Second, dilated filtering treats semantic segmentation exactly as if it were ImageNet classification, which, in our view, should not be the case. Semantic segmentation has to provide accurate location information: one pixel left or right must make a difference between one class and another. We find it hard to accept that deep semantic layers alone are the optimal place for bringing such location-dependent decisions, and that brings us close to the focus of our research.

We shall now consider semantic segmentation approaches which upsample feature maps without dilated convolutions and are therefore more related to our work. Ghiasi et al [6] upsample the subsampled feature maps by predicting coefficients of 10 per-class basis functions which were trained without supervision using PCA. They additionally propose a VGG-based architecture with lateral skip-connections between class scores at three different resolutions (subsampling ×4, ×2 and ×1). The blending is performed by a custom function which prefers the deeper layer in the middle of the object, while favouring the earlier layer near the object boundary. The architecture is trained by simultaneously optimizing the cross-entropy loss at each resolution. Pohlen et al [26] propose a two-stream residual architecture where one stream is always at the full resolution, while the other stream has pooling and unpooling layers. The two streams blend in each convolutional group and pass through multiple residual connections. Experiments have shown that this kind of blending is able to considerably improve the semantic segmentation performance over the single-stream baseline. Lin et al [19] propose a ResNet-based ladder-style architecture with four lateral connections at subsampling levels ×4, ×8, ×16 and ×32. The blending is performed as a sum of the coresponding early layer and the upsampled result of the previous blended layer. The resulting tensor subsequently passes through a complex residual network with three successions of 5×5 max-poolings (stride 1) and 3×3 convolutions. The final blended tensor is obtained by processing the sum of these four tensors with two additional residual convolutions. Similar architectures with ResNet front-end and very complex ladder-style blending have been proposed in [11, 25]. Jégou et al [12] propose a DenseNet-based ladder-style architecture with five lateral connections at subsampling levels ×1 through ×32. The blending is performed by concatenation, and the resulting tensor is further processed by a complex dense block. They present very good results in experiments on CamVid and Gatech datasets although their models have not been pretrained on ImageNet.

Similarly to [6, 19], we optimize loss at different resolutions of the generated semantic maps. Our experiments confirm advantages of such design and agrees with the findings in the previous work. Our ladder-style blending is similar to [27] who, however, do not use DenseNet architecture and do not present experiments on Pascal and Cityscapes datasets. Our latteral connections differ from [6, 11, 25], since they blend class scores while we blend abstract features. Our approach has a modelling advantage, since the classes often share features. Our architecture is much simpler than any of [6, 19, 11, 25]. Our blending transformation consists of one projection, one concatenation and one 3×3 convolution instead of softmax and custom boundary masking [6], 9 convolutions and 3 poolings [19] or 4-12 convolutions [12]. Additionally, none of the previous work succeeds to achieve competitive performance by leveraging 64× subsampled representation.

To the best of our knowledge, there is only one other work [12] on DenseNet-based semantic segmentation. However, their upsampling and downsampling datapaths are symmetric, that is, they have exactly equal complexities in number of parameters and feature maps. We believe that such design is suboptimal because, intuitively, patch-level classification should be more difficult than finding the object boundary by blending. If we accept that the most subsampled tensor is capable to achieve high semantic quality (as suggested by ImageNet classification results), then it seems clear that upsampling transformation should not be overly complex. Besides representing a useful regularization factor, architecture simplicity becomes especially important when very large images need to be processed, due to scarcity of the GPU memory.

3. The proposed architecture

The proposed semantic segmentation architecture consists of two horizontal datapaths as shown in Figure 1. The downsampling datapath (top rail in Fig. 1) is characteristic for image classification: it recovers the abstract image representation by gradually reducing spatial resolution and increasing semantic dimensionality. The upsampling datapath (bottom rail in Fig. 1) is characteristic for semantic segmentation: it gradually transforms semantic vectors to the pixel-level posterior distributions over classes. This transformation is achieved by efficient blending of context-aware semantic representations from deeper layers with well-localized low-level representations from early layers.

