# Efficient Ladder-style DenseNets for Semantic Segmentation of Large Images

Ivan Krešo Josip Krapac Siniša Šegvić
University of Zagreb, Faculty of Electrical Engineering and Computing
ikreso@realnetworks.com josip.krapac@zalando.de sinisa.segvic@fer.hr

**Abstract**—Recent progress of deep image classification models has provided a great potential for improving 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 input resolution is in the megapixel range. To address these concerns, we propose a novel ladder-style DenseNet-based architecture which features high modelling power, efficient upsampling, and inherent spatial efficiency which we unlock with checkpointing. The resulting models deliver high performance and allow training at megapixel resolution on commodity hardware. The presented experimental results outperform the state-of-the-art in terms of prediction accuracy and execution speed on Cityscapes, VOC 2012, CamVid and ROB 2018 datasets. Source code at https://github.com/ivankreso/LDN.

## 1 Introduction

Semantic segmentation is a computer vision task in which a trained model classifies pixels into meaningful high-level classes. Due to being complementary to object localization, it represents an important step towards advanced image understanding. Most attractive applications include autonomous control [1], intelligent transportation [2], photo editing [3] and medical imaging [4]. Early approaches optimized a trade-off between multiple local classification cues (texture, color etc), and their global agreement [5]. Later work improved these ideas with non-linear embeddings [6], multi-scale analysis [7], depth [8], and improved spatial consistency [9], [10]. However, none of these approaches has been able to match the improvement due to deep convolutional models [7], [11].

Deep convolutional models have caused an unprecedented rate of computer vision development. Much recent attention has been directed towards residual models (also known as ResNets) [12], [13] in which each processing step is expressed as a sum between a compound non-linear unit and its input. This introduces an auxiliary information path which allows a direct gradient propagation across the layers, similarly to the flow of the state vector across LSTM cells. Recent approaches replicate and exceed the success of residual models by introducing skip-connections across layers. Our work is based on densely connected models (also known as DenseNets) [14] in which the convolutional units operate on concatenations of all previous features at the current resolution. This encourages feature sharing and discourages overfitting [14], while also favouring the gradient flow towards early layers. Our semantic segmentation experiments show that DenseNet-based models outperform their counterparts based on ResNets [13] and more recent dual path networks [15]. Besides outstanding generalization, DenseNets carry a great memory-saving potential due to extensive feature reuse. However, this potential is not easily materialized due to inefficient activation caching during automatic differentiation. We show that this can be effectively alleviated by extensive gradient checkpointing [16] which leads to five-fold reduction of memory requirements.

In general, deep segmentation models must decrease the spatial resolution of deep layers in order to enable training under strict GPU memory limitations. Subsequently, deep features have to be carefully upsampled in order to generate correct predictions at semantic borders and small objects. Most previous work reduces the subsampling by dilated filtering [17], [18], [19], [20], [21]. We take another approach, where deep features are upsampled by exploiting activations from earlier layers. Different than previous such methods [4], [22], [23], [24], [25], we conjecture that upsampling requires much less capacity than the downsampling path. Consequently, we propose a minimalistic upsampling datapath which is very well suited for efficient processing of large images.

This paper presents an effective lightweight architecture for semantic segmentation of large images, based on DenseNet features and ladder-style [26] upsampling. We propose several improvements with respect to our previous work [27], which lead to better accuracy and faster execution while using less memory and fewer parameters. Our consolidated contribution is three-fold. First, we increase computational efficiency by subsampling the representation in the middle of a DenseNet block. Second, we propose an efficient ladder-style upsampling datapath which requires less memory and achieves a better $\overline{\text{IoU}}$/FLOP trade-off than related previous work [4], [22]. Third, we unlock the potential of our method for memory-efficient training with a novel checkpointing approach. Our method now requires less than half memory than ResNet with in-place activated batchnorm [20]. These contributions strike an excellent balance between prediction accuracy and model complexity. Experiments on Cityscapes, CamVid, ROB 2018 and Pascal VOC 2012 demonstrate state-of-the-art recognition performance and competitive inference speed.

## 2 Related Work

Early convolutional models for semantic segmentation had only a few pooling layers and were trained from scratch [7]. Later work built on feature extractors pre-trained on ImageNet [12], [14], [28], which produce 32× subsampled representations. This improved generalization [29] and reduced the training footprint due to smaller resolution of feature tensors. Early upsampling approaches leveraged trained filters [30] and cached max-pool switches [31]. Many recent approaches modify some strided layers to produce non-strided output while doubling the dilation factor of all subsequent convolutions [18], [32]. This decreases the extent of subsampling while preserving pre-trained semantics of the feature extractor.

However, we believe that dilated filtering should be used sparingly [33] or completely avoided [27], [34] due to following two shortcomings. First, dilated filtering substantially increases the training footprint and inference complexity, and thus hinders single-GPU training and real-time inference. Practical implementations alleviate this by recovering only up to the last two subsamplings, which allows subsequent inference at 8× subsampled resolution [17], [18]. Second, dilated filtering treats semantic segmentation exactly as if it were ImageNet classification: each pixel is classified independently from its neighbours. This does not feel right since we know that neighboring pixels are highly correlated in practice.

We therefore prefer to keep the downsampling and restore the resolution by blending semantics of deep features with location accuracy of the earlier layers [35]. This encourages the deep layers to discard location information and focus on abstract image properties [26]. Practical realizations avoid high-dimensional features at output resolution [35] by ladder-style upsampling [4], [26], [34]. In symmetric encoder-decoder approaches, [4], [22], [31] the upsampling datapath mirrors the structure of the downsampling datapath. These methods are unable to process large images or deliver high execution speed due to excessive upsampling complexity. Ghiasi et al [23] blend predictions (instead of blending features) by prefering the deeper layer in the middle of the object, while favouring the earlier layer near the object boundary. Pohlen et al [36] propose a two-stream residual architecture where one stream is always at the full resolution, while the other stream is first subsampled and subsequently upsampled by blending with the first stream. Lin et al [24] perform the blending by a sub-model comprised of 8 convolutional and several other layers in each upsampling step. Islam et al [37] blend upsampled predictions with two layers from the downsampling datapath. This results in 4 convolutions and one elementwise multiplication in each upsampling step. Peng et al [25] blend predictions produced by convolutions with very large kernels. The blending is performed by one 3×3 deconvolution, two 3×3 convolutions, and one addition in each upsampling step.

Contrary to previous approaches, we conjecture that it is much more difficult to recognize semantic classes than to delineate semantic borders. We therefore argue for an asymetric encoder-decoder approach consisting of a powerful backbone and efficient upsampling. Each upsampling step has only one 3×3 convolution whereby the number of feature maps is much lower than in the corresponding layers of the downsampling path [27]. To the best of our knowledge, such organization has not been previously used for semantic segmentation, although there exists related work in object detection [34], and instance-level segmentation [38]. Note that our lateral connections differ from [23], [25], [37], since they blend predictions, while we blend features. Blending features improves the modelling power, but is more computationally demanding. We can afford to blend features due to efficient upsampling and gradient checkpointing which will be explained later.

Pretrained convolutional representations are not directly applicable to large images due to insufficient receptive range. This issue typically shows up in large indistinctive regions which are common in urban scenes. These regions are often projected from nearby objects, which makes them very important for high-level tasks (as exemplified by circumstances of the first fatal incident of a level-2 autopilot). Some approaches address this issue with dilated convolutions [18], [33], however sparse sampling may hurt the accuracy due to aliasing. The receptive range can also be enlarged by increasing the model depth [37]. However, the added capacity may result in overfitting. Correlation between distant parts of the scene can also be modelled with long-range connections [10], [39]. However, these may be unsuitable due to large capacity and computational complexity. A better ratio between receptive range and complexity is achieved with spatial pyramid pooling (SPP) [17], [40], [41] which augments the features with their spatial pools over rectangular regions of varying size.

