Superpixel Convolutional Networks using Bilateral Inceptions

Raghudeep Gadde[1⋆], Varun Jampani[2⋆], Martin Kiefel[2,3], Daniel Kappler[2], and Peter V. Gehler[2,3]

[1]Université Paris-Est, LIGM (UMR 8049), CNRS, ENPC, ESIEE, UPEM, France
[2]Max Planck Institute for Intelligent Systems, Tübingen, Germany
[3]Bernstein Center for Computational Neuroscience, Tübingen, Germany

Abstract. In this paper we propose a CNN architecture for semantic image segmentation. We introduce a new "bilateral inception" module that can be inserted in existing CNN architectures and performs bilateral filtering, at multiple feature-scales, between superpixels in an image. The feature spaces for bilateral filtering and other parameters of the module are learned end-to-end using standard backpropagation techniques. The bilateral inception module addresses two issues that arise with general CNN segmentation architectures. First, this module propagates information between (super) pixels while respecting image edges, thus using the structured information of the problem for improved results. Second, the layer recovers a full resolution segmentation result from the lower resolution solution of a CNN. In the experiments, we modify several existing CNN architectures by inserting our inception module between the last CNN (1×1 convolution) layers. Empirical results on three different datasets show reliable improvements not only in comparison to the baseline networks, but also in comparison to several dense-pixel prediction techniques such as CRFs, while being competitive in time.

1 Introduction

In this paper we propose a CNN architecture for semantic image segmentation. Given an image $\mathcal{I} = (x_1, \ldots, x_N)$ with N pixels x_i the task of semantic segmentation is to infer a labeling $Y = (y_1, \ldots, y_N)$ with a label $y_i \in \mathcal{Y}$ for every pixel. This problem can be naturally formulated as a structured prediction problem $g : \mathcal{I} \rightarrow Y$. Empirical performance is measured by comparing Y to a human labeled Y^* via a loss function $\Delta(Y, Y^*)$, *e.g.*, with the Intersection over Union (IoU) or pixel-wise Hamming Loss.

A direct way to approach this problem would be to ignore the structure of the output variable Y and train a classifier that predicts the class membership of the center pixel of a given image patch. This procedure reduces the problem to a standard multi-class classification problem and allows the use of standard learning algorithms. The resulting classifier is then evaluated at every possible

⋆ The first two authors contribute equally to this work.

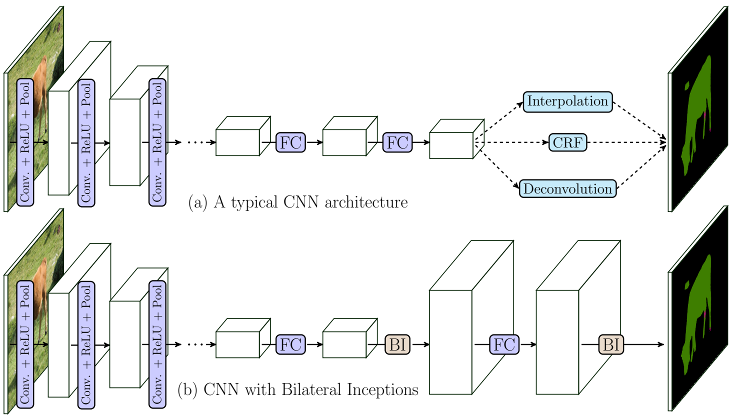

Fig. 1. Illustration of CNN layout. We insert the *Bilateral Inception (BI)* modules between the *FC* (1×1 convolution) layers found in most networks thus removing the necessity of further up-scaling algorithms. Bilateral Inception modules also propagate information between distant pixels based on their spatial and color similarity and work better than other label propagation approaches.

patch in a sliding window fashion (or using coarse-to-fine strategies) to yield a full segmentation of the image. With high capacity models and large amounts of training data this approach would be sufficient, given that the loss decomposes over the pixels. Such a per-pixel approach ignores the relationship between the variables $(y_1, \ldots, y_N)$, which are not i.i.d. since there is an underlying common image. Therefore, besides learning discriminative per-pixel classifiers, most segmentation approaches further encode the output relationship of Y. A dominating approach is to use Conditional Random Fields (CRF) [1], which allows an elegant and principled way to combine single pixel predictions and shared structure through unary, pairwise and higher order factors.