The downsampling datapath corresponds to a slightly adapted DenseNet architecture [10]. The blocks labeled as DB and TD correspond to the dense blocks and transition layers from the original DenseNet. We split the fourth dense block (DB4) into two halves (DB4a and DB4b) and introduce an additional average-pooling layer (D) between them. This change increases the receptive field of the DB4b features and saves some GPU memory. The resulting loss of resolution is restored with ladder-style upsampling, in the same way as in other parts of the architecture. We initialize the DB4b filters from the original model trained on ImageNet, although the introduced pooling layer alters the feature tensor in a way that has not been seen during training. However, further training on the target dataset succeeds to recover from this anomaly and achieve high performance on test sets of semantic segmentation datasets. At the end of DB4 we have a 64x subsampled representation. We feed this representation to the context classification layer which consists of a projection to 512 dimensions followed by a 3x3 convolution with 128 filters dilated with rate=2. The role of this layer is to capture wide context information.

Subsequently, we feed the context representation to the upsampling datapath of the ladder architecture in order to recover fine details lost due to the downsampling. The upsampling datapath consists of transition-up blocks labeled as TU. The goal of TU blocks is to blend two representations whose spatial resolutions differ by a factor of 2. The smaller representation comes from the upsampling datapath while the larger representation comes from the downsampling datapath of the ladder architecture. To achieve the

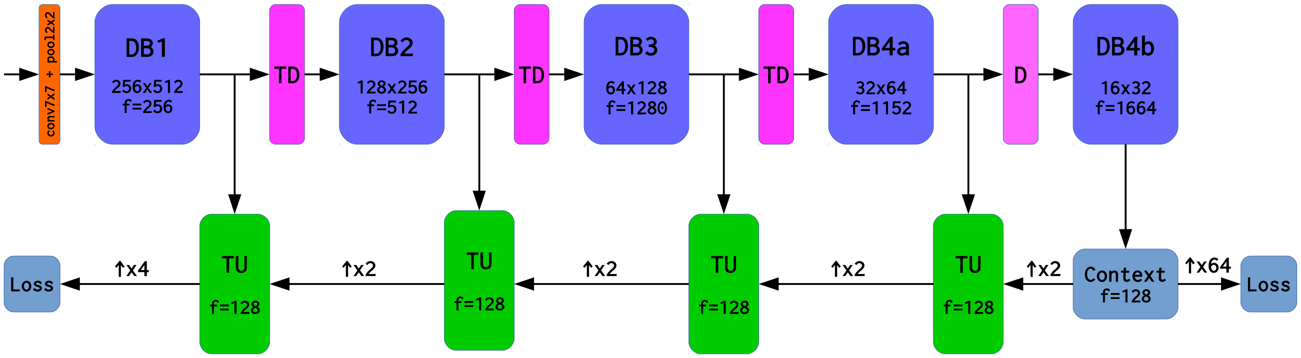

Figure 1. Architecture of the proposed segmentation network with DenseNet-169 in the downsampling datapath. Each dense block (DBx) is annotated with the corresponding resolution assuming the Cityscapes resolution of 1024x2048. Number of output feature maps is denoted with f. Transition-up (TU) blocks perform blending of low-resolution high-level features and high-resolution low-level features.

blending, we first upsample the smaller representation with bilinear interpolation so that the two representations have the same resolution. Subsequently, we project the larger representation to a lower-dimensional space so that the two representations have the same number of feature maps. This saves memory and helps to balance the relative influence of the two datapaths. Finally, we concatenate the two representations and blend them by applying a single $3{\times}3$ convolution. We have tried to increase the capacity of the blending element, however these experiments did not bring significant accuracy improvements. This led us to conclude that representation blending is an easy task and that further improvements should be sought by improving the downsampling datapath of the ladder architeture.

The blending procedure is recursively repeated along the upsampling datapath with skip-connections arriving from the outputs of each dense block. Finally we retrieve semantic segmentation at the resolution produced by the DeneseNet stem (4 times subsampled input resolution) and use bilinear upsampling to recover dense predictions at the input resolution.

4. DenseNet vs ResNet comparison

Recent classification architectures have a common high-level structure in which processing blocks are intertwined with downsampling elements (cf. top rail in Fig. 1). Each processing block groups a number of convolutional processing units operating on the common resolution. We illustrate internal organizations of the convolutional units in two prominent recent architectures (ResNet and DenseNet) in Fig.2. The number of convolutional units in a block is designated with n. F_{in} and F_{out} denote semantic dimensionality at input and output of the processing block. ResNet units operate on a sum of previous unit and the input. The information path in a ResNet block therefore has constant dimensionality F_{out} (in implementations, the first unit of a block often increases the semantic dimensionality by projection). On the other hand, DenseNet units operate on a concatenation of the input with the output of all preceeding units in the current block. Thus, the dimensionality of a DenseNet block increases after each convolutional unit. The number of feature maps produced by each DenseNet unit is called the growth rate of a DenseNet architecture and is defined by the parameter k (all DenseNet variations used here have k=32).