To the best of our knowledge, there are only two previous works on DenseNet-based semantic segmentation [21], [22]. However, these approaches fail to position DenseNet as the backbone with the largest potential for memory-efficient feature extraction. This potential is caused by a specific design which encourages inter-layer sharing [14] instead of forwarding features across the layers. Unfortunately, automatic differentiation is unable to exploit this potential due to concatenation, batchnorm and projection layers. Consequently, straightforward DenseNet implementations actually require a little bit more memory than their residual counterparts [27]. Here we show that the problem can be resolved with checkpointing [14], [16]. Previous work on checkpointed segmentation models considered only residual models [20], and achieved only two-fold memory reduction. On the other hand, our custom checkpointing scheme is specifically crafted for our architecture, and is therefore able to achieve up to six-fold memory reduction with respect to the baseline.

## 3 The Proposed Architecture

The proposed architecture consists of two datapaths which roughly correspond to two horizontal rails in Figure 1. The downsampling datapath includes a customized DenseNet feature extractor [14], and a lightweight spatial pyramid pooling module (SPP) [17]. The feature extractor transforms the input image into convolutional features $\mathbf{F}$ by gradually reducing spatial resolution, and increasing the number of feature maps (top rail in Fig. 1). The SPP module enriches

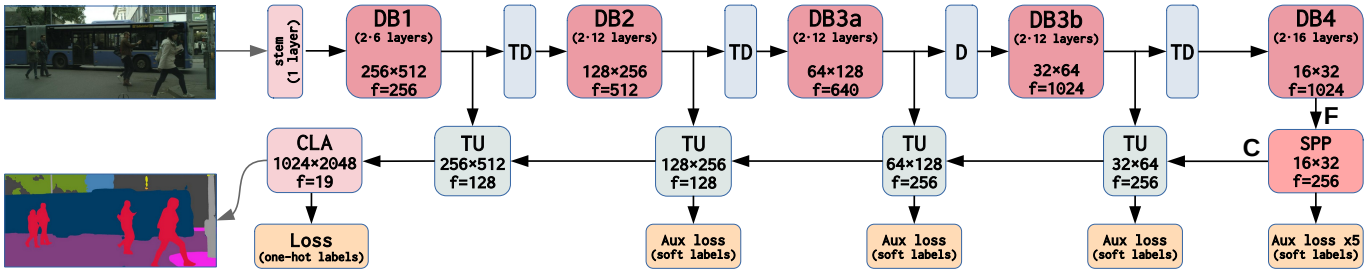


Fig. 1. Diagram of the proposed segmentation architecture. Each processing block is annotated with dimensions of the resulting feature tensor in case of DenseNet-121 and Cityscapes input (f denotes the number of feature maps). We split the dense block DB3 in order to enlarge receptive range, improve speed and reduce the memory footprint. The transition-up (TU) blocks blend low-resolution semantic features with high-resolution low-level features. The classifer (CLA) projects features to 19 semantic maps and bilinearly upsamples them to the input resolution.

$\mathbf{F}$ with context information, and produces context-aware features $\mathbf{C}$. The upsampling datapath transforms $\mathbf{C}$ into high-resolution semantic predictions (bottom rail in Fig. 1) by exploiting fine details from the downsampling datapath.

### 3.1 Feature extraction

Our feature extractor consists of densely connected blocks (DB) and transition layers (TD, D) (cf. Figure 1). Each DB is a concatenation of convolutional units, while each convolutional unit operates on a concatenation of all preceding units and the DB input [14]. We customize the original DenseNet design by splitting DB3 into two fragments (DB3a and DB3b), and placing a strided average-pooling layer (D) in-between them. This enlarges the receptive field of all convolutions after DB3a, while decreasing their computational complexity. In comparison with dilated filtering [18], this approach trades-off spatial resolution (which we later restore with ladder-style blending) for improved execution speed and reduced memory footprint. We initialize DB3b filters with the original ImageNet-pretrained weights, although the novel pooling layer alters the features in a way that has not been seen during ImageNet pretraining. Despite this discrepancy, fine-tuning succeeds to recover and achieve competitive generalization. The feature extractor concludes by concatenating all DB4 units into the 64× subsampled representation $\mathbf{F}$.

### 3.2 Spatial pyramid pooling

Spatial pyramid pooling captures wide context information [17], [40], [41] by augmenting $\mathbf{F}$ with average pools over spatial grids. We first project $\mathbf{F}$ to d/2 maps, where d denotes the dimensionality of features in $\mathbf{F}$. The resulting tensor is then average-pooled over four grids with 1, 2, 4, and 8 rows. The number of grid columns is set in accordance with the image size so that all cells have a square shape. We project each pooled tensor to d/8 maps and then bilinearly upsample to the common resolution. We concatenate all results with the projected $\mathbf{F}$, and finally blend with a 1×1×d/4 convolution. The resulting context-aware feature tensor $\mathbf{C}$ is h×w×d/4 where h=H/64, w=W/64, while H,W denotes the input resolution. If we use DenseNet-121 (d=1024), $\mathbf{C}$ has 48 times less dimensions than the input image.

### 3.3 Ladder-style upsampling

Ladder-style upsampling recovers fine details through a series of efficient transition-up (TU) blocks. Each TU block blends the preceding representation along the upsampling path ($\mathbf{U_{i-1}}$) with a skip-connection from the corresponding densely connected block ($\mathbf{DBx}$). $\mathbf{U_{i-1}}$ has strong semantics while $\mathbf{DBx}$ has more spatial detail. The output $\mathbf{U_i}$ gets the best of both worlds. We first upsample $\mathbf{U_{i-1}}$ so that the two representations have the same resolution. Subsequently, we project $\mathbf{DBx}$ to a lower-dimensional space so that the two representations have the same number of feature maps. This balances the relative influence of the two datapaths and allows blending by simple summation. Subsequently, we reduce the dimensionality of the sum with 1×1 convolution, and blend the result with 3×3 convolution to deliver the output $\mathbf{U_i}$ as shown in Fig. 3. The described blending procedure is recursively applied along the upsampling datapath. The last transition-up block produces features at the resolution of the DenseNet stem. Finally, the CLA module recovers dense predictions by projection onto semantic classes and 4× bilinear upsampling. The proposed

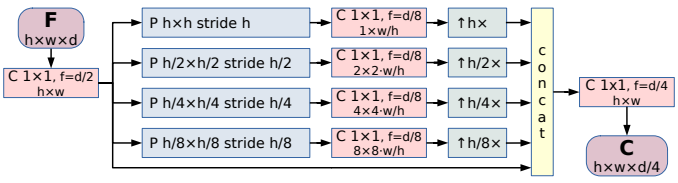


Fig. 2. The proposed SPP module. P, C, and ↑ denote pooling, convolution and bilinear upsampling. We introduce two modifications with respect to [17]: i) we adapt the grid to the aspect ratio of input features $\mathbf{F}$, and ii) we reduce the dimensionality throughout the module in order to decrease memory footprint and discourage overfitting.

