Full-Resolution Residual Networks for Semantic Segmentation in Street Scenes

Tobias Pohlen Alexander Hermans Markus Mathias Bastian Leibe
Visual Computing Institute
RWTH Aachen University, Germany
tobias.pohlen@rwth-aachen.de {hermans, mathias, leibe}@vision.rwth-aachen.de

Abstract

*Semantic image segmentation is an essential component of modern autonomous driving systems, as an accurate understanding of the surrounding scene is crucial to navigation and action planning. Current state-of-the-art approaches in semantic image segmentation rely on pre-trained networks that were initially developed for classifying images as a whole. While these networks exhibit outstanding recognition performance (*i.e.*, what is visible?), they lack localization accuracy (*i.e.*, where precisely is something located?). Therefore, additional processing steps have to be performed in order to obtain pixel-accurate segmentation masks at the full image resolution. To alleviate this problem we propose a novel ResNet-like architecture that exhibits strong localization and recognition performance. We combine multi-scale context with pixel-level accuracy by using two processing streams within our network: One stream carries information at the full image resolution, enabling precise adherence to segment boundaries. The other stream undergoes a sequence of pooling operations to obtain robust features for recognition. The two streams are coupled at the full image resolution using residuals. Without additional processing steps and without pre-training, our approach achieves an intersection-over-union score of 71.8% on the Cityscapes dataset.*

1. Introduction

Recent years have seen an increasing interest in self driving cars and in driver assistance systems. A crucial aspect of autonomous driving is to acquire a comprehensive understanding of the surroundings in which a car is moving. Semantic image segmentation [49, 38, 21, 53, 33], the task of assigning a set of predefined class labels to image pixels, is an important tool for modeling the complex relationships of the semantic entities usually found in street scenes, such as cars, pedestrians, road, or sidewalks. In automotive scenarios it is used in various ways, *e.g.* as a pre-processing step to discard image regions that are unlikely to contain objects of interest [42, 15], to improve object detection [4, 23, 24, 58], or in combination with 3D scene geometry [32, 17, 35]. Many of those applications require precise region boundaries [20]. In this work, we therefore pursue the goal of achieving high-quality semantic segmentation with precise boundary adherence.

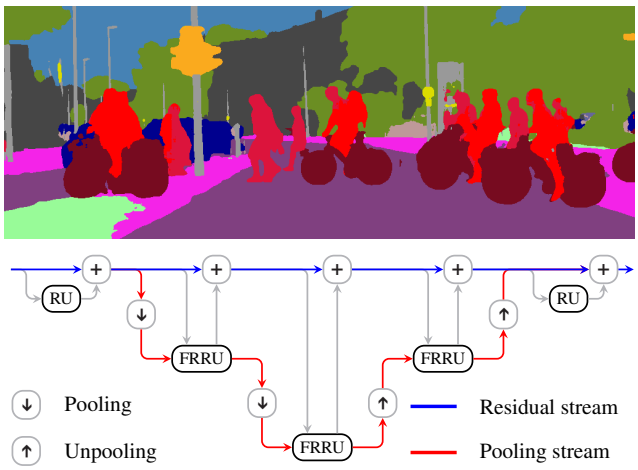

Figure 1. Example output and the abstract structure of our full-resolution residual network. The network has two processing streams. The residual stream (blue) stays at the full image resolution, the pooling stream (red) undergoes a sequence of pooling and unpooling operations. The two processing streams are coupled using full-resolution residual units (FRRUs).

Current state-of-the-art approaches for image segmentation all employ some form of *fully convolutional network (FCN)* [38] that takes the image as input and outputs a probability map for each class. Many papers rely on network architectures that have already been proven successful for image classification such as variants of the ResNet [25] or the VGG architecture [50]. Starting from pre-trained nets, where a large number of weights for the target task can be pre-set by an auxiliary classification task, reduces training time and often yields superior performance compared to training a network from scratch using the (possibly limited amount of) data of the target application. However, a main limitation of using such pre-trained networks is that they

severely restrict the design space of novel approaches, since new network elements such as batch normalization [27] or new activation functions often cannot be added into an existing architecture.

When performing semantic segmentation using FCNs, a common strategy is to successively reduce the spatial size of the feature maps using pooling operations or strided convolutions. This is done for two reasons: First, it significantly increases the size of the receptive field and second, it makes the network robust against small translations in the image. While pooling operations are highly desirable for recognizing objects in images, they significantly deteriorate localization performance of the networks when applied to semantic image segmentation. Several approaches exist to overcome this problem and obtain pixel-accurate segmentations. Noh *et al*. [41] learn a mirrored VGG network as a decoder, Yu and Koltun [55] introduce dilated convolutions to reduce the pooling factor of their pre-trained network. Ghiasi *et al*. [20] use multi-scale predictions to successively improve their boundary adherence. An alternative approach used by several methods is to apply post-processing steps such as CRF-smoothing [30].

In this paper, we propose a novel network architecture that achieves state-of-the-art segmentation performance without the need for additional post-processing steps and without the limitations imposed by pre-trained architectures. Our proposed ResNet-like architecture unites strong recognition performance with precise localization capabilities by combining two distinct processing streams. One stream undergoes a sequence of pooling operations and is responsible for understanding large-scale relationships of image elements; the other stream carries feature maps at the full image resolution, resulting in precise boundary adherence. This idea is visualized in Figure 1, where the two processing streams are shown in blue and red. The blue residual stream reflects the high-resolution stream. It can be combined with classical residual units (left and right), as well as with our new full-resolution residual units (FRRU). The FRRUs from the red pooling lane act as residual units for the blue stream, but also undergo pooling operations and carry high-level information through the network. This results in a network that successively combines and computes features at two resolutions.

This paper makes the following contributions: *(i)* We propose a novel network architecture geared towards precise semantic segmentation in street scenes which is not limited to pre-trained architectures and achieves state-of-the-art results. *(ii)* We propose to use two processing streams to realize strong recognition and strong localization performance: One stream undergoes a sequence of pooling operations while the other stream stays at the full image resolution. *(iii)* In order to foster further research in this area, we published our code and the trained models on GitHub.

2. Related Work

The dramatic performance improvements from using CNNs for semantic segmentation have brought about an increasing demand for such algorithms in the context of autonomous driving scenarios. As a large amount of annotated data is crucial in order to train such deep networks, multiple new datasets have been released to encourage further research in this area, including Synthia [45], Virtual KITTI [18], and Cityscapes [11]. In this work, we focus on Cityscapes, a recent large-scale dataset consisting of real-world imagery with well-curated annotations. Given their success, we will constrain our literature review to deep learning based semantic segmentation approaches and deep learning network architectures.