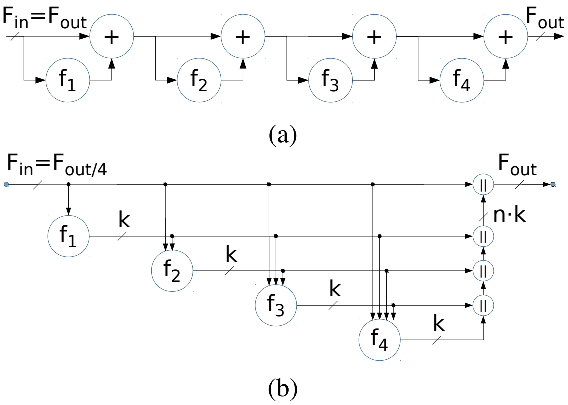

Figure 2. ResNet block with n=4 units (a) and the corresponding DenseNet-BC block (b). All connections are 3D tensors W×H×D, where D is designated above the connection line.

In order to reduce the computational complexity, both ResNet and DenseNet units f_i reduce the semantic dimensionality of the input tensor before performing standard 3×3 convolutions. ResNet reduces the dimensionality to $F_{out}/4$, while DenseNet goes to $4 \cdot k$. Therefore, DenseNet units have two convolutions (1x1, 3x3) while ResNet Units require three convolutions (1x1, 3x3, 1x1) in order to restore the dimensionality used in residual connections. The ResNet convolution kernels have the following shapes: $1{\times}1{\times}F_{out}{\times}F_{out}/4$, $3{\times}3{\times}F_{out}/4{\times}F_{out}/4$, and $1{\times}1{\times}F_{out}/4{\times}F_{out}$. The shapes of DenseNet convolutions are: $1{\times}1{\times}\,[F_{in} + (i-1)\cdot 4\text{k}]\,{\times}\text{k}$, and $3{\times}3{\times}4\text{k}{\times}\text{k}$. Note finally that each convolution is preceded (DenseNet) or followed (ResNet) by one batchnorm and one ReLU layer.

Both architectures encourage exposure of early layers to the loss signal. However, the distribution of the representation dimensionality differs: ResNet keeps the dimensionality constant throughout the block, while DenseNet increases it towards the end. A straight-forward analysis would show that DenseNet blocks have several computational advantages including i) producing less feature maps on output (k vs F_{out}), ii) less caching of input tensors due to heavy reuse of the feature maps ($F_{out} < nF_{out}$), iii) lower average input dimensionality despite larger total output dimensionality, iv) less convolution layers: 2 vs 3.

However, current DenseNet variants have more convolutional units than their ResNet counterparts with similar recognition performance. For instance, the four blocks of ResNet-50 have n=$[3, 4, 6, 3]$, while DenseNet-121 blocks have n=$[6, 12, 24, 16]$. Thus, in practice, it turns out that these two architectures have roughly equal memory requirements and achieve similar execution speed.

5. Experiments

We evaluate our method on two different semantic segmentation datasets: Cityscapes [2] and Pascal VOC2012 [4]. The Cityscapes dataset is a recent semantic segmentation benchmark which contains outdoor traffic scenes with 19 classes recorded in 50 cities during spring, summer and autumn. The dataset features good and medium weather conditions, large number of dynamic objects, varying scene layout and varying background. It consists of 5000 images with fine annotations and 20000 images with coarse annotations. In experiments we use only the fine annotations. The resolution of the images is 1024×2048.

Pascal VOC2012 [4] contains 1464 training images and 1449 validation images. Following the common practice, in some experiments we extend the data with the augmented set [7]. All images are semantically annotated at pixel level with 20 object classes and the additional background class.