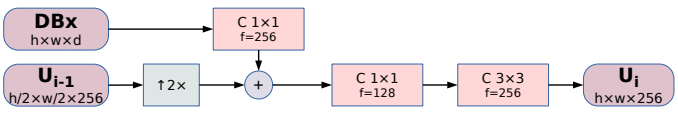


Fig. 3. The proposed transition-up module (UP in Fig. 1). C, and ↑ denote convolution and bilinear upsampling. The number of feature maps in $\mathbf{U_{i-1}}$ and $\mathbf{U_i}$ varies according to Fig. 1. Minimalistic design ensures fast inference (only one 3×3 convolution), and low memory footprint (few convolutions, low feature dimensionality).

design has much fewer parameters than the downsampling datapath and therefore discourages overfitting as we show in the experiments.

## 4 Gradient checkpointing

Semantic segmentation requires a lot of caching during backprop, especially for large input resolutions. This may lead to difficulties due to strict limitations of GPU RAM. For example, it is well known that small training batches result in poor learning due to unstable batchnorm statistics. This problem can not be overcome by accumulating backward passes, and therefore hinders the accuracy of the model.

The extent of backprop-related caching can be reduced with gradient checkpointing [16]. The main idea is to instruct forward pass to cache only a carefully selected subset of all activations which are denoted as gradient checkpoints. The subgraph between two gradient checkpoints is denoted as a checkpointing segment. The backward pass iterates over all checkpointing segments and processes them as follows: i) forward pass activations are recomputed starting from the checkpoint, and ii) the gradients are computed via the standard backward pass. The local cache is released as soon as a segment is processed, before continuing to the next segment.

We note that segment granularity affects space and time efficiency. Enlarging the checkpoint segments always reduces the memory footprint of the forward pass. However, the influence to backward pass memory requirements is non-trivial. Larger segments require more memory individually as they need to re-compute all required activations and store them in the local cache. At some point, we start to lose the gains obtained during forward pass. Our best heuristic was to checkpoint only outputs from 3×3 convolutions as they are the most compute-heavy operations. In other words, we propose to re-compute the stem, all projections, all batchnorms and all concatenations during the backward pass. Experiments show that this approach strikes a very good balance between maximum memory allocation in forward and backward passes.

The proposed checkpointing strategy is related to the previous approach [14] which puts a lot of effort into explicit management of shared storage. However, here we show that similar results can be obtained by relying on the standard PyTorch memory manager. We also show that custom backprop operations can be completely avoided by leveraging the standard PyTorch module `torch.utils.checkpoint`. Finally, we propose to achieve further memory gains by checkpointing outputs of 3×3 convolutions. This implies re-computing the stem, transition-down and transition-up blocks, and each DenseNet unit except the 3×3 convolution. To the best of our knowledge, this is the first account of applying extensive checkpointing for semantic segmentation.

## 5 Experiments

Most of our experiments target road-driving images since the corresponding applications require large input resolution (subsections 5.2, 5.3, and 5.4). For completeness, we also present results on photographic collections (5.5), and ablation experiments (5.6). Finally, we show that our method is extremely memory-efficient when used with checkpointing (5.7).

### 5.1 Training details and notation

We train our models using AMSGrad [42], [43] with the initial learning rate $4 \cdot 10^{-4}$. The learning rate is decreased after each epoch according to cosine learning rate policy. We divide the learning rate by 4 for all pre-trained weights. Batch size is an important hyper-parameter of the optimization procedure. If we train with batch size 1, then the batchnorm statistics fit exactly the image we are training on. This hinders learning due to large covariate shift across different images [44]. We combat this by training on random crops with batch size 16. Before cropping, we apply a random flip and rescale the image with a random factor between 0.5 and 2. The crop size is set to 448, 512 or 768 depending on the resolution of the dataset. If a crop happens to be larger than the rescaled image, then the undefined pixels are filled with the mean pixel. We train for 300 epochs unless otherwise stated. We employ multiple cross entropy losses along the upsampling path as shown in Figure 1. In order to promote efficient learning, auxiliary losses are formulated on the native k× subsampled resolution as cross-entropy between predictions $\hat{\mathbf{y}}$ and soft targets $\mathbf{y}^{\mathrm{k}}$:

$$\mathcal{L}_k^{\mathrm{aux}} = -\frac{k^2}{WH} \sum_{rc} \sum_{f} y_{rcf}^{\mathrm{k}} \log \hat{y}_{rcf} \ . \tag{1}$$

The soft targets correspond to the distribution of one-hot labels $\mathbf{y}^{\mathrm{OH}}$ in the corresponding k×k window:

$$\mathbf{y}_{rc}^{\mathrm{k}} = \frac{1}{k^2} \sum_{\Delta r \Delta c} \mathbf{y}_{kr+\Delta r, kc+\Delta c}^{\mathrm{OH}} \ . \tag{2}$$

We apply loss after each upsampling step, and to each of the four pooled tensors within the SPP module. The loss of the final predictions and the mean auxiliary loss are weighted with factors 0.6 and 0.4, respectively. These values were validated on the Cityscapes dataset. After the training, we recompute the batchnorm statistics as exact averages over the training set instead of decayed moving averages used during training. This practice slightly improves the model generalization.

We employ the following notation to describe our experiments throughout the section. LDN stands for Ladder DenseNet, the architecture proposed in Section 3. The symbol $d{\rightarrow}u$ denotes a model which downsamples the input resolution $d$ times, and then uses ladder-style upsampling to produce predictions subsampled $u$ times. For example, LDN121 64→4 denotes the model shown in Figure 1. Similarly, DDN and LDDN denote a dilated DenseNet, and a dilated DenseNet with ladder-style upsampling. The symbol $d{\downarrow}$ denotes a model which reduces the input resolution $d$ times and has no upsampling path. MS denotes multi-scale evaluation on 5 scales (0.5, 0.75, 1, 1.5 and 2), and respective horizontal flips. $\overline{\text{IoU}}$ and Cat. $\overline{\text{IoU}}$ denote the standard mean IoU metric over classes and categories. The instance-level mean IoU ($\overline{\text{iIoU}}$) metric [45] emphasizes contribution of pixels at small instances, and is therefore evaluated only on 8 object classes. The model size is expressed as the total number of parameters in millions (M). FLOP denotes the number of fused multiply-add operations required for inference on a single 1024×1024 (1MPx) image.

### 5.2 Cityscapes

Cityscapes has been recorded in 50 cities during daylight and fine weather [45]. It features 19 classes, many dynamic objects, and varying scene layout and background. Our experiments target the finely annotated dataset consisting of 2975 training, 500 validation, and 1525 test images of 1024×2048 pixels.

Table 1 validates several popular backbones coupled with the same SPP and upsampling modules. Due to hardware constraints, here we train and evaluate on half resolution images, and use 448×448 crops. The first section presents the DN121 32↓ baseline. The second section presents our models with ladder-style upsampling. The LDN121 64→4 model outperforms the baseline for 10 percentage points (pp) of $\overline{\text{IoU}}$ improvement. Most of this improvement occurs on very small objects (due to blending with high-resolution features) and very large objects (due to enlarged spatial context caused by increased subsampling, 64 vs 32). Note that LDN121 32→4 slightly outperforms LDN121 64→4 at this resolution due to better accuracy at semantic borders. However, the situation will be opposite in full resolution images due to larger objects (which require a larger receptive field) and off-by-one-pixel annotation errors. The LDN169 32→4 model features a stronger backbone, but obtains a slight deterioration (0.8pp) with respect to LDN121 32→4. We conclude that half resolution images do not contain enough training pixels to support the capacity of DenseNet-169. The LDN121 32→2 model shows that further upsampling doubles the computational complexity while bringing only a slight $\overline{\text{IoU}}$ improvement. The third section demonstrates that residual and DPN backbones achieve worse generalization than their DenseNet counterparts. The last section presents the models which avoid the resolution loss by dilated convolutions. The DDN-121 8↓ model removes the strided pooling layers before the DenseNet blocks DB3 and DB4, and introduces dilation in DB3 (rate=2) and DB4 (rate=4). The SPP output is now 8× downsampled. From there we produce logits and finally restore the input resolution with bilinear upsampling. The LDDN-121 8→4 model continues with one step of ladder-style upsampling to obtain 4× downsampled predictions as in previous LDN experiments. We observe a 3pp $\overline{\text{IoU}}$ improvement due to ladder-style upsampling. The LDDN-121 16→4 model dilates only the last dense block and performs two steps of ladder-style upsampling. We observe a marginal improvement which, however, still comes short of LDN121 32→4 from Table 1. Training the DDN-121 4↓ model was infeasible due to huge computational requirements when the last three blocks operate on 4× subsampled resolution. A comparison of computational complexity reveals that the dilated LDDN-121 8→4 model has almost 3× more FLOPs than LDN models with similar $\overline{\text{IoU}}$ performance. Finally, our memory consumption measurements show that LDDN-121 8→4 consumes around 2× more GPU memory than LDN121 32→4. We conclude that dilated models achieve a worse generalization than their LDN counterparts while requiring more computational power.