Semantic Segmentation Approaches. Over the last years, the most successful semantic segmentation approaches have been based on convolutional neural networks (CNNs). Early approaches constrained their output to a bottom-up segmentation followed by a CNN based region classification [54]. Rather than classifying entire regions in the first place, the approach by Farabet *et al*. performs pixel-wise classification using CNN features originating from multiple scales, followed by aggregation of these noisy pixel predictions over superpixel regions [16].

The introduction of so-called *fully convolutional networks (FCNs)* for semantic image segmentation by Long *et al*. [38] opened a wide range of semantic segmentation research using end-to-end training [13]. Long *et al*. further reformulated the popular VGG architecture [50] as a fully convolutional network (FCN), enabling the use of pre-trained models for this architecture. To improve segmentation performance at object boundaries, *skip connections* were added which allow information to propagate directly from early, high-resolution layers to deeper layers.

Pooling layers in FCNs fulfill a crucial role in order to increase the receptive field size of later units and with it the classification performance. However, they have the downside that the resulting network outputs are at a lower resolution. To overcome this, various strategies have been proposed. Some approaches extract features from intermediate layers via some sort of skip connections [38, 8, 36, 7]. Noh *et al*. propose an encoder/decoder network [41]. The encoder computes low-dimensional feature representations via a sequence of pooling and convolution operations. The decoder, which is stacked on top of the encoder, then learns an upscaling of these low-dimensional features via subsequent unpooling and deconvolution operations [56]. Similarly, Badrinarayanan *et al*. [2, 3] use convolutions instead of deconvolutions in the decoder network. In contrast, our approach preserves high-resolution information throughout the entire network by keeping a separate high-resolution processing stream.

Many approaches apply smoothing operations to the output of a CNN in order to obtain more consistent predictions. Most commonly, *conditional random fields (CRFs)* [30] are applied on the network output [9, 8, 12, 34, 6]. More recently, some papers approximate the *mean-field inference* of CRFs using specialized network architectures [57, 48, 37]. Other approaches to smoothing the network predictions include *domain transform* [8, 19] and superpixel-based smoothing [16, 39]. Our approach is able to swiftly combine high- and low-resolution information, resulting in already smooth output predictions. Experiments with additional CRF smoothing therefore did not result in significant performance improvements.

Network architectures. Since the success of the AlexNet architecture [31] in the ImageNet Large-Scale Visual Classification Challenge (ILSVRC) [47], the vision community has seen several milestones with respect to CNN architectures. The network depth has been constantly increased, first with the popular VGG net [50], then by using batch normalization with GoogleNet [51]. Lately, many computer vision applications have adopted the ResNet architecture [25], which often leads to signification performance boosts compared to earlier network architectures. All of these developments show how important a proper architecture is. However, so far most of these networks have been specifically tailored towards the task of classification, in many cases including a pre-training step on ILSVRC. As a result, some of their design choices may contribute to a suboptimal performance when performing pixel-to-pixel tasks such as semantic segmentation. In contrast, our proposed architecture has been specifically designed for segmentation tasks and reaches competitive performance on the Cityscapes dataset without requiring ILSVRC pre-training.

3. Network Architectures for Segmentation

Feed-Forward Networks. Until recently, the majority of feedforward networks, such as the VGG-variants [50], were composed of a linear sequence of layers. Each layer in such a network computes a function $\mathcal{F}$ and the output $\mathbf{x}_n$ of the n-th layer is computed as

$$\mathbf{x}_n = \mathcal{F}(\mathbf{x}_{n-1}; \mathcal{W}_n) \tag{1}$$

where $\mathcal{W}_n$ are the parameters of the layer (see 2a). We refer to this class of network architectures as *traditional feedforward networks*.

Residual Networks (ResNets). He *et al*. observed that deepening traditional feedforward networks often results in an increased training loss [25]. In theory, however, the training loss of a shallow network should be an upper bound on the training loss of a corresponding deep network. This is due to the fact that increasing the depth by adding layers strictly increases the expressive power of the model. A deep network can express all functions that the original shallow network can express by using identity mappings for the added layers. Hence a deep network should perform at least as well as the shallower model on the training data. The violation of this principle implied that current training algorithms have difficulties optimizing very deep traditional feedforward networks. He *et al*. proposed *residual networks (ResNets)* that exhibit significantly improved training characteristics, allowing network depths that were previously unattainable.

A ResNet is composed of a sequence of *residual units (RUs)*. As depicted in Figure 2b, the output $\mathbf{x}_n$ of the n-th RU in a ResNet is computed as

$$\mathbf{x}_n = \mathbf{x}_{n-1} + \mathcal{F}(\mathbf{x}_{n-1}; \mathcal{W}_n) \tag{2}$$

where $\mathcal{F}(\mathbf{x}_{n-1}\,; \mathcal{W}_n)$ is the residual, which is parameterized by $\mathcal{W}_n$. Thus, instead of computing the output $\mathbf{x}_n$ directly, $\mathcal{F}$ only computes a residual that is added to the input $\mathbf{x}_{n-1}$. One commonly refers to this design as *skip connection*, because there is a connection from the input $\mathbf{x}_{n-1}$ to the output $\mathbf{x}_n$ that skips the actual computation $\mathcal{F}$.

It has been empirically observed that ResNets have superior training properties over traditional feedforward networks. This can be explained by an improved gradient flow within the network. In order to understand this, consider the n-th and m-th residual units in a ResNet where $m > n$ (*i.e*., the m-th unit is closer to the output layer of the network). By applying the recursion (2) several times, He *et al*. showed in [26] that the output of the m-th residual unit admits a representation of the form

$$\mathbf{x}_m = \mathbf{x}_n + \sum_{i=n}^{m-1} \mathcal{F}(\mathbf{x}_i; \mathcal{W}_{i+1}). \tag{3}$$

Furthermore, if l is the loss that is used to train the network, we can use the chain rule of calculus and express the derivative of the loss l with respect to the output $\mathbf{x}_n$ of the n-th RU as

$$\frac{\partial l}{\partial \mathbf{x}_n} = \frac{\partial l}{\partial \mathbf{x}_m}\frac{\partial \mathbf{x}_m}{\partial \mathbf{x}_n} = \frac{\partial l}{\partial \mathbf{x}_m} + \frac{\partial l}{\partial \mathbf{x}_m}\sum_{i=n}^{m-1}\frac{\partial \mathcal{F}(\mathbf{x}_i; \mathcal{W}_{i+1})}{\partial \mathbf{x}_n}. \tag{4}$$