5.1. Implementation Details

We train our networks using Adam [14] with a base learning rate of $5e^{-4}$ and use poly learning rate policy which multiplies the initial learning rate by $(1-\frac{iter}{max\_iter})^z$ where $z = 1.5$. Additionally dividing the learning rate by a factor of 5 for the ImageNet pre-trained network part yields a small increase in accuracy. We observe that batch size is an important hyper-parameter of the optimization procedure. If we train with batches of single images, batch normalization statistics (mean and variance) will fit exactly the image we are training on. This may cause a covariance shift across different images [31], so we train with two entire images in the batch. In order to make this feasible on Cityscapes images, we distribute our model across two GPU cards. Dense blocks DB1 and DB2 are placed on GPU #1, while the rest of the model is placed on GPU #2. We define two cross entropy losses as shown in Figure 1. The main loss is placed at the output of the last transition-up unit, while the auxiliary loss is placed at the output of the context layer. The main loss is multiplied by a factor of 0.7 and auxiliary by a factor of 0.3. We employ random image flipping as the only jittering technique on both datasets.

5.2. Cityscapes

Table 1 shows the performance figures obtained on the Cityscapes validation set while training on 2× subsampled images. We compare our ladder-style upsampling with dilated convolutions, evaluate DenseNet-121, DenseNet-169 and ResNet-50 architectures, and explore the influence of the auxiliary loss and dense block splitting. The configuration LadderDenseNet-169 4× corresponds to the architecture described in Figure 1. DenseNet-121 32× is a baseline result without the upsampling datapath, where we simply took 32× subsampled output from the context block. In the configuration Dilated8×DenseNet-121 4× we first produced 8× subsampled representation by dilating the last two dense blocks and then arrived at 4× resolution with one TU layer. We observe that full ladder-style upsampling brings more than 1pp mIoU with respect to the dilated filtering. The experiment labeled single loss shows that the auxiliary loss contributes around 1pp mIoU. We also illustrate the contribution of the auxiliary loss in Fig.3. In split3+4 we have inserted additional pooling and transition-up in DB3 which enabled downsampling to 128×. In SPP we tried spatial pyramid pooling similar to the one used in [37] instead of our simple context block. Interestingly, this did not bring any improvement, and we hypothesize that is a consequence of heavy subsampling in our architecture. In the configuration Ladder ResNet-50 4× we have replaced the

Method	mIoU(%)	PixAcc
ResNet-50 32×	60.95	91.98
LadderResNet-50 4×	69.53	94.42
Dilated8×ResNet-50 4×	69.23	94.47
DenseNet-121 32×	62.52	92.29
Dilated8×DenseNet-121 4×	71.56	94.92
LadderDenseNet-121 4× single loss	71.73	94.99
LadderDenseNet-121 4×	72.82	95.06
LadderDenseNet-169 4× split3+4	72.69	95.08
LadderDenseNet-169 4× SPP	72.34	95.04
LadderDenseNet-169 4×	**73.72**	**95.11**

Table 1. Segmentation results on Cityscapes val. Configurations labeled n× denote results with n× times subsampled predictions. All training and evaluation images in this experiment were resized to 1024×448, while batch size was set to 4.

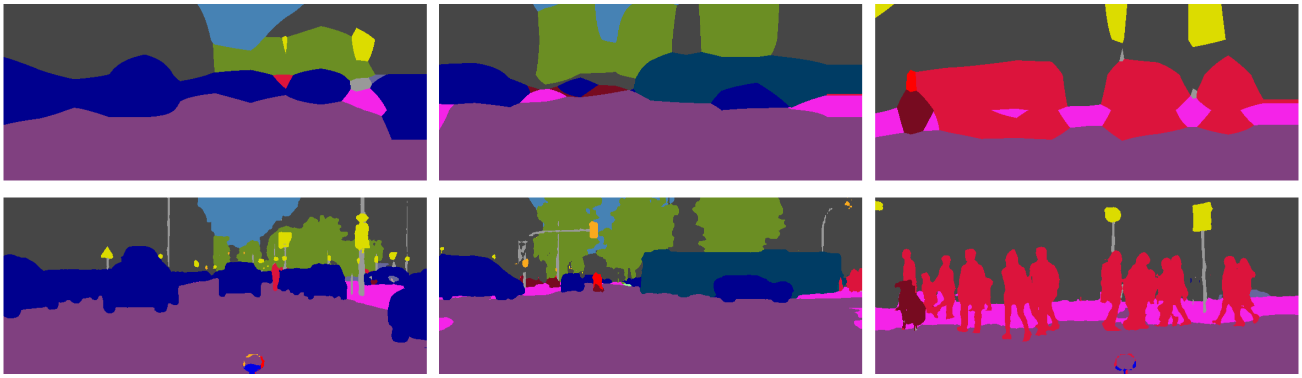

Figure 3. Predictions obtained from 64× subsampled auxiliary loss (top) and 4× subsampled final prediction after upsampling (bottom). We observe that the transition-up (TU) layers are able to recover small objects and fine details lost due the subsampling.