TABLE 1
Validation of backbone and upsampling architectures on Cityscapes val. Both training and evaluation images were resized to 1024×512.

| Method | Class $\overline{\text{IoU}}$ | Class $\overline{\text{iIoU}}$ | Cat. $\overline{\text{IoU}}$ | Model size | FLOP 1MPx |
|---|---|---|---|---|---|
| DN121 32↓ | 66.2 | 46.7 | 78.3 | 8.2M | 56.1G |
| LDN121 64→4 | 75.3 | 54.8 | 88.1 | 9.5M | 66.5G |
| LDN121 32→4 | **76.6** | 57.5 | 88.6 | 9.0M | 75.4G |
| LDN169 32→4 | 75.8 | 55.5 | 88.4 | 15.6M | 88.8G |
| LDN121 32→2 | **77.5** | **58.9** | 89.3 | 9.4M | 154.5G |
| ResNet18 32→4 | 70.9 | 49.7 | 86.7 | 13.3M | 55.7G |
| ResNet101 32→4 | 73.7 | 54.3 | 87.8 | 45.9M | 186.7G |
| ResNet50 32→4 | 73.9 | 54.2 | 87.8 | 26.9M | 109.0G |
| DPN68 32→4 | 74.0 | 53.0 | 87.8 | 13.7M | 59.0G |
| DDN-121 8↓ | 72.5 | 52.5 | 85.5 | 8.2M | 147.8G |
| LDDN-121 8→4 | 75.5 | 55.3 | 88.3 | 8.6M | 174.8G |
| LDDN-121 16→4 | 75.8 | 55.9 | 88.4 | 8.9M | 87.0G |

Table 2 shows experiments on full Cityscapes val images where we train on 768×768 crops. We obtain the most interesting results with the LDN121 64→4 model presented in Figure 1: 79% $\overline{\text{IoU}}$ with a single forward pass and 80.3% with multi-scale (MS) inference. Models with stronger backbones (DenseNet-169, DenseNet-161) validate only slightly better. We explain that by insufficient training data and we expect that successful models need less capacity for Cityscapes than for a harder task of discriminating ImageNet classes.

TABLE 2
Validation of various design options on full-resolution Cityscapes val.

| Method | Class $\overline{\text{IoU}}$ | Cat. $\overline{\text{IoU}}$ | Model size | FLOP 1MPx |
|---|---|---|---|---|
| LDN121 32→4 | 78.4 | 90.0 | 9.0M | 75.4G |
| LDN121 64→4 | **79.0** | 90.3 | 9.5M | 66.5G |
| LDN121 128→4 | 78.4 | 90.1 | 9.9M | 66.2G |
| LDN161 64→4 | 79.1 | 90.2 | 30.0M | 138.7G |
| LDN121 64→4 MS | **80.3** | 90.6 | 9.5M | 536.2G |

Table 3 compares our models with the state-of-the-art on validation and test sets. Our models generalize better than or equal to all previous approaches, while being much more efficient. In particular, we are the first to reach 80% $\overline{\text{IoU}}$ on Cityscapes test with only 66.5 GFLOP per Mpx. Table 4 presents a comparison with efficient approaches. It reports normalized processing time according to GPU conversion factors proposed in [29]. Label dws denotes depthwise separable convolutions in the upsampling path (cf. Table 9). Figure 4 plots the best performing models from Tables 3 and 4 in ($\overline{\text{IoU}}$,TFLOP) coordinates. The figure clearly shows that our models achieve the best trade-off between accuracy and computational complexity.

### 5.3 CamVid

The CamVid dataset contains images of urban road driving scenes. We use the 11-class split from [31], which consists of 367 training, 101 validation, and 233 testing images. The resolution of all images is 720×960. Following the common practice, we incorporate the val subset into train because it is too small and too easy to be useful for validation. We train all models from random initialization (RI), and by fine-tuning the parameters pre-trained on ImageNet (PT). We

TABLE 3
Comparison with the state-of-the-art when training only on Cityscapes fine. FLOP metrics marked with '†' include only the backbone.

| Method | Backbone | IoU Val | IoU Test | Tflop@1Mpx single scale |
|---|---|---|---|---|
| LKM [25] | rn50 d32↓ | 77.4 | 76.9 | 0.110† |
| TuSimple [46] | rn101 d8↓ | 76.4 | 77.6 | 0.720† |
| SAC-multiple [47] | rn101 d8↓ | 78.7 | 78.1 | 0.720† |
| ResNet-38 [48] | wrn38 d8↓ | 77.9 | 78.4 | 2.110† |
| PSPNet [17] | rn101 d8↓ | n/a | 78.4 | 0.720† |
| Multi Task [49] | rn101 d8↓ | n/a | 78.5 | 0.720 |
| TKCN [50] | rn101 d8↓ | n/a | 79.5 | 0.720† |
| DFN [51] | rn101 d32↓ | n/a | 79.3 | 0.450† |
| Mapillary [20] | wrn38 d8↓ | 78.3 | n/a | 2.110† |
| DeepLab v3 [19] | rn101 d8↓ | 79.3 | n/a | 0.720† |
| DeepLabv3+ [33] | x-65 d8↓ | 79.1 | n/a | 0.710 |
| DRN [52] | wrn38 d8↓ | 79.7 | 79.9 | 2.110† |
| DenseASPP [21] | dn161 d8↓ | 78.9 | **80.6** | 0.500† |
| LDN121 64→4 | dn121 64↓ | **80.3** | 80.0 | **0.066** |
| LDN161 64→4 | dn161 64↓ | **80.7** | **80.6** | 0.139 |

TABLE 4
Comparison with the state-of-the-art in efficient inference after training on Cityscapes fine. We report normalized [29] time of FP32 single-scale inference on Titan Xp GPU for 1024×2048 images.