Thus, we find

$$\begin{aligned}\frac{\partial l}{\partial \mathcal{W}_n} &= \frac{\partial l}{\partial \mathbf{x}_n}\frac{\partial \mathbf{x}_n}{\partial \mathcal{W}_n} \\ &= \frac{\partial \mathbf{x}_n}{\partial \mathcal{W}_n}\left(\frac{\partial l}{\partial \mathbf{x}_m} + \frac{\partial l}{\partial \mathbf{x}_m}\sum_{i=n}^{m-1}\frac{\partial \mathcal{F}(\mathbf{x}_i; \mathcal{W}_{i+1})}{\partial \mathbf{x}_n}\right).\end{aligned} \tag{5}$$

We see that the weight updates depend on two sources of information, $\frac{\partial l}{\partial \mathbf{x}_m}$ and $\frac{\partial l}{\partial \mathbf{x}_m}\sum_{i=n}^{m-1}\frac{\partial \mathcal{F}(\mathbf{x}_i; \mathcal{W}_{i+1})}{\partial \mathbf{x}_n}$. While the amount of information that is contained in the latter may depend crucially on the depth n, the former allows a gradient

flow that is independent of the depth. Hence, gradients can flow unhindered from the deeper unit to the shallower unit. This makes training even extremely deep ResNets possible.

Full-Resolution Residual Networks (FRRNs). In this paper, we unify the two above-mentioned principles of network design and propose *full-resolution residual networks (FRRNs)* that exhibit the same superior training properties as ResNets but have two processing streams. The features on one stream, the *residual stream*, are computed by adding successive residuals, while the features on the other stream, the *pooling stream*, are the direct result of a sequence of convolution and pooling operations applied to the input.

Our design is motivated by the need to have networks that can jointly compute good high-level features for recognition and good low-level features for localization. Regardless of the specific network design, obtaining good high-level features requires a sequence of pooling operations. The pooling operations reduce the size of the feature maps and increase the network's receptive field, as well as its robustness against small translations in the image. While this is crucial to obtaining robust high-level features, networks that employ a deep pooling hierarchy have difficulties tracking low-level features, such as edges and boundaries, in deeper layers. This makes them good at recognizing the elements in a scene but bad at localizing them to pixel accuracy. On the other hand, a network that does not employ any pooling operations behaves the opposite way. It is good at localizing object boundaries, but performs poorly at recognizing the actual objects. By using the two processing streams together, we are able to compute both kinds of features simultaneously. While the residual stream of an FRRN computes successive residuals at the full image resolution, allowing low level features to propagate effortlessly through the network, the pooling stream undergoes a sequence of pooling and unpooling operations resulting in good high-level features. Figure 1 visualizes the concept of having two distinct processing streams.

An FRRN is composed of a sequence of *full-resolution residual units (FRRUs)*. Each FRRU has two inputs and two outputs, because it simultaneously operates on both streams. Figure 2c shows the structure of an FRRU. Let $\mathbf{z}_{n-1}$ be the residual input to the n-th FRRU and let $\mathbf{y}_{n-1}$ be its pooling input. Then the outputs are computed as

$$\mathbf{z}_n = \mathbf{z}_{n-1} + \mathcal{H}(\mathbf{y}_{n-1}, \mathbf{z}_{n-1}; \mathcal{W}_n) \tag{6}$$

$$\mathbf{y}_n = \mathcal{G}(\mathbf{y}_{n-1}, \mathbf{z}_{n-1}; \mathcal{W}_n), \tag{7}$$

where $\mathcal{W}_n$ are the parameters of the functions $\mathcal{G}$ and $\mathcal{H}$, respectively.

If $\mathcal{G} \equiv 0$, then an FRRU corresponds to an RU since it disregards the pooling input $\mathbf{y}_n$, and the network effectively becomes an ordinary ResNet. On the other hand, if $\mathcal{H} \equiv 0$, then the output of an FRRU only depends on its input via the function $\mathcal{G}$. Hence, no residuals are computed and we obtain a traditional feedforward network. By carefully constructing $\mathcal{G}$ and $\mathcal{H}$, we can combine the two network principles.

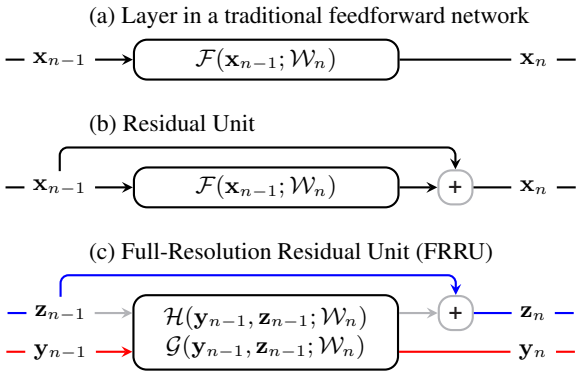

Figure 2. The figure compares the structures of different network design elements. (a) shows a layer in a traditional feedforward network; (b) shows a residual unit; (c) shows a full-resolution residual unit.

In order to show that FRRNs have similar training characteristics as ResNets, we adapt the analysis presented in [26] to our case. Using the same recursive argument as before, we find that for $m > n$, $\mathbf{z}_m$ has the representation

$$\mathbf{z}_m = \mathbf{z}_n + \sum_{i=n}^{m-1} \mathcal{H}(\mathbf{y}_i, \mathbf{z}_i; \mathcal{W}_{i+1}). \tag{8}$$

We can then express the derivative of the loss l with respect to the weights $\mathcal{W}_n$ as

$$\begin{aligned} \frac{\partial l}{\partial \mathcal{W}_n} &= \frac{\partial l}{\partial \mathbf{z}_n}\frac{\partial \mathbf{z}_n}{\partial \mathcal{W}_n} + \frac{\partial l}{\partial \mathbf{y}_n}\frac{\partial \mathbf{y}_n}{\partial \mathcal{W}_n} \\ &= \frac{\partial \mathbf{z}_n}{\partial \mathcal{W}_n}\left(\frac{\partial l}{\partial \mathbf{z}_m} + \frac{\partial l}{\partial \mathbf{z}_m}\sum_{i=n}^{m-1}\frac{\partial \mathcal{H}(\mathbf{y}_i, \mathbf{z}_i; \mathcal{W}_{i+1})}{\partial \mathbf{z}_n}\right) \\ &\quad + \frac{\partial l}{\partial \mathbf{y}_n}\frac{\partial \mathbf{y}_n}{\partial \mathcal{W}_n}. \end{aligned} \tag{9}$$