DenseNet-121 classifier with the corresponding ResNet-50 network likewise pre-trained on ImageNet. We observe that DenseNet achieves a significant improvement over ResNet despite having less parameters and somewhat larger error on the ImageNet dataset (cf. Table 2).

Method	Top-1(%)	Parameters
Resnet-101	23.6	45M
Resnet-50	24.7	25M
DenseNet-169	23.6	13M
DenseNet-121	25.0	8M

Table 2. Classification results on ImageNet. We compare the achieved performance with the number of parameters.

Table 3 shows that training from scratch results in only 10pp performance loss with respect to the ImageNet pre-trained model, which is again better than ResNet.

Method	Mean IoU(%)	Pixel Acc.(%)
Ladder Resnet-50 4×	55.49	92.29
Ladder DenseNet-121 4×	**62.32**	**93.63**

Table 3. Segmentation results on Cityscapes val. The models were trained from scratch (i.e. without ImageNet pretraining).

Table 4 shows the results when training on full Cityscapes images. We first initialized the weights from the model trained on smaller resolution and in experiment labeled as fine-tune just continued training on large images. In experiment frozen-BN we freeze all BatchNorm layers and put them in inference mode during training. This significantly improved the speed and reduced the required memory, while yielding similar accuracy as above.

Method	mIoU(%)	PixAcc
Ladder DenseNet-169 4× frozen-BN	75.24	96.02
Ladder DenseNet-169 4× fine-tune	75.75	95.88

Table 4. Segmentation results on Cityscapes val after training on full 2048×1024 resolution. Batch size was set to 2.

Table 5 shows the results on the Cityscapes test set. We have achieved 74.55% IoU and 51.54% iIoU, which outperforms RefineNet [19] (73.6% IoU, 47.2% iIoU) despite a smaller baseline architecture (DenseNet-169 vs ResNet-101). A lighter recent approach [24] reported accuracy below 60 mIoU. Our submission at the benchmark web site is labeled Ladder DenseNet.

Method	base architecture	mIoU(%)
Dilation10 [36]	VGG-16	67.1
LRR 4× [6]	VGG-16	69.7
RefineNet [19]	ResNet-101	73.6
TUSimple [34]	ResNet-152	76.1
LargeKernel† [25]	ResNet-152	76.9
PSPNet [37]	ResNet-101	78.4
Ladder DenseNet-169 4× (ours)	DenseNet-169	74.6

Table 5. Segmentation results on Cityscapes test after training on fine annotations only (except †). Base architectures have the following numbers of parameters (in millions): DenseNet121: 8, DenseNet169: 13, ResNet50: 25, ResNet101: 45, ResNet152: 60.

5.3. Pascal VOC2012

Table 6 shows the results on the PASCAL VOC 2012 validation dataset. In the experiment labeled AUG we trained on the union of train and augmented sets, while in all other experiments we have trained the model only on the original train set. Here again DensetNet gives a large increase in accuracy when compared to ResNet. We additionally trained the same proposed architecture on Pascal VOC2012 train+val+aug data and obtained 78.25% mean IoU on the test set. Our submission is labeled Ladder_DenseNet at the web site of the benchmark.

6. Conclusion

We have presented a ladder-style adaptation of the DenseNet architecture for real-time semantic segmentation of large natural images. The proposed design uses lateral

Method	mIoU(%)	PixAcc(%)
LadderResNet-50 4×	62.97	91.56
LadderDenseNet-169 4×	70.21	93.29
LadderDenseNet-169 4× AUG	78.01	95.11

Table 6. Segmentation results on Pascal VOC2012 val. Predictions were generated at 4× subsampled resolution and then upsampled by bilinear interpolation.

skip connections to blend higher level features at lower spatial resolution with their lower-level higher-resolution counterparts. These connections relieve deep representational levels from the necessity to forward low-level details and allow them to focus on abstract invariant features. The resulting architecture is able to perform accurate semantic segmentation with a very lean upsampling path. This significantly reduces memory requirements and allows us to perform experiments on large natural images.