| Method | Backbone | IoU Val | IoU Test | Tflop 1Mpx | ms Titan Xp |
|---|---|---|---|---|---|
| ERFNet [53] | erfnet d8↓ | 71.5 | 69.7 | **0.055** | 89.0 |
| SwiftNet [29] | rn18 d32↓ | 75.4 | 75.5 | **0.052** | **22.7** |
| LinkNet [54] | rn18 d32↓ | 76.4 | n/a | 0.201 | 108.8 |
| BiSeNet [55] | rn18 d8↓ | 74.8 | 74.7 | **0.049** | 29.4 |
| LRN18 32→4 | rn18 32↓ | 76.1 | n/a | **0.056** | **23.5** |
| LRN50 32→4 | rn50 32↓ | 77.5 | n/a | 0.109 | 52.0 |
| LDN121 dws | dn121 64↓ | 78.9 | n/a | **0.054** | 42.6 |
| LDN121 64→4 | dn121 64↓ | **79.0** | **79.3** | 0.066 | 40.7 |

train on 512×512 crops for 400 epochs with pre-training, and 800 epochs with random init. All other hyperparameters are the same as in Cityscapes experiments. Table 5 shows our results on full-resolution CamVid test. The conclusions are similar as on half-resolution Cityscapes val (cf. Table 1), which does not surprise us due to similar input resolutions. LDN121 32→4 wins both in the pre-trained and in the random init case, with LDN121 64→4 being the runner-up. Table 6 compares our best results with the related work on

TABLE 5
Single-scale inference on full-resolution CamVid test with ImageNet pre-training (PT) and random initialization (RI).

| Method | PT IoU | RI IoU | Model size | FLOP 1MPx |
|---|---|---|---|---|
| LDN121 32→4 | **77.3** | **70.9** | 9.0M | 75.4G |
| LDN121 64→4 | 76.9 | 68.7 | 9.5M | 66.5G |
| ResNet18 32→4 | 73.2 | 70.0 | 13.3M | 55.7G |
| ResNet50 32→4 | 76.1 | 69.9 | 26.9M | 109.0G |
| ResNet101 32→4 | 76.7 | 69.4 | 45.9M | 186.7G |

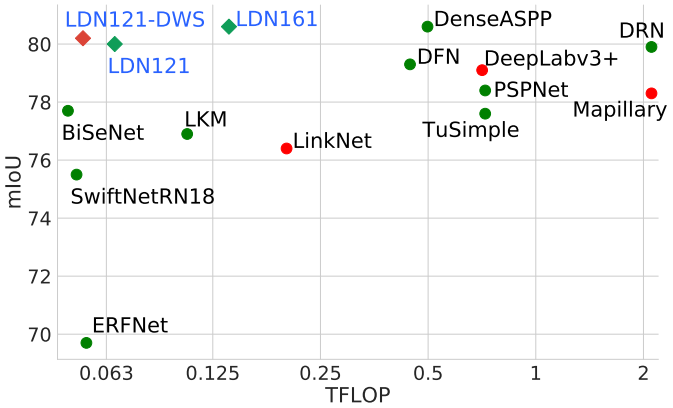


Fig. 4. Accuracy vs forward pass complexity on Cityscapes test (green) and val (red) for approaches from Table 3. LDN121 is the first method to achieve 80% IoU while being applicable in real-time.

CamVid test where, to the best of our knowledge, we obtain state-of-the-art results.

TABLE 6
Comparison of our models with the state-of-the-art on CamVid test. We use multi-scale inference in experiments on full resolution.

| Method | Backbone | ImgNet | Resolution | IoU |
|---|---|---|---|---|
| Tiramisu [22] | DenseNet | | half | 66.9 |
| FC-DRN [56] | DenseResNet | | half | 69.4 |
| G-FRNet [57] | VGG-16 | ✓ | half | 68.8 |
| BiSeNet [55] | Xception39 | ✓ | full | 65.6 |
| ICNet [58] | ResNet-50 | ✓ | full | 67.1 |
| BiSeNet [55] | ResNet-18 | ✓ | full | 68.7 |
| LDN121 16→2 | DenseNet | | half | **69.5** |
| LDN121 32→4 | DenseNet | | full | **71.9** |
| LDN121 16→2 | DenseNet | ✓ | half | **75.8** |
| LDN121 32→4 | DenseNet | ✓ | full | **78.1** |

### 5.4 Cross-dataset generalization

We explore the capability of our models to generalize across related datasets. Mapillary Vistas [59] is a large road driving dataset featuring five continents and diverse lighting, seasonal and weather conditions. It contains 18000 training, 2000 validation, and 5000 test images. In our experiments, we remap annotations to 19 Cityscapes classes, and resize all images to width 2048. The KITTI dataset contains road driving images recorded in Karlsruhe [60]. It features the Cityscapes labeling convention and depth reconstruction groundtruth. There are 200 training and 200 test images. All images are 370×1226.

Table 7 shows that training only on Cityscapes results in poor generalization due to urban bias and constant acquisition setup. On the other hand, models trained on Vistas generalize much better due to better diversity of the training dataset. Training on both datasets achieves the best results.

We briefly describe our submission [61] to the Robust vision challenge held at CVPR 2018. The semantic segmentation challenge featured 1 indoor (ScanNet), and 3 road-driving (Cityscapes, KITTI, WildDash) datasets. A qualifying model had to be trained and evaluated on all four

TABLE 7
Cross-dataset evaluation on half-resolution images. We train separate models on Cityscapes, Vistas and their union, and show results on validation sets of three driving datasets. These experiments present an older variant of LDN121 64→4 which splits DB4 instead of DB3.

| Method | Training dataset | Cityscapes IoU | Vistas IoU | KITTI IoU |
|---|---|---|---|---|
| LDN121 64→4 s4 | Cityscapes | 76.0 | 44.0 | 59.5 |
| LDN121 64→4 s4 | Vistas | 68.7 | 73.0 | 64.7 |
| LDN121 64→4 s4 | Cit. + Vis. | **76.2** | **73.9** | **68.7** |

benchmarks, and it had to predict at least 39 classes: 19 from Cityscapes and 20 from ScanNet. We have trained a LDN169 64→4 model on Cityscapes train+val, KITTI train, WildDash val and ScanNet train+val. Our submission `ROB_LDN` had received the runnner-up prize. Unlike the winners [20], our model was not trained on any additional datasets like Vistas.

### 5.5 Pascal VOC 2012

PASCAL VOC 2012 [62] contains photographs from private collections. There are 6 indoor classes, 7 vehicles, 7 living beings, and one background class. The dataset contains 1464 train, 1449 validation and 1456 test images of variable size. Following related work, we also train on 10582 images from the AUG set [63]. Due to annotation errors in the AUG labels, we first train for 100 epochs on AUG, and then fine-tune for another 100 epochs on train (or train+val). We use 512×512 crops and divide the learning rate of pretrained weights by 8. All other hyperparameters are the same as in Cityscapes experiments. Table 8 shows that our models set the new state-of-the-art among models which do not pre-train on COCO.

TABLE 8
Experimental evaluation on Pascal VOC 2012 validation and test.

| Method | AUG | MS | Val IoU | Test IoU |
|---|---|---|---|---|
| DeepLabv3+ Res101 | ✓ | ✓ | 80.6 | n/a |
| DeepLabv3+ Xcept | ✓ | ✓ | 81.6 | n/a |
| DDSC [64] | ✓ | | n/a | 81.2 |
| AAF [65] | ✓ | ✓ | n/a | 82.2 |
| PSPNet [17] | ✓ | ✓ | n/a | 82.6 |
| DFN [51] | ✓ | ✓ | 80.6 | 82.7 |
| EncNet [66] | ✓ | ✓ | n/a | 82.9 |
| LDN121 32→4 | | | 76.4 | n/a |
| LDN169 32→4 | ✓ | ✓ | 80.5 | 81.6 |
| LDN161 32→4 | | | 78.6 | n/a |
| LDN161 32→4 | ✓ | | 80.4 | n/a |
| LDN161 32→4 | ✓ | ✓ | **81.9** | **83.6** |

### 5.6 Ablation experiments on Cityscapes

We interpret the operation of our models by disabling an increasingly large portion of the backbone tail, and evaluating the accuracy of the pruned model. We disable convolutional units by setting their output to zero. In order to allow pruning in all processing blocks, we preserve all transition-down units and 1×1 convolutions in residual connections. The left graph in Figure 5 compares LDN121 and LRN50 models after pruning an equal portion of convolutional units from the last two processing blocks of the backbone. We observe that DenseNet converges faster than ResNet, although ladder-style upsampling encourages both backbones to recognize small classes (eg. traffic sign) in early layers. The right graph shows per-class performance of pruned LDN121 models. We observe that small and less complex classes (bicycle, pole, traffic sign and road) are recognized by early layers, while hard and large classes (truck, bus and train) are recognized in deeper layers. This can be viewed as a kind of self-regularization: although the back-end of the backbone can have a huge capacity, this capacity is not used for detecting easy classes.