Hence, the weight updates depend on three sources of information. Analogous to the analysis of ResNets, the two sources $\frac{\partial l}{\partial \mathbf{y}_n}\frac{\partial \mathbf{y}_n}{\partial \mathcal{W}_n}$ and $\frac{\partial l}{\partial \mathbf{z}_m}\sum_{i=n}^{m-1}\frac{\partial \mathcal{H}(\mathbf{y}_i, \mathbf{z}_i; \mathcal{W}_{i+1})}{\partial \mathbf{z}_n}$ depend crucially on the depth n, while the term $\frac{\partial l}{\partial \mathbf{z}_m}$ is independent of the depth. Thus, we achieve a depth-independent gradient flow for all parameters that are used by the residual function $\mathcal{H}$. If we use some of these weights in order to compute the output of $\mathcal{G}$, all weights of the unit benefit from the improved gradient flow. This is most easily achieved by reusing the output of $\mathcal{G}$ in order to compute $\mathcal{H}$. However, we note that other designs are possible.

Figure 3 shows our proposed FRRU design. The unit first concatenates the two incoming streams by using a pooling layer in order to reduce the size of the residual stream. Then the concatenated features are fed through two convolution

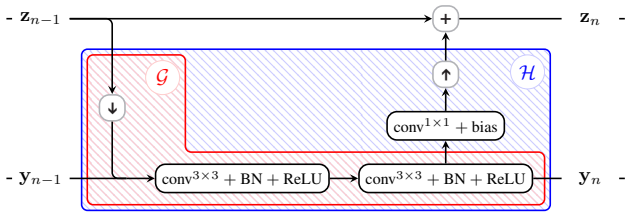

Figure 3. The figure shows our design of a full-resolution residual unit (FRRU). The inner red box marks the parts of the unit that are computed by the function $\mathcal{G}$ while the outer blue box indicates the parts that are computed by the function $\mathcal{H}$.

units. Each convolution unit consists of a 3×3 convolution layer followed by a batch normalization layer [27] and a ReLU activation function. The result of the second convolution unit is used in two ways. First, it forms the *pooling stream* input of the next FRRU in the network and second it is the basis for the computed residual. To this end, we first adjust the number of feature channels using a 1×1 convolution and then upscale the spatial dimensions using an unpooling layer. Because the features might have to be upscaled significantly (*e.g.*, by a factor of 16), we found that simply upscaling by repeating the entries along the spatial dimensions performed superior to bilinear interpolation.

In Figure 3, the inner red box corresponds to the function $\mathcal{G}$ while the outer blue box corresponds to the function $\mathcal{H}$. We can see that the output of $\mathcal{G}$ is used in order to compute $\mathcal{H}$, because the red box is entirely contained within the blue box. As shown above, this design choice results in superior gradient flow properties for all weights of the unit.

Table 1 shows the two network architectures that we used in order to assess our approach's segmentation performance. The proposed architectures are based on several principles employed by other authors. We follow Noh *et al.* [41] and use an encoder/decoder formulation. In the encoder, we reduce the size of the pooling stream using max pooling operations. The pooled feature maps are then successively upscaled using bilinear interpolation in the decoder. Furthermore, similar to Simonyan and Zisserman [50], we define a number of base channels that we double after each pooling operation (up to a certain upper limit). Instead of choosing 64 base channels as in VGG net, we use 48 channels in order to have a manageable number of trainable parameters. Depending on the input image resolution, we use FRRN A or FRRN B to keep the relative size of the receptive fields consistent.

4. Training Procedure

Following Wu *et al.*, we train our network by minimizing a bootstrapped cross-entropy loss [52]. Let c be the number of classes, $y_1, ..., y_N \in \{1, ..., c\}$ be the target class labels for the pixels $1, ..., N$, and let $p_{i,j}$ be the posterior class probability for class j and pixel i. Then, the bootstrapped

FRRN A	FRRN B
$\text{conv}_{48}^{5\times5}$ + BN + ReLU	$\text{conv}_{48}^{5\times5}$ + BN + ReLU
$3 \times \text{RU}_{48}$	$3 \times \text{RU}_{48}$
pooling stream / **residual stream**	**pooling stream** / **residual stream**
max pool / $\text{conv}_{32}^{1\times1}$	max pool / $\text{conv}_{32}^{1\times1}$
$3 \times \text{FRRU}_{96}$	$3 \times \text{FRRU}_{96}$
max pool	max pool
$4 \times \text{FRRU}_{192}$	$4 \times \text{FRRU}_{192}$
max pool	max pool
$2 \times \text{FRRU}_{384}$	$2 \times \text{FRRU}_{384}$
max pool	max pool
$2 \times \text{FRRU}_{384}$	$2 \times \text{FRRU}_{384}$
	max pool
	$2 \times \text{FRRU}_{384}$
	unpool
	$2 \times \text{FRRU}_{192}$
unpool	unpool
$2 \times \text{FRRU}_{192}$	$2 \times \text{FRRU}_{192}$
unpool	unpool
$2 \times \text{FRRU}_{192}$	$2 \times \text{FRRU}_{192}$
unpool	unpool
$2 \times \text{FRRU}_{96}$	$2 \times \text{FRRU}_{96}$
unpool	unpool
pooling stream / **residual stream**	**pooling stream** / **residual stream**
concatenate	concatenate
$3 \times \text{RU}_{48}$	$3 \times \text{RU}_{48}$
$\text{conv}_c^{1\times1}$ + Bias	$\text{conv}_c^{1\times1}$ + Bias
Softmax	Softmax
17.7M parameters	24.8M parameters

Table 1. The table shows our two network designs. By $\text{conv}_m^{k\times k}$ we denote a convolution layer having m kernels each of size $k \times k$. The notations RU_m and FRRU_m refer to residual units and full-resolution residual units whose convolutions have m channels, respectively. The parameter c indicates the number of classes to predict.

cross-entropy loss over K pixels is defined as

$$l = -\frac{1}{K} \sum_{i=1}^{N} \mathbf{1}[p_{i,y_i} < t_K] \log p_{i,y_i} \,, \tag{10}$$

where $\mathbf{1}[x] = 1$ iff x is true and $t_k \in \mathbb{R}$ is chosen such that $|\{i \in \{1, ..., N\} : p_{i,y_i} < t_k\}| = K$. The threshold parameter t_k can easily be determined by sorting the predicted log probabilities and choosing the $K + 1$-th one as threshold. Figure 4 visualizes the concept. Depending on the number of pixels K that we consider, we select misclassified pixels or pixels where we predict the correct label with a small probability. We minimize the loss using ADAM [28].