What relates the outputs $(y_1, \ldots, y_N)$? The common hypothesis that we use in this paper could be summarized as: *Pixels that are spatially and photometrically similar are more likely to have the same label.* Particularly if two pixels x_i, x_j are close in the image and have similar *RGB* values, then their corresponding labels y_i, y_j will most likely be the same. The most prominent example of spatial similarity encoded in a CRF is the Potts model (Ising model for the binary case). The work of [2] described a densely connected pairwise CRF (DenseCRF) that includes pairwise factors encoding both spatial *and* photometric similarity. The DenseCRF has been used in many recent works on image segmentation which find also empirically improved results over pure pixel-wise CNN classifiers [3,4,5,6].

In this paper, we implement the above-mentioned hypothesis of photometrically similar and near-by pixels share common labels, by designing a new "Bilateral Inception" (BI) module that can be inserted before/after the last 1×1 convolution layers (which we refer to as 'FC' layers - 'Fully-Connected' in the original image classification network) of the standard segmentation CNN architectures. The bilateral inception module does edge-aware information propagation across different spatial CNN units of the previous FC layer. Instead of using the spatial grid-layout that is common in CNNs, we incorporate the superpixel-layout for information propagation. The information propagation is performed using standard bilateral filters with Gaussian kernels, at different feature scales. This construction is inspired by [7,8]. Feature spaces and other parameters of the modules can be learned end-to-end using standard backpropagation techniques. The application of superpixels reduces the number of necessary computations and implements a long-range edge-aware inference between different superpixels. Moreover, since superpixels provides an output at the full image resolution it removes the need for any additional post-processing step.

We introduce BI modules in the CNN segmentation models of [3,5,4]. See Fig. 1 for an illustration. This achieves better segmentation results on all three datasets we experimented with than the proposed interpolation/inference techniques of DenseCRF [4,3] while being faster. Moreover, the results compare favorably against some recently proposed dense pixel prediction techniques. As illustrated in Fig. 1, the BI modules provides an alternative approach to commonly used up-sampling and CRF techniques.

2 Related Work

The literature on semantic segmentation is large and therefore we will limit our discussion to those works that perform segmentation with CNNs and discuss the different ways to encode the output structure.

A natural combination of CNNs and CRFs is to use the CNN as unary potential and combine it with a CRF that also includes pairwise or higher order factors. For instance [3,4] observed large improvements in pixel accuracy when combining a DenseCRF [2] with a CNN. The mean-field steps of the DenseCRF can be learned and back-propagated as noted by [9] and implemented by [5,10,11,12] for semantic segmentation and [13] for human pose estimation. The works of [14,15,16] use CNNs also in pairwise and higher order factors for more expressiveness. The recent work of [6] replaced the costly DenseCRF with a faster domain transform performing smoothing filtering while predicting the image edge maps at the same time. Our work was inspired by DenseCRF approaches but with the aim to replace the expensive mean-field inference. Instead of propagating information across unaries obtained by a CNN, we aim to do the edge-aware information propagation across *intermediate* representations of the CNN. Experiments on different datasets indicate that the proposed approach generally gives better results in comparison to DenseCRF while being faster.

A second group of works aims to inject the structural knowledge in intermediate CNN representations by using structural layers among CNN internal layers. The deconvolution layers model from [17] are being widely used for local propagation of information. They are computationally efficient and are used in segmentation networks, for *e.g.* [18]. They are however limited to small receptive fields. Another architecture proposed in [19] uses spatial pyramid pooling layers to max-pool over different spatial scales. The work of [20] proposed specialized structural layers such as normalized-cut layers with matrix back-propagation techniques. All these works have either fixed local receptive fields and/or have their complexity increasing exponentially with longer range pixel connections. Our technique allows for modeling long range (super-) pixel dependencies without compromising the computational efficiency. A very recent work [21] proposed the use of dilated convolutions for propagating multi-scale contextual information among CNN units.

A contribution of this work is to define convolutions over superpixels by defining connectivity among them. In [22], a method to use superpixels inside CNNs has been proposed by re-arranging superpixels based on their features. The technique proposed here is more generic and alleviates the need for rearranging superpixels. A method to filter irregularly sampled data has been developed in [23] which may be applicable to superpixel convolutions. The difference being that their method requires a pre-defined graph structure for every example/image separately while our approach directly works on superpixels. We experimented with Isomap embeddings [24] of superpixels but for speed reasons opted for the more efficient kernels presented in this paper. The work of [25] extracted multi-scale features at each superpixel and perform semantic segmentation by classifying each superpixel independently. In contrast, we propagate information across superpixels by using bilateral filters with learned feature spaces.