Table 9 evaluates the impact of auxiliary loss, SPP, and depthwise separable convolutions on generalization accuracy. The experiment labeled NoSPP replaces the SPP module with a single 3×3 convolution. The resulting 1.5pp performance drop suggests that SPP brings improvement even with 64 times subsampled features. The subsequent experiment shows that the SPP module proposed in [17] does not work well with our training on Cityscapes. We believe that the 1.4pp performance drop is due to inadequate pooling grids and larger feature dimensionality which encourages overfitting. The NoAux model applies the loss only to final predictions. The resulting 1.2pp performance hit suggests that auxiliary loss reduces overfitting to low-level features within the upsampling path. The DWS model reduces the computational complexity by replacing all 3×3 convolutions in the upsampling path with depthwise separable convolutions. This improves upsampling efficiency while only marginally decreasing accuracy.

TABLE 9
Impact of auxiliary loss, SPP, and depthwise separable convolutions on generalization accuracy on full-resolution Cityscapes val.

| Method | IoU | Model size | FLOP 1MPx |
|---|---|---|---|
| LDN121 64→4 NoSPP | 77.5 | 10.2M | 66.7G |
| LDN121 64→4 SPP [17] | 77.6 | 10.6M | 66.9G |
| LDN121 64→4 NoAux | 77.8 | 9.5M | 66.5G |
| LDN121 64→4 DWS | 78.6 | 8.7M | 54.2G |
| LDN121 64→4 | **79.0** | 9.5M | 66.5G |

Table 10 shows ablation experiments which evaluate the

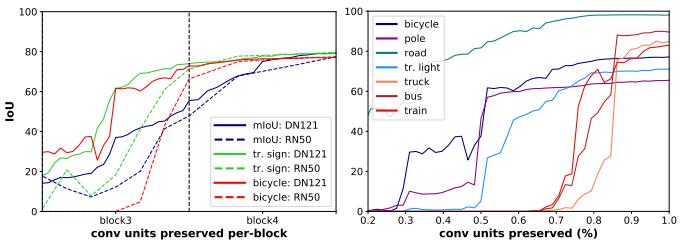


Fig. 5. Influence of pruning convolutional units from the backbone tail to the Cityscapes val accuracy (there was no re-training). DenseNet is more resilient than ResNet (left). DenseNet recognizes small object classes in early layers, which saves later capacity for large classes (right).

impact of data augmentations on generalization. We observe that random image flip, crop, and scale jitter improve $\overline{\text{IoU}}$ by almost 5pp, and conclude that data augmentation is of great importance for semantic segmentation.

TABLE 10
Impact of data augmentation to the segmentation accuracy ($\overline{\text{IoU}}$) on Cityscapes val while training LDN121 64→4 on full images.

| augmentation: | none | flip | flip/crop | flip/crop/scale |
|---|---|---|---|---|
| accuracy ($\overline{\text{IoU}}$): | 74.0 | 75.7 | 76.7 | 79.0 |

### 5.7 Gradient checkpointing

Table 11 explores effects of checkpointing to the memory footprint and the execution speed while training the default LDN model from Figure 1. The columns show i) the maximum memory allocation while training with batch size 6, ii) the maximum batch size we could fit into GPU memory, and iii) the corresponding training speed in frames per second (FPS). We start from the straightforward baseline and gradually introduce more and more extensive checkpointing. The checkpointing approaches are designated as follows. The label *cat* refers to the concatenation at the input of a DenseNet unit. The labels 1×1 and 3×3 refer to the first and the second BN-ReLu-conv group within a DenseNet unit. The label *stem* denotes the 7×7 convolution at the very beginning of DenseNet [14], including the following batchnorm, ReLU and max-pool operations. Labels TD and TU correspond to the transition-down and the transition-up blocks. The label *block* refers to the entire processing block (this approach is applicable to most backbones). Parentheses indicate the checkpoint segment. For example, (cat 1×1 3×3) caches only the concatenation inputs, while the first batchnorm, the 1×1 convolution and the second batchnorm are re-computed during backprop. On the other hand, (cat 1×1) (3×3) means that each convolution is in a separate segment. Here we cache the concatenation inputs and the input to the second batchnorm, while the two batchnorms are recomputed. Consequently, training with (cat 1×1 3×3) accommodates larger batches.

Now we present the most important results. Checkpointing the (cat 1×1) subgraph brings the greatest savings with respect to the baseline (4.5 GB), since it has most feature maps on input. Nevertheless, checkpointing the whole DenseNet unit (cat 1×1 3×3) frees further 3 GB. Finally, checkpointing stem, transition-down and transition-up blocks relieves additional 1.8 GB. Altogether, this results in a more than five-fold reduction of memory requirements, from 11.3 GB to 2.1 B.

Experiments with the label (block) treat each dense block as a checkpoint segment. This requires more memory than (cat 1×1 3×3) because additional memory needs to be allocated during re-computation. The approach (cat 1×1) (3×3) is similar to the related previous work [14]. These experiments show that the smallest memory footprint is achieved by checkpointing the stem, transition-down and transition-up blocks, as well as each DenseNet unit as a whole.

Table 12 shows that our checkpointing approach allows training the LDN161 model with a six-fold increase of batch size with respect to the baseline. On the other hand, the only previous checkpointing technique for semantic segmentation [20] yields only a two-fold increase of batch size.

TABLE 11
Impact of checkpointing to memory footprint and training speed. We train LDN 32→4 on 768×768 images on a Titan Xp with 12 GB RAM.

| Checkpointing variant | Memory bs=6 (MB) | Max BS | Train FPS |
|---|---|---|---|
| baseline - no ckpt | 11265 | 6 | 11.3 |
| (3x3) | 10107 | 6 | 10.5 |
| (cat 1x1) | 6620 | 10 | 10.4 |
| (cat 1x1) (3x3) | 5552 | 12 | 9.7 |
| (block) (stem) (TD) (UP) | 3902 | 16 | 8.4 |
| (cat 1x1 3x3) | 3620 | 19 | 10.1 |
| (cat 1x1 3x3) (stem) (TD) (UP) | 2106 | 27 | 9.2 |

TABLE 12
Comparison of memory footprint and training speed across various models. We process 768×768 images on a Titan Xp with 12 GB RAM.

| Model | Uses Ckpt | Memory bs=6 (MB) | Max BS | Train FPS |
|---|---|---|---|---|
| LDN161 32→4 | | 20032 | 3 | 5.6 |
| ResNet101 32→4 | | 15002 | 4 | 7.8 |
| LDN121 32→4 | | 11265 | 6 | 11.3 |
| ResNet50 32→4 | | 10070 | 6 | 11.6 |
| ResNet18 32→4 | | 3949 | 17 | 24.4 |
| LDN161 32→4 | ✓ | 3241 | 19 | 4.4 |
| LDN121 32→4 | ✓ | 2106 | 27 | 9.2 |

### 5.8 Qualitative Cityscapes results

Finally, we present some qualitative results on Cityscapes. Figure 6 shows predictions from different stages of the upsampling path. Predictions from early layers miss most small objects, however ladder-style upsampling succeeds to recover them after blending with high-resolution features.