Because each FRRU processes features at the full image resolution, training a full-resolution residual network is very memory intensive. Recall that in order for the backpropagation algorithm [46] to work, the entire forward pass has to be stored in memory. If the memory required to store the forward pass for a given network exceeds the available GPU memory, we can no longer use the standard backpropagation algorithm. In order to alleviate this problem,

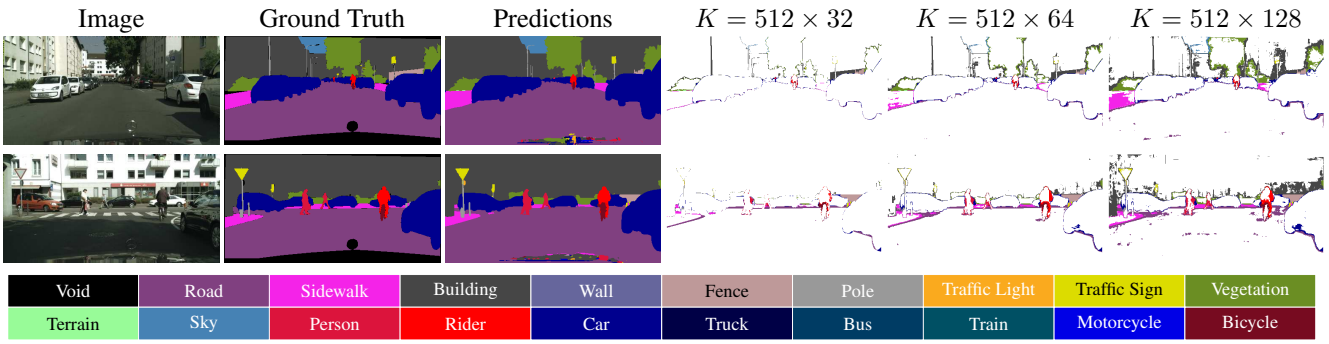

Figure 4. Pixels used by the bootstrapped cross-entropy loss for varying values of K. The images and ground truth annotations originate from the twice-subsampled Cityscapes validation set [11]. Pixels that are labeled *void* are not considered for the bootstrapping process.

we partition the computation graph into several subsequent blocks by manually placing cut points in the graph. We then compute the derivatives for each block individually. To this end, we perform one (partial) forward pass per block and only store the feature maps for the block whose derivatives are computed given the derivative of the subsequent block. This simple scheme allows us to manually control a space-time trade-off. The idea of recomputing some intermediate results on demand is also used in [22] and [10]. Note that these memory limitations only apply during training. During testing, there is no need to store results of each operation in the network and our architecture's memory footprint is comparable to that of a ResNet encoder/decoder architecture. We made code for the gradient computation for arbitrary networks publicly available in Theano/Lasagne.

In order to reduce overfitting, we used two methods of data augmentation: *translation augmentation* and *gamma augmentation*. The former method randomly translates an image and its annotations. In order to keep consistent image dimensions, we have to pad the translated images and annotations. To this end, we use reflection padding on the image and constant padding with void labels on the annotations. Our second method of data augmentation is gamma augmentation. We use a slightly modified gamma augmentation method detailed in the supplementary material.

5. Experimental Evaluation

We implemented our models using Theano/Lasagne [1, 14]. We evaluate our approach on the recently released Cityscapes benchmark [11] containing images recorded in 50 different cities. This benchmark provides 5,000 images with high-quality annotations split up into a training, validation, and test set (2,975, 500, and 1,525 images, respectively). The dense pixel annotations span 30 classes frequently occurring in urban street scenes, out of which 19 are used for actual training and evaluation. Annotations for the test set remain private and comparison to other methods is performed via a dedicated evaluation server.

We report the results of our FRRNs for two settings: FRRN A trained on quarter-resolution (256×512) Cityscapes images; and FRRN B trained on half-resolution (512×1024). We then upsample our predictions using bilinear interpolation in order to report scores at the full image resolution of 1024×2048 pixels. Directly training at the full Cityscapes resolution turned out to be too memory intensive with our current design. However, as our experimental results will show, even when trained only on half-resolution images, our FRRN B's results are competitive with the best published methods trained on full-resolution data. Unless specified otherwise, the reported results are based on the Cityscapes test set. Qualitative results are shown in Figure 6, in the supplementary material, and in our result video[1].

5.1. Residual Network Baseline

Our network architecture can be described as a ResNet [25] encoder/decoder architecture, where the residuals remain at the full input resolution throughout the network. A natural baseline is thus a traditional ResNet encoder/decoder architecture with long-range skip connections [38, 41]. In fact, such an architecture resembles a single deep hourglass module in the stacked hourglass network architecture [40]. This baseline differs from our proposed architecture in two important ways: While the feature maps on our residual stream are processed by each FRRU, the feature maps on the long-range skip connections are not processed by intermediate layers. Furthermore, long-range skip connections are scale dependent, meaning that features at one scale travel over a different skip connection than features at another scale. This is in contrast to our network design, where the residual stream can carry upscaled features from several pooling stages simultaneously.