Another core contribution of this work is the end-to-end trained bilateral filtering module. Several recent works on bilateral filtering [26,27,28,10] back-propagate through permutohedral lattice approximation [29], to either learn the filter parameters [28,10] or do optimization in the bilateral space [26,27]. Most of the existing works on bilateral filtering use pre-defined feature spaces. In [30], the feature spaces for bilateral filtering are obtained via a non-parametric embedding into an Euclidean space. In contrast, by explicitly computing the bilateral filter kernel, we are able to back-propagate through features, thereby learning the task-specific feature spaces for bilateral filters through integration into end-to-end trainable CNNs.

3 Superpixel Convolutional Networks

We first formally introduce superpixels in Sec. 3.1 before we describe the bilateral inception modules in Sec. 3.2.

3.1 Superpixels

The term *superpixel* refers to a set of n_i pixels $S_i = \{t_1, \dots, t_{n_i}\}$ with $t_k \in \{1, \dots, N\}$ pixels. We use a set of M superpixels $S = \{S_1, \dots, S_M\}$ that are disjoint $S_i \cap S_j = \emptyset, \forall i, j$ and decompose the image, $\cup_i S_i = \mathcal{I}$.

Superpixels have long been used for image segmentation in many previous works, *e.g.* [31,32,33,25], as they provide a reduction of the problem size. Instead of predicting a label y_i for every pixel x_i, the classifier predicts a label y_i per superpixel S_i and extends this label to all pixels within. A superpixel algorithm can pre-group pixels based on spatial and photometric similarity, reducing the number of elements and also thereby regularizing the problem in a meaningful way. The downside is that superpixels introduce a quantization error whenever pixels within one segment have different ground truth label assignments.

Figure 2 shows the superpixel quantization effect with the best achievable performance as a function in the number of superpixels, on two different segmentation datasets: PascalVOC [34] and Materials in Context [4]. We find that the quantization effect is small compared to the current best segmentation performance. Practically, we use the SLIC superpixels [35] for their runtime and [36] for their lower quantization error to decompose the image into superpixels. For details of the algorithms, please refer to the respective papers. We use publicly-available real-time GPU implementation of SLIC, called gSLICr [37], which runs at over 250Hz per second. And the publicly available Dollar superpixels code [36] computes a super-pixelization for a 400×500 image in about 300ms using an Intel Xeon 3.33GHz CPU.

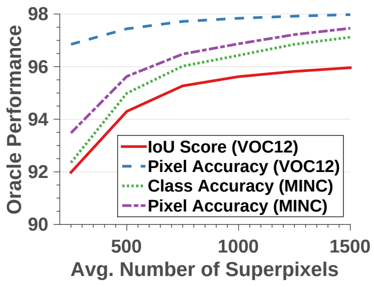

Fig. 2. Superpixel Quantization Error. Best achievable segmentation performance with a varying number of superpixels on Pascal VOC12 segmentation [34] and MINC material segmentation [4] datasets.

3.2 Bilateral Inceptions

Next, we describe the *Bilateral Inception Module* (BI) that performs Gaussian Bilateral Filtering on multiple scales of the representations within a CNN. The BI module can be inserted in between layers of existing CNN architectures.

Bilateral Filtering: We first describe the Gaussian bilateral filtering, the building block of the BI module. A visualisation of the necessary computations is shown in Fig. 3. Given the previous layer CNN activations $\mathbf{z} \in \mathbb{R}^{P \times C}$, that is P points and C filter responses. With $\mathbf{z}_c \in \mathbb{R}^P$ we denote the vector of activations of filter c. Additionally we have for every point j a feature vector $\mathbf{f}_j \in \mathbb{R}^D$. This denotes its spatial position ($D = 2$, not necessarily a grid), position and RGB color ($D = 5$), or others. Separate from the input points with features $F_{in} = \{\mathbf{f}_1, \dots, \mathbf{f}_P\}$ we have Q output points with features F_{out}. These can be the same set of points, but also fewer ($Q < P$), equal ($Q = P$), or more ($Q > P$) points. For example we can filter a 10×10 grid ($P = 100$) and produce the result on a 50×50 grid ($Q = 2500$) or vice versa.