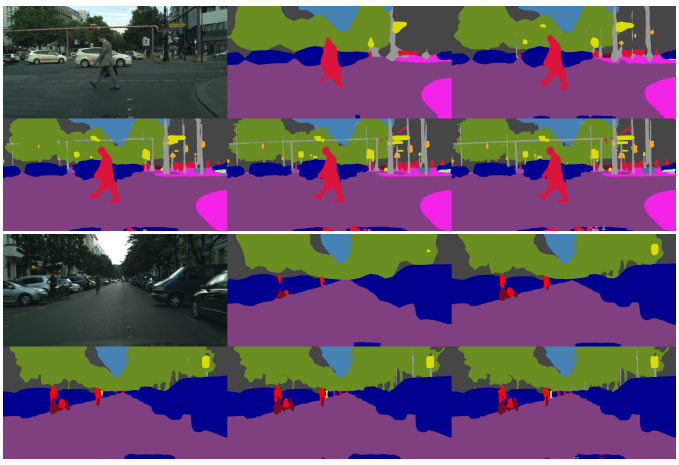

Fig. 6. Impact of ladder-style upsampling to the accuracy of semantic borders and small objects. We shows predictions recovered from 64×, 32×, 16×, 8×, and 4× subsampled features along the upsampling path.

Figure 7 shows images from Cityscapes test in which our best model makes mistakes. It is our impression that most of these errors are due to insufficient context, despite our efforts to enlarge the receptive field. Overall, we achieve the worst IoU on fences (60%) and walls (61%).

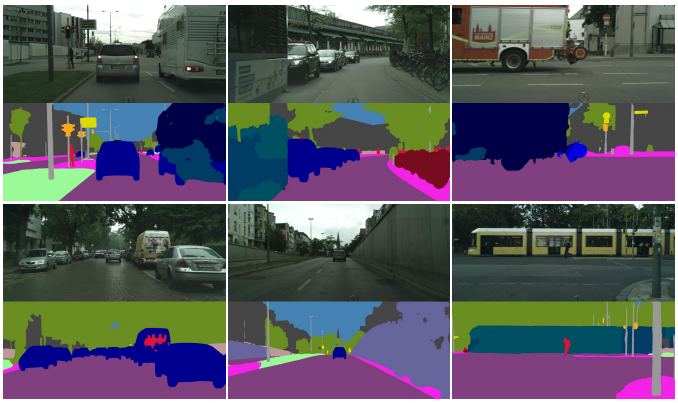

Fig. 7. Images from Cityscapes test where our model misclassified some parts of the scene. These predictions are produced by the LDN161 64→4 model which achieves 80.6% IoU on the test set.

Figure 8 shows some images from Cityscapes test where our best model performs well in spite of occlusions and large objects. We note that small objects are very well recognized which confirms the merit of ladder-style upsampling. A video demonstration of a single-scale LDN-121 64→4 model is available here: https://youtu.be/QrB7Np_8GXY.

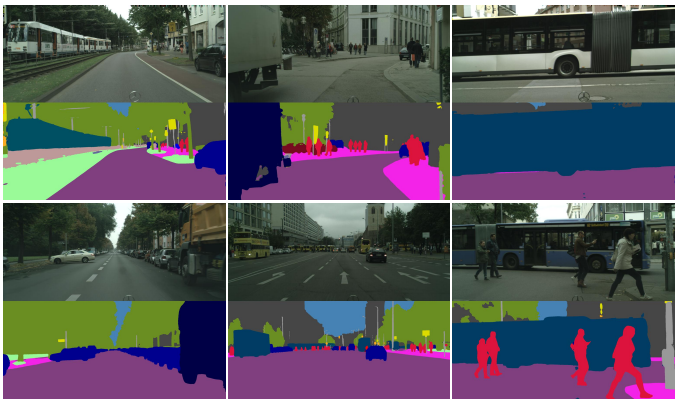

Fig. 8. Images from Cityscapes test where our best model (LDN161 64→4, 80.6% IoU test) makes no significant errors in spite of large objects, occlusions, small objects and difficult classes.

## 6 Conclusion

We have presented a novel semantic segmentation approach based on DenseNet architecture and ladder-style upsampling. Different than concurrent encoder-decoder approaches, we argue for an asymmetric architecture with thick encoder and thin decoder, which assigns much more capacity to recognition than to localization. In comparison with widely used dilated approaches, our design substantially decreases computational complexity (both space and time) while generalizing better.

The proposed design exhibits outstanding spatial efficiency which however has to be unlocked by recomputing all concats, batchnorms and projections during backprop. This decreases memory footprint for up to 6× while only slightly increasing training time. Thus, LDN121 32→4 can be trained on 768×768 crops with batch size 16 in only 5.3 GB RAM.

We have performed extensive experiments on Cityscapes, CamVid, ROB 2018 and Pascal VOC 2012. We achieve state-of-the-art on Cityscapes test without the coarse training subset (LDN161: 80.6% IoU) as well as on Pascal VOC 2012 test without COCO pretraining (LDN161: 83.6% IoU). None of the competing approaches is able to train on a single GPU.

To the best of our knowledge, this is the first DenseNet-based approach for efficient dense prediction on 2 MPx images. A TensorRT implementation of our single-scale LDN121 model processes 1024×2048 images at 25 Hz on a single Titan Xp, while achieving 79.3% IoU on Cityscapes test. Suitable directions for future work include further improvements of run-time speed, as well as exploiting the reclaimed memory for dense prediction and forecasting in video.

## Acknowledgments

This work has been supported by the Croatian Science Foundation under the project I-2433-2014, and the European Regional Development Fund under the grant KK.01.1.1.01.0009.

## References

[1] R. Hadsell, P. Sermanet, J. Ben, A. Erkan, M. Scoffier, K. Kavukcuoglu, U. Muller, and Y. LeCun, "Learning long-range vision for autonomous off-road driving," *J. Field Robotics*, vol. 26, no. 2, pp. 120–144, 2009.

[2] A. Petrovai and S. Nedevschi, "Efficient instance and semantic segmentation for automated driving," in *IV*, 2019.

[3] Y. Aksoy, T. Oh, S. Paris, M. Pollefeys, and W. Matusik, "Semantic soft segmentation," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 72:1–72:13, 2018.

[4] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015, pp. 234–241.

[5] J. Shotton, J. M. Winn, C. Rother, and A. Criminisi, "Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context," *International Journal of Computer Vision*, vol. 81, no. 1, pp. 2–23, 2009.

[6] G. Csurka and F. Perronnin, "An efficient approach to semantic segmentation," *International Journal of Computer Vision*, vol. 95, no. 2, pp. 198–212, 2011.

[7] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1915–1929, 2013.

[8] I. Kreso, D. Causevic, J. Krapac, and S. Segvic, "Convolutional scale invariance for semantic segmentation," in *GCPR*, 2016, pp. 64–75.

[9] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected crfs with gaussian edge potentials," in *NIPS*, 2011, pp. 109–117.