The presented implementation is able to perform the forward-pass on entire 2048×1024 images with a single GPU. It is also able to perform the backward pass on batches of two 2048×1024 images on two GTX 1070 (2×8GB) in the case of DenseNet-121, and one Titan X and one GTX 1070 (12GB+8GB) in the case of DenseNet-169. The model based on DenseNet-121 performs forward pass in real-time (31 Hz) on half-resolution images (1024×448) on a single Titan X GPU. To the best of our knowledge, this is the first account of applying DenseNet architecture for dense prediction at Cityscapes resolution, as well as recovering highly accurate semantic segmentations on half megapixel images in real-time.

We have performed experiments on Cityscapes and Pascal VOC 2012 validation and test sets. In both cases our best results came close to the state-of-the art mIoU performance. Ablation experiments confirm the utility of the ladder-style design as well as the advantage of DenseNet concatenations over residual connections.

Acknowledgments

This work has been fully supported by Croatian Science Foundation under the project I-2433-2014. The Titan X used in experiments was donated by NVIDIA Corporation. We thank anonymous reviewers on valuable comments.

References

[1] 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. 2

[2] M. Cordts, M. Omran, S. Ramos, T. Scharwächter, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset. In *CVPRW*, 2015. 5

[3] G. Csurka and F. Perronnin. An efficient approach to semantic segmentation. *International Journal of Computer Vision*, 95(2):198–212, 2011. 1

[4] M. Everingham, S. M. A. Eslami, L. V. Gool, C. K. I. Williams, J. M. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015. 5

[5] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1915–1929, 2013. 1, 2

[6] G. Ghiasi and C. C. Fowlkes. Laplacian pyramid reconstruction and refinement for semantic segmentation. In *ECCV*, pages 519–534, 2016. 2, 3, 6

[7] B. Hariharan, P. Arbelaez, L. D. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *ICCV*, pages 991–998, 2011. 5

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 1, 2

[9] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. *CoRR*, abs/1603.05027, 2016. 1

[10] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2016. 1, 2, 3

[11] M. A. Islam, M. Rochan, B. Neil D. B. and Y. Wang. Gated feedback refinement network for dense image labeling. In *CVPR*, July 2017. 2, 3

[12] 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. 2, 3

[13] 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. 2

[14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 5

[15] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *NIPS*, pages 109–117, 2011. 1

[16] I. Kreso, D. Causevic, J. Krapac, and S. Segvic. Convolutional scale invariance for semantic segmentation. In *GCPR*, pages 64–75, 2016. 1

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114, 2012. 1

[18] G. Larsson, M. Maire, and G. Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. In *ICLR*, 2017. 1

[19] G. Lin, A. Milan, C. Shen, and I. D. Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *CVPR*, 2017. 2, 3, 6

[20] G. Lin, C. Shen, A. van den Hengel, and I. D. Reid. Efficient piecewise training of deep structured models for semantic segmentation. In *CVPR*, pages 3194–3203, 2016. 1