In order to illustrate the benefits of our approach over the natural baseline, we converted the architecture FRRN A (Table 1a) to a ResNet as follows: We first replaced all FRRUs by RUs and then added skip connections that connect the input of each pooling layer to the output of the corresponding unpooling layer. The resulting ResNet has slightly fewer parameters than the original FRRN (16.7 $\times$

[1] https://www.youtube.com/watch?v=PNzQ4PNZSzc

Method	Subsample	Coarse	Pre-trained	Mean	Road	Sidewalk	Building	Wall	Fence	Pole	Traf. Light	Traf. Sign	Vegetation	Terrain	Sky	Person	Rider	Car	Truck	Bus	Train	Motorcycle	Bicycle
SegNet [2]	×4		✓	57.0	96.4	73.2	84.0	28.5	29.0	35.7	39.8	45.2	87.0	63.8	91.8	62.8	42.8	89.3	38.1	43.1	44.2	35.8	51.9
FRRN A	×4			63.0	97.6	79.1	88.3	32.0	36.4	51.7	57.1	62.5	90.9	69.5	93.3	75.2	51.3	91.6	30.2	43.1	39.2	46.0	62.6
ENet [44]	×2			58.3	96.3	74.2	85.0	32.2	33.2	43.5	34.1	44.0	88.6	61.4	90.6	65.5	38.4	90.6	36.9	50.5	48.1	38.8	55.4
DeepLab [43]	×2	✓	✓	64.8	97.4	78.3	88.1	***47.5***	44.2	29.5	44.4	55.4	89.4	67.3	92.8	71.0	49.3	91.4	***55.9***	66.6	**56.7**	48.1	58.1
FRRN B	×2			**71.8**	**98.2**	**83.3**	**91.6**	45.8	***51.1***	**62.2**	**69.4**	***72.4***	**92.6**	70.0	***94.9***	**81.6**	**62.7**	**94.6**	49.1	***67.1***	***55.3***	53.5	69.5
Dilation [55]	×1		✓	67.1	97.6	79.2	89.9	37.3	47.6	53.2	58.6	65.2	91.8	69.4	93.7	78.9	55.0	93.3	45.5	53.4	47.7	52.2	66.0
Adelaide [34]	×1		✓	***71.6***	***98.0***	***82.6***	90.6	44.0	50.7	51.1	65.0	71.7	92.0	**72.0**	94.1	***81.5***	***61.1***	***94.3***	**61.1**	65.1	53.8	**61.6**	**70.6**
LRR [20]	×1		✓	69.7	97.7	79.9	90.7	44.4	48.6	58.6	***68.2***	72.0	***92.5***	69.3	94.7	**81.6**	60.0	94.0	43.6	56.8	47.2	54.8	***69.7***
LRR [20]	×1	✓	✓	**71.8**	97.9	81.5	***91.4***	**50.5**	**52.7**	***59.4***	66.8	**72.7**	***92.5***	***70.1***	**95.0**	81.3	60.1	***94.3***	51.2	**67.7**	54.6	***55.6***	69.6

Table 2. IoU scores from the Cityscapes test set. We highlight the best published baselines for the different sampling rates. (Additional anonymous submissions exist as concurrent work.) Bold numbers represent the best, italic numbers the second best score for a class. We also indicate the subsampling factor used on the input images, whether additional coarsely annotated data was used, and whether the model was initialized with pre-trained weights.

10^6 vs. 17.7×10^6). This is due to the concatenated features in FRRUs and the additional 1×1 convolutions that connect the pooling to the residual stream.

We train both networks on the quarter-resolution Cityscapes dataset for 45,000 iterations at a batch size of 3. We use a learning rate of 10^{-3} for the first 35,000 iterations and then reduce it to 10^{-4} for the following 10,000 iterations. Both networks converged within these iterations. The FRRN A resulted in a validation set mean IoU score of 65.7% while the ResNet baseline only achieved 62.8%, showing a significant advantage of our FRRNs. Training FRRN B is performed in a similar fashion. Detailed training curves are shown in the supplementary material.

5.2. Quantitative Evaluation

Overview. In Table 2 we compare our method to the best (published) performers on the Cityscapes leader board, namely LRR [20], Adelaide [23], and Dilation [55]. Note that our network performs on par with the very complex and well engineered system by Ghiasi *et al*. (LRR). Among the top performers on Cityscapes, only ENet refrain from using a pre-trained network. However, they target real-time performance and trade accuracy for speed. Thus, they do not obtain top scores. To the best of our knowledge, we are the first to show that it is possible to obtain state-of-the-art results even without pre-training.

Subsampling Factor. An interesting observation that we made on the Cityscapes test set is a correlation between the subsampling factor and the test performance. This correlation can be seen in Figure 5 where we show the scores of several approaches currently listed on the leader board against their respective subsampling factors. Unsurprisingly, most of the best performers operate on the full-resolution input images. Throughout our experiments, we consistently outperformed other approaches who trained on

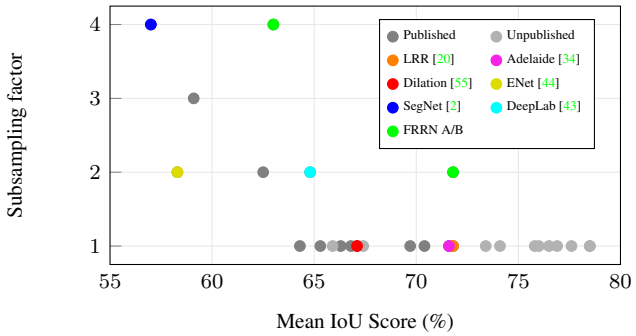

Figure 5. Comparison of the mean IoU scores of all approaches on the leader board of the Cityscapes segmentation benchmark based on the subsampling factor of the images that they were trained on.

the same image resolutions. Even though we only train on half-resolution images, Figure 5 clearly shows we can match the current published state-of-the-art (LRR [20]). We expect that further improvements can be obtained by switching to full-resolution training.

5.3. Boundary Adherence

Due to several pooling operations (and subsequent upsampling) in many of today's FCN architectures, boundaries are often overly smooth, resulting in lost details and edge-bleeding. This leads to suboptimal scores, but it also makes the output of a semantic segmentation approach harder to use without further post-processing. Since inaccurate boundaries are often not apparent from the standard evaluation metric scores, a typical approach is a trimap evaluation in order to quantify detailed boundary adherence [29, 30, 20]. During trimap evaluation, all predictions are ignored if they do not fall within a certain radius r of a ground truth label boundary. Figure 7 visualizes our trimap evaluation performed on the validation set for varying trimap widths r between 1 and 80 pixels. We compare to LRR [20] and Dilation [55], who made code and pre-trained