The bilateral filtered result will be denoted as $\hat{\mathbf{z}} \in \mathbb{R}^{Q \times C}$. We apply the same Gaussian bilateral filter to every channel c separately. A filter has two free

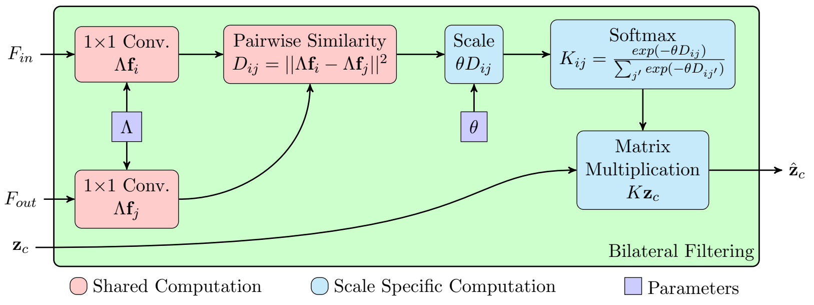
F_{in}
1×1 Conv. $\Lambda\mathbf{f}_i$
Pairwise Similarity $D_{ij} = ||\Lambda\mathbf{f}_i - \Lambda\mathbf{f}_j||^2$
Scale θD_{ij}
Softmax $K_{ij} = \frac{exp(-\theta D_{ij})}{\sum_{j'} exp(-\theta D_{ij'})}$
Λ
θ
Matrix Multiplication $K\mathbf{z}_c$
$\hat{\mathbf{z}}_c$
F_{out}
1×1 Conv. $\Lambda\mathbf{f}_j$
$\mathbf{z}_c$
Bilateral Filtering
Shared Computation
Scale Specific Computation
Parameters

Fig. 3. Computation flow of the Gaussian Bilateral Filtering. We implemented the bilateral convolution with five separate computation blocks. Λ and θ are the free parameters.

parameters: the filter specific scale $\theta \in \mathbb{R}_+$ and the global feature transformation parameters $\Lambda \in \mathbb{R}^{D\times D}$. For Λ, a more general scaling could be applied using more features or a separate CNN. Technically the bilateral filtering amounts to a matrix-vector multiplication $\forall c$:

$$\hat{\mathbf{z}}_c = K(\theta, \Lambda, F_{in}, F_{out})\mathbf{z}_c, \tag{1}$$

where $K \in \mathbb{R}^{Q\times P}$ and values for $f_i \in F_{out}, f_j \in F_{in}$:

$$K_{i,j} = \frac{\exp(-\theta\|\Lambda\mathbf{f}_i - \Lambda\mathbf{f}_j\|^2)}{\sum_{j'}\exp(-\theta\|\Lambda\mathbf{f}_i - \Lambda\mathbf{f}_{j'}\|^2)}. \tag{2}$$

From a kernel learning terminology, K is nothing but a Gaussian Gram matrix and it is symmetric if $F_{in} = F_{out}$. We implemented this filtering in Caffe [38] using different layers as depicted in Fig. 3. While approximate computations of $K\mathbf{z}_c$ exist and have improved runtime [29,39,40,41], we chose an explicit computation of K due to its small size. Our implementation makes use of GPU and the intermediate pairwise similarity computations are re-used across different modules. The entire runtime is only a fraction of the CNN runtime, but of course applications to larger values of P and Q would require aforementioned algorithmic speed-ups.

Bilateral Inception Module: The *bilateral inception module* (BI) is a weighted combination of different bilateral filters. We combine the output of H different filter kernels K, with different scales $\theta^1, \ldots, \theta^H$. All kernels use the same feature transformation Λ which allows for easier pre-computation of pairwise difference and avoids an over-parametrization of the filters. The outputs of different filters $\hat{\mathbf{z}}^h$ are combined linearly to produce $\bar{\mathbf{z}}$:

$$\bar{\mathbf{z}}_c = \sum_{h=1}^{H} \mathbf{w}_c^h \hat{\mathbf{z}}_c^h, \tag{3}$$