[21] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015. 1, 2

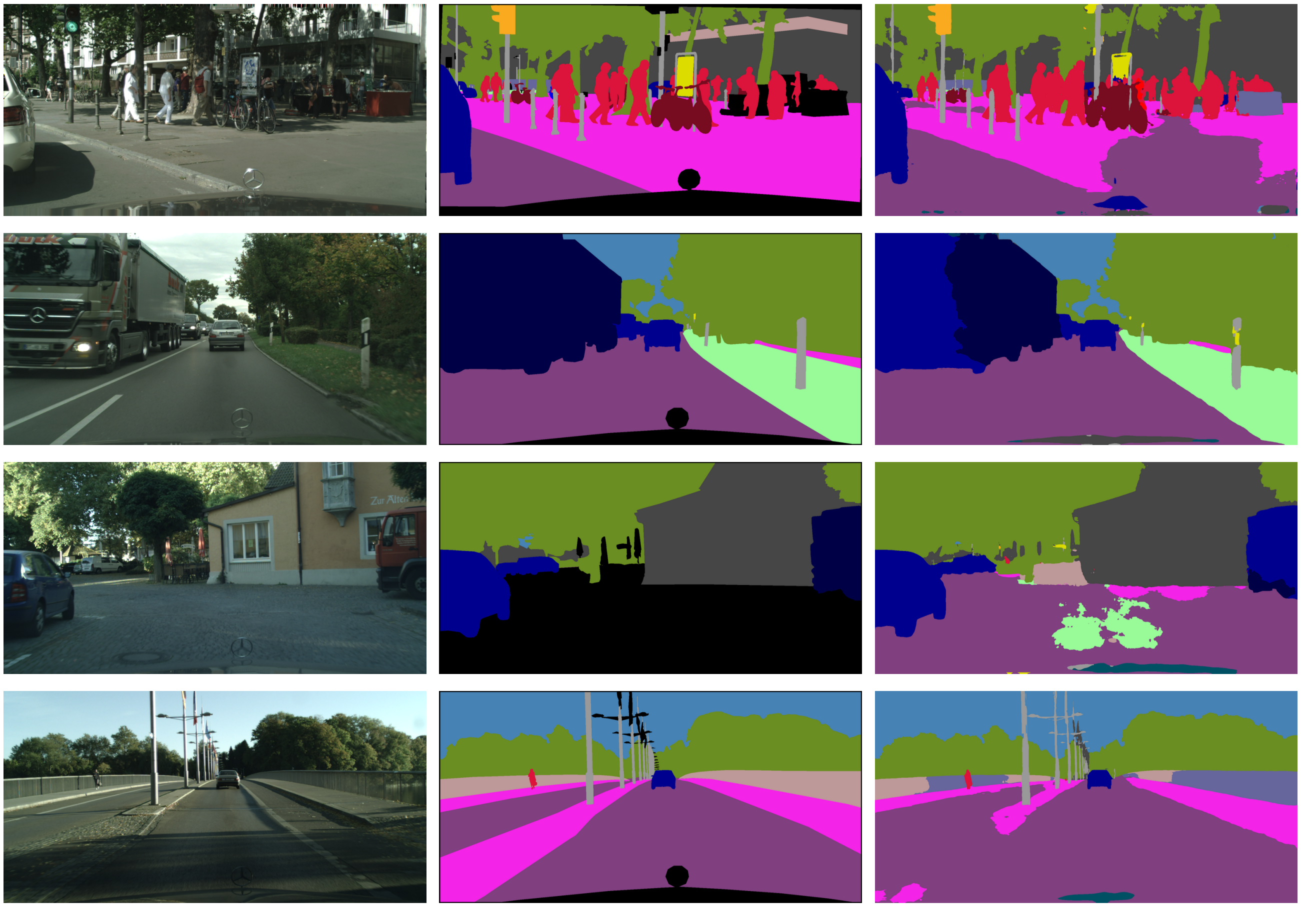

Figure 4. Cityscapes validation images with **worst** mIoU. From left to right: input, ground-truth, prediction. We observe largest problems with ambiguous sidewalks (row 1), large objects such as trucks (row 2), occluded objects (row 3), and rare classes such as fence (row 4).

[22] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich. Feedforward semantic segmentation with zoom-out features. In *CVPR*, pages 3376–3385, 2015. 1

[23] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *ICCV*, 2015. 2

[24] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *CoRR*, abs/1606.02147, 2016. 6

[25] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun. Large kernel matters - improve semantic segmentation by global convolutional network. In *CVPR*, July 2017. 2, 3, 6

[26] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe. Full-resolution residual networks for semantic segmentation in street scenes. In *CVPR*, July 2017. 3

[27] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. 2, 3

[28] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, pages 1–16, 2014. 2

[29] A. Shrivastava, R. Sukthankar, J. Malik, and A. Gupta. Beyond skip connections: Top-down modulation for object detection. *CoRR*, abs/1612.06851, 2016. 2

[30] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, pages 1–16, 2014. 1, 2

[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. In *CVPR*, 2015. 1, 5

[32] H. Valpola. From neural PCA to deep unsupervised learning. *CoRR*, abs/1411.7783, 2014. 1, 2

[33] A. Veit, M. J. Wilber, and S. J. Belongie. Residual networks behave like ensembles of relatively shallow networks. In *NIPS*, pages 550–558, 2016. 1, 2

[34] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. Cottrell. Understanding Convolution for Semantic Segmentation. *CoRR*, abs/1702.08502, 2017. 2, 6

[35] Z. Wu, C. Shen, and A. van den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *CoRR*, abs/1611.10080, 2016. 2

[36] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, pages 1–9, 2016. 2, 6

[37] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *ICCV*, 2017. 2, 5, 6