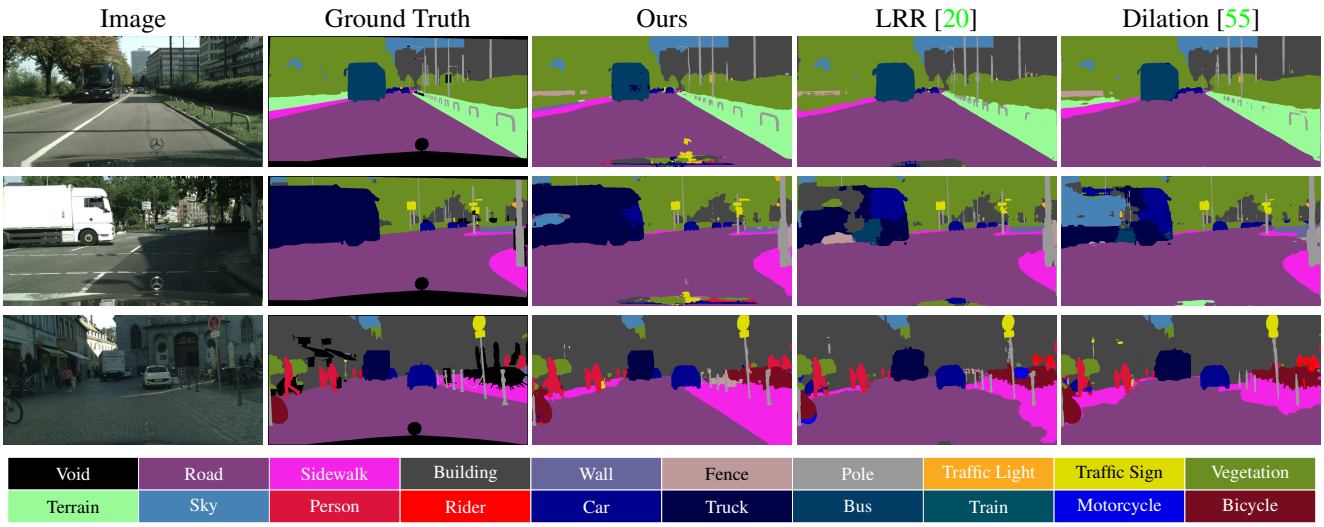

Figure 6. Qualitative comparison on the Cityscapes validation set. Interesting cases are the fence in the first row, the truck in the second row, or the street light poles in the last row. An interesting failure case is shown in the third row: all methods struggle to find the correct sidewalk boundary, however our network makes a clean and reasonable prediction. Please consult the supplemental material for more qualitative results.

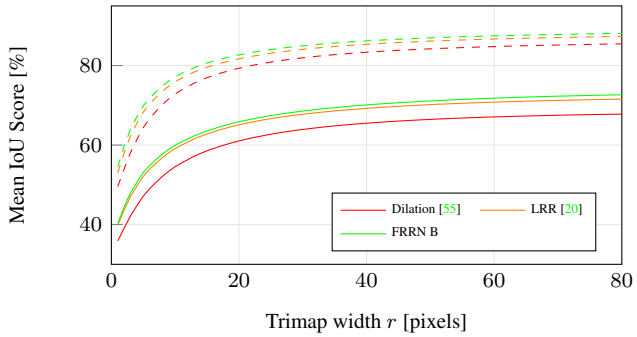

Figure 7. The trimap evaluation on the validation set. The solid lines show the mean IoU score of our approach and two top performing methods that released their code. The dashed lines show the mean IoU score when using the 7 Cityscapes category labels.

models available. We see that our approach outperforms the competition consistently for all radii r. Furthermore, it should be noted that the method of [20] is based on an architecture specifically designed for clean boundaries. Our method achieves better boundary adherence, both numerically and qualitatively (see Figure 6), with a much simpler architecture and without ImageNet pre-training.

Often one can boost both the numerical score and the boundary adherence by using a fully connected CRF as post-processing step. We tried to apply a fully connected CRF with Gaussian kernel, as introduced by Krähenbühl and Koltun [30]. We used the standard appearance and smoothness kernels and tuned parameters on the validation set by running several thousand Hyperopt iterations [5]. Applying this post processing step yielded a marginal increase in the average IoU score of $\sim 0.5\%$ on the validation set. This further supports the claim that our architecture natively solves problems that were conventionally addressed using costly post processing steps. Given the high computation time and low yield we decided against any post-processing steps.

5.4. Runtime

A forward pass of our proposed architectures FRRN A and FRRN B takes 0.166s and 0.469s on images of size 256×512 and 512×1024, respectively. This compares to 0.081s of the ResNet architecture on images of size 256×512. All measurements were averaged over 100 individual forward passes on an NVidia Titan X Pascal GPU. While FRRNs are indeed slower than the ResNet baseline, they are faster or as fast as other methods on the Cityscapes leaderboard that report runtimes.

6. Conclusion

In this paper we propose a novel network architecture for semantic segmentation in street scenes. Our architecture is clean, does not require additional post-processing, can be trained from scratch, shows superior boundary adherence, and reaches state-of-the-art results on the Cityscapes benchmark. We published code and all trained models on GitHub[2]. Since we do not incorporate design choices specifically tailored towards semantic segmentation, we believe that our architecture will also be applicable to other tasks such as stereo or optical flow where predictions are performed per pixel.

Acknowledgment. This work was funded by the EU project STRANDS (ICT-2011-600623) and the ERC Starting Grant project CV-SUPER (ERC-2012-StG-307432).

[2]https://github.com/TobyPDE/FRRN

References

[1] R. Al-Rfou, G. Alain, A. Almahairi, et al. Theano: A Python framework for fast computation of mathematical expressions. abs/1605.02688, 2016. 6

[2] V. Badrinarayanan, A. Handa, and R. Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Robust Semantic Pixel-Wise Labelling. arXiv:1505.07293, 2015. 2, 7

[3] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegmentationNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. arXiv:1511.00561, 2015. 2

[4] M. Bansal, B. Matei, H. Sawhney, S.-H. Jung, and J. Eledath. Pedestrian Detection with Depth-Guided Structure Labeling. In *ICCV Workshop*, 2009. 1

[5] J. Bergstra, D. Yamins, and D. D. Cox. Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. In *ICML*, 2013. 8

[6] S. Chandra and I. Kokkinos. Fast, Exact and Multi-Scale Inference for Semantic Image Segmentation with Deep Gaussian CRFs. arXiv:1603.08358, 2016. 3

[7] H. Chen, Q. Dou, L. Yu, and P. Heng. VoxResNet: Deep Voxelwise Residual Networks for Volumetric Brain Segmentation. arXiv:1608.05895, 2016. 2

[8] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. ICLR, 2015. 2, 3

[9] 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. arXiv:1606.00915, 2016. 3

[10] T. Chen, B. Xu, C. Zhang, and C. Guestrin. Training Deep Nets with Sublinear Memory Cost. arXiv:1604.06174, 2016. 6