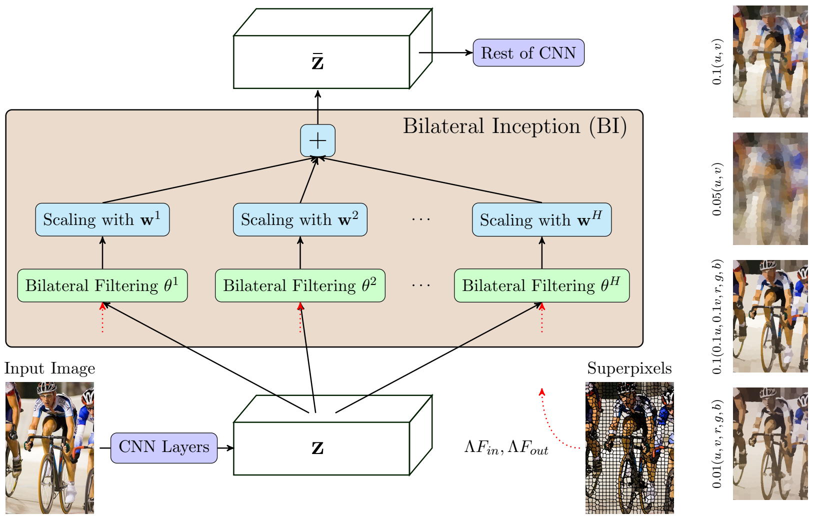
$\bar{\mathbf{z}}$
Rest of CNN
Bilateral Inception (BI)
+
Scaling with $\mathbf{w}^1$
Scaling with $\mathbf{w}^2$
Scaling with $\mathbf{w}^H$
Bilateral Filtering θ^1
Bilateral Filtering θ^2
Bilateral Filtering θ^H
Input Image
CNN Layers
$\mathbf{z}$
$\Lambda F_{in}, \Lambda F_{out}$
Superpixels
$0.1(u,v)$
$0.05(u,v)$
$0.1(0.1u,0.1v,r,g,b)$
$0.01(u,v,r,g,b)$

Fig. 4. Visualization of a Bilateral Inception (BI) Module. The unit activations $\mathbf{z}$ are passed through several bilateral filters defined over different feature spaces. The result is linearly combined to $\bar{\mathbf{z}}$ and passed on to the next network layer. Also shown are sample filtered superpixel images using bilateral filters defined over different example feature spaces. (u,v) correspond to position and (r,g,b) correspond to color features.

using individual weights $\mathbf{w}_c^h$ per scale θ^h and channel c. The weights $\mathbf{w} \in \mathbb{R}^{H \times C}$ are learned using error-backpropagation. The result of the inception module has C channels for every of its Q points, thus $\bar{\mathbf{z}} \in \mathbb{R}^{Q \times C}$. The inception module is schematically illustrated in Fig. 4. In short, information from CNN layers below is filtered using bilateral filters defined in transformed feature space ($\Lambda \mathbf{f}$). Most operations in the inception module are parallelizable resulting in fast runtimes on a GPU. In this work, inspired from the DenseCRF architecture from [2], we used pairs of BI modules: one with position features (u,v) and another with both position and colour features (u,v,r,g,b), each with multiple scales $\{\theta^h\}$.

Motivation and Comparison to DenseCRF: A BI module filters the activations of a CNN layer. Contrast this with the use of a DenseCRF on the CNN output. At that point the fine-grained information that intermediate CNN layers represent has been condensed already to a low-dimensional vector representing beliefs over labels. Using a mean-field update is propagating information between these beliefs. Similar behaviour is obtained using the BI modules but on different scales (using multiple different filters $K(\theta^h)$) and on the intermediate CNN activations $\mathbf{z}$. Since in the end, the to-be-predicted pixels are not i.i.d., this blurring leads to better performance both when using a bilateral filter as an approximate message passing step of a DenseCRF as well in the system outlined here. Both attempts are encoding prior knowledge about the problem, namely that pixels close in position and color are likely to have the same label. Therefore such pixels can also have the same intermediate representation. Consider one would average CNN representations for all pixels that have the same ground truth label. This would result in an intermediate CNN representation that would be very easy to classify for the later layers.