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

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc˙ IEEE*, 1998.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[13] ——, "Identity mappings in deep residual networks," in *ECCV*, 2016, pp. 630–645.
[14] G. Huang, Z. Liu, G. Pleiss, L. V. D. Maaten, and K. Weinberger, "Convolutional networks with dense connectivity," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019.
[15] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng, "Dual path networks," in *NIPS*, 2017, pp. 4470–4478.
[16] T. Chen, B. Xu, C. Zhang, and C. Guestrin, "Training deep nets with sublinear memory cost," *CoRR*, vol. abs/1604.06174, 2016.
[17] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *ICCV*, 2017.
[18] 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," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, 2018.
[19] L. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *CoRR*, vol. abs/1706.05587, 2017.
[20] S. Rota Bulò, L. Porzi, and P. Kontschieder, "In-place activated batchnorm for memory-optimized training of DNNs," in *CVPR*, June 2018.
[21] M. Yang, K. Yu, C. Zhang, Z. Li, and K. Yang, "DenseASPP for semantic segmentation in street scenes," in *CVPR*, 2018, pp. 3684–3692.
[22] S. Jégou, M. Drozdzal, D. Vázquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," *CoRR*, vol. abs/1611.09326, 2016.
[23] G. Ghiasi and C. C. Fowlkes, "Laplacian pyramid reconstruction and refinement for semantic segmentation," in *ECCV*, 2016, pp. 519–534.
[24] G. Lin, A. Milan, C. Shen, and I. D. Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation," in *CVPR*, 2017.
[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.
[26] H. Valpola, "From neural PCA to deep unsupervised learning," *CoRR*, vol. abs/1411.7783, 2014.
[27] I. Kreso, J. Krapac, and S. Segvic, "Ladder-style densenets for semantic segmentation of large natural images," in *ICCV CVRSUAD*, 2017, pp. 238–245.
[28] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2014, pp. 1–16.
[29] M. Orsic, I. Kreso, P. Bevandic, and S. Segvic, "In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images," in *CVPR*, 2019.
[30] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 640–651, 2017.
[31] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, 2017.
[32] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in *ICLR*, 2014, pp. 1–16.
[33] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *ECCV*, 2018, pp. 833–851.
[34] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," in *CVPR*, 2017, pp. 936–944.
[35] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015, pp. 3431–3440.
[36] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe, "Full-resolution residual networks for semantic segmentation in street scenes," in *CVPR*, July 2017.
[37] M. A. Islam, M. Rochan, B. Neil D. B. and Y. Wang, "Gated feedback refinement network for dense image labeling," in *CVPR*, July 2017.
[38] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," in *ICCV*, 2017, pp. 2980–2988.
[39] H. Zhao, Y. Zhang, S. Liu, J. Shi, C. C. Loy, D. Lin, and J. Jia, "Psanet: Point-wise spatial attention network for scene parsing," in *ECCV*, 2018, pp. 270–286.
[40] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *CVPR*, 2006, pp. 2169–2178.
[41] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, 2015.
[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.
[43] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," in *ICLR*, 2018.
[44] 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, pp. 1–9.
[45] 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.
[46] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. Cottrell, "Understanding Convolution for Semantic Segmentation," *CoRR*, vol. abs/1702.08502, 2017.
[47] R. Zhang, S. Tang, Y. Zhang, J. Li, and S. Yan, "Scale-adaptive convolutions for scene parsing," in *ICCV*, Oct 2017.
[48] Z. Wu, C. Shen, and A. van den Hengel, "Wider or deeper: Revisiting the resnet model for visual recognition," *CoRR*, vol. abs/1611.10080, 2016.
[49] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *CVPR*, 2018, pp. 7482–7491.
[50] T. Wu, S. Tang, R. Zhang, J. Cao, and J. Li, "Tree-structured kronecker convolutional network for semantic segmentation," *CoRR*, vol. abs/1812.04945, 2018.
[51] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Learning a discriminative feature network for semantic segmentation," in *CVPR*, 2018, pp. 1857–1866.
[52] Y. Zhuang, F. Yang, L. Tao, C. Ma, Z. Zhang, Y. Li, H. Jia, X. Xie, and W. Gao, "Dense relation network: Learning consistent and context-aware representation for semantic image segmentation," in *ICIP*, 2018, pp. 3698–3702.
[53] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo, "Erfnet: Efficient residual factorized convnet for real-time semantic segmentation," *IEEE Trans. Intelligent Transportation Systems*, vol. 19, no. 1, pp. 263–272, 2018.
[54] A. Chaurasia and E. Culurciello, "Linknet: Exploiting encoder representations for efficient semantic segmentation," in *VCIP*. IEEE, 2017, pp. 1–4.
[55] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Bisenet: Bilateral segmentation network for real-time semantic segmentation," in *ECCV*, 2018, pp. 334–349.
[56] A. Casanova, G. Cucurull, M. Drozdzal, A. Romero, and Y. Bengio, "On the iterative refinement of densely connected representation levels for semantic segmentation," *CoRR*, vol. abs/1804.11332, 2018.
[57] M. A. Islam, M. Rochan, S. Naha, N. D. B. Bruce, and Y. Wang, "Gated feedback refinement network for coarse-to-fine dense semantic image labeling," *CoRR*, vol. abs/1806.11266, 2018.
[58] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "Icnet for real-time semantic segmentation on high-resolution images," in *ECCV*, vol. 11207, 2018, pp. 418–434.
[59] G. Neuhold, T. Ollmann, S. Rota Bulò, and P. Kontschieder, "The mapillary vistas dataset for semantic unde rstanding of street scenes," in *ICCV*, 2017.
[60] H. Alhaija, S. Mustikovela, L. Mescheder, A. Geiger, and C. Rother, "Augmented reality meets computer vision: Efficient data generation for urban driving scenes," *International Journal of Computer Vision (IJCV)*, 2018.
[61] I. Kreso, M. Orsic, P. Bevandic, and S. Segvic, "Robust semantic segmentation with ladder-densenet models," *CoRR*, vol. abs/1806.03465, 2018.
[62] 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*, vol. 111, no. 1, pp. 98–136, 2015.
[63] B. Hariharan, P. Arbelaez, L. D. Bourdev, S. Maji, and J. Malik, "Semantic contours from inverse detectors," in *ICCV*, 2011, pp. 991–998.
[64] P. Bilinski and V. Prisacariu, "Dense decoder shortcut connections for single-pass semantic segmentation," in *CVPR*, 2018, pp. 6596–6605.

[65] T. Ke, J. Hwang, Z. Liu, and S. X. Yu, "Adaptive affinity fields for semantic segmentation," in *ECCV*, 2018, pp. 605–621.
[66] H. Zhang, K. J. Dana, J. Shi, Z. Zhang, X. Wang, A. Tyagi, and A. Agrawal, "Context encoding for semantic segmentation," in *CVPR*, 2018, pp. 7151–7160.

**Ivan Krešo** received his MSc degree in computer science from the University of Zagreb. He spent six years at the Faculty of Electrical Engineering and Computing, University of Zagreb, as a PhD student and teaching assistant. Currently he is a computer vision scientist at RealNetworks. His research interests include object classification and dense prediction for semantic segmentation and detection, as well as self-supervised and semi-supervised learning.

**Josip Krapac** received a Ph.D degree from Université de Caen Basse-Normandie, France. He spent two years as a post-doctoral researcher at INRIA Rennes, France, and three and a half years as a post-doctoral researcher at University of Zagreb. Currently he is a senior computer vision scientist at Zalando, Berlin. His research interests include image representations for object classification, detection and segmentation, as well as learning with minimal supervision.

**Siniša Šegvić** received a Ph.D. degree in computer science from the University of Zagreb, Croatia. He spent a year as a post-doctoral researcher at IRISA Rennes, and another year as a post-doctoral researcher at TU Graz. He is currently a full professor at University of Zagreb Faculty of Electrical Engineering and Computing. His research addresses efficient convolutional architectures for classification, dense prediction, and dense semantic forecasting.