[11] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *CVPR*, 2016. 2, 6

[12] J. Dai, K. He, and J. Sun. BoxSup: Exploiting Bounding Boxes to Supervise Convolutional Networks for Semantic Segmentation. In *ICCV*, 2015. 3

[13] J. Dai, K. He, and J. Sun. Convolutional Feature Masking for Joint Object and Stuff Segmentation. In *CVPR*, 2015. 2

[14] S. Dieleman, J. Schlter, C. Raffel, et al. Lasagne: First release., Aug. 2015. 6

[15] A. Ess, T. Müller, H. Grabner, and L. Van Gool. Segmentation-Based Urban Traffic Scene Understanding. In *BMVC*, 2009. 1

[16] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning Hierarchical Features for Scene Labeling. *PAMI*, 35(8), 2013. 2, 3

[17] G. Floros and B. Leibe. Joint 2d-3d temporally consistent semantic segmentation of street scenes. In *CVPR*, pages 2823–2830. IEEE, 2012. 1

[18] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual Worlds as Proxy for Multi-Object Tracking Analysis. In *CVPR*, 2016. 2

[19] E. S. L. Gastal and M. M. Oliveira. Domain Transform for Edge-Aware Image and Video Processing. In *ACM Trans. Graphics*, 2011. 3

[20] G. Ghiasi and C. C. Fowlkes. Laplacian Reconstruction and Refinement for Semantic Segmentation. In *ECCV*, 2016. 1, 2, 7, 8

[21] S. Gould, J. Rodgers, D. Cohen, G. Elidan, and D. Koller. Multi-class segmentation with relative location prior. *IJCV*, 80(3):300–316, 2008. 1

[22] A. Gruslys, R. Munos, I. Danihelka, M. Lanctot, and A. Graves. Memory-Efficient Backpropagation Through Time. arXiv:1606.03401, 2016. 6

[23] C. Gu, J. J. Lim, P. Arbeláez, and J. Malik. Recognition using Regions. In *CVPR*. IEEE, 2009. 1, 7

[24] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous Detection and Segmentation. In *ECCV*. Springer, 2014. 1

[25] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016. 1, 3, 6

[26] K. He, X. Zhang, S. Ren, and J. Sun. Identity Mappings in Deep Residual Networks. In *ECCV*, 2016. 3, 4

[27] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*, 2015. 2, 5

[28] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015. 5

[29] P. Kohli, P. H. Torr, et al. Robust Higher Order Potentials for Enforcing Label Consistency. *IJCV*, 82(3):302–324, 2009. 7

[30] P. Krähenbühl and V. Koltun. Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials. In *NIPS*, 2011. 2, 3, 7, 8

[31] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet Classification with Deep Convolutional Networks. In *NIPS*, 2012. 3

[32] A. Kundu, Y. Li, F. Dellaert, F. Li, and J. M. Rehg. Joint semantic segmentation and 3d reconstruction from monocular video. In *ECCV*, pages 703–718, 2014. 1

[33] L. Ladicky, J. Shi, and M. Pollefeys. Pulling things out of perspective. In *CVPR*, pages 89–96, 2014. 1

[34] G. Lin, C. Shen, I. D. Reid, and A. van den Hengel. Efficient piecewise training of deep structured models for semantic segmentation. In *CVPR*, 2016. 3, 7

[35] B. Liu, S. Gould, and D. Koller. Single image depth estimation from predicted semantic labels. In *CVPR*, pages 1253–1260. IEEE, 2010. 1

[36] W. Liu, A. Rabinovich, and A. C. Berg. ParseNet: Looking Wider to See Better. 2015. 2

[37] Z. Liu, X. Li, P. Luo, C. C. Loy, and X. Tang. Semantic Image Segmentation via Deep Parsing Network. In *ICCV*, 2015. 3

[38] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. In *CVPR*, 2015. 1, 2, 6

[39] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich. Feedforward Semantic Segmentation With Zoom-Out Features. In *CVPR*, 2015. 3

[40] A. Newell, K. Yang, and J. Deng. Stacked Hourglass Networks for Human Pose Estimation. In *ECCV*, 2016. 6

[41] H. Noh, S. Hong, and B. Han. Learning Deconvolution Network for Semantic Segmentation. In *ICCV*, 2015. 2, 5, 6

[42] A. Ošep, A. Hermans, F. Engelmann, D. Klostermann, M. Mathias, and B. Leibe. Multi-Scale Object Candidates for Generic Object Tracking in Street Scenes. In *ICRA*, 2016. 1

[43] G. Papandreou, L. Chen, K. P. Murphy, and A. L. Yuille. Weakly-and Semi-Supervised Learning of a Deep Convolutional Network for Semantic Image Segmentation. In *ICCV*, 2015. 7

[44] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation. arXiv:1606.02147, 2016. 7

[45] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. Lopez. The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes. In *CVPR*, 2016. 2

[46] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323, 1986. 5

[47] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3), 2015. 3

[48] A. G. Schwing and R. Urtasun. Fully Connected Deep Structured Networks. arXiv:1503.02351, 2015. 3

[49] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for Image categorization and segmentation. In *CVPR*, 2008. 1

[50] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*, 2015. 1, 2, 3, 5

[51] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going Deeper with Convolutions. In *CVPR*, 2015. 3

[52] Z. Wu, C. Shen, and A. v. d. Hengel. Bridging Category-level and Instance-level Semantic Image Segmentation. arXiv:1605.06885, 2016. 5

[53] J. Xiao and L. Quan. Multiple view semantic segmentation for street view images. In *ICCV*. IEEE, 2009. 1

[54] J. Yan, Y. Yu, X. Zhu, Z. Lei, and S. Z. Li. Object Detection by Labeling Superpixels. In *CVPR*, 2015. 2

[55] F. Yu and V. Koltun. Multi-Scale Context Aggregation by Dilated Convolutions. In *ICLR*, 2016. 2, 7, 8

[56] M. D. Zeiler, G. W. Taylor, and R. Fergus. Adaptive Deconvolutional Networks for Mid and High Level Feature Learning. In *CVPR*, 2011. 2

[57] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr. Conditional Random Fields as Recurrent Neural Networks. In *ICCV*, 2015. 3

[58] Y. Zhu, R. Urtasun, R. Salakhutdinov, and S. Fidler. segDeepM: Exploiting Segmentation and Context in Deep Neural Networks for Object Detection. In *CVPR*, 2015. 1