3.3 Superpixel Convolutions

The bilateral inception module allows to change how information is stored in the higher level of a CNN. This is where the superpixels are used. Instead of storing information on a fixed grid, we compute for every image, superpixels S and use the mean color and position of their included pixels as features. We can insert bilateral inception modules to change from grid representations to superpixel representations and vice versa. Inception modules in between superpixel layers convolve the unit activations between all superpixels depending on their distance in the feature space. This retains all properties of the bilateral filter, superpixels that are spatially close and have a similar mean color will have a stronger influence on each other.

Superpixels are not the only choice, in principle one can also sample random points from the image and use them as intermediate representations. We are using superpixels for computational reasons, since they can be used to propagate label information to the full image resolution. Other interpolation techniques are possible, including the well known bilinear interpolation, up-convolution networks [17], and DenseCRFs [2]. The quantization error mentioned in Sec. 3.1 only enters because the superpixels are used for interpolation. Also note that a fixed grid, that is independent of the image is a hard choice of where information should be stored. One could in principle evaluate the CNN densely, at all possible spatial locations, but we found that this resulted in poor performance compared to interpolation methods.

Backpropagation and Training. All free parameters of the inception module $\mathbf{w}$, $\{\theta^h\}$ and Λ are learned via backpropagation. We also backpropagate the error with respect to the module inputs thereby enabling the integration of our inception modules inside CNN frameworks without breaking the end-to-end learning paradigm. As shown in Fig. 3, the bilateral filtering can be decomposed into 5 different sub-layers. Derivatives with respect to the open parameters are obtained by the corresponding layer and standard backpropagation through the directed acyclic graph. For example, Λ is optimized by back-propagating gradients through 1×1 convolution. Derivatives for non-standard layers (pairwise similarity, matrix multiplication) are straight forward to obtain using matrix calculus. To let different filters learn the information propagation at different scales, we initialized $\{\theta^h\}$ with well separated scalar values (*e.g.* $\{1, 0.7, 0.3, ...\}$). The learning is performed using Adam stochastic optimization method [42]. The implementation is done in Caffe neural network framework [38], and the code is available online at http://segmentation.is.tuebingen.mpg.de.

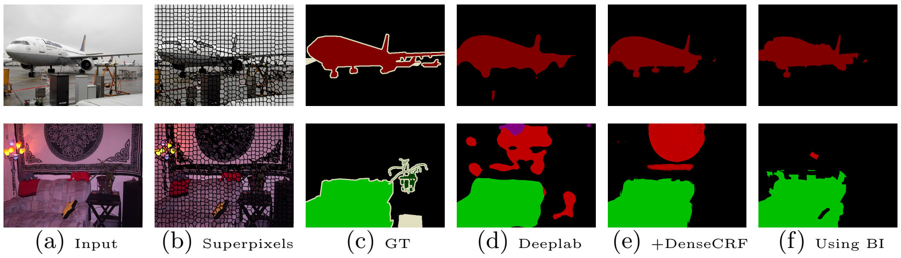

Fig. 5. Semantic Segmentation. Example results of semantic segmentation on Pascal VOC12 dataset. (d) depicts the DeepLab CNN result, (e) CNN + 10 steps of mean-field inference, (f) result obtained with bilateral inception (BI) modules ($BI_6(2)+BI_7(6)$) between FC layers.

4 Experiments

We study the effect of inserting and learning bilateral inception modules in various existing CNN architectures. As a testbed we perform experiments on semantic segmentation using the Pascal VOC2012 segmentation benchmark dataset [34], Cityscapes street scene dataset [43] and on material segmentation using the Materials in Context (MINC) dataset from [4]. We take different CNN architectures from the works of [3,5,4] and insert the inception modules before and/or after the spatial FC layers. In the supplementary, we presented some quantitative results with approximate bilateral filtering using the permutohedral lattice [29].

4.1 Semantic Segmentation

We first use the Pascal VOC12 segmentation dataset [34] with 21 object classes. For all experiments on VOC2012, we train using the extended training set of 10581 images collected by [44]. Following [5], we use a reduced validation set of 346 images for validation. We experiment on two different network architectures, (a) DeepLab model from [3] which uses CNN followed by DenseCRF and (b) CRFasRNN model from [5] which uses CNN with deconvolution layers followed by DenseCRF trained end-to-end.

DeepLab We use the publicly available state-of-the-art pre-trained CNN models from [3]. We use the DeepLab-LargeFOV variant as a base architecture and refer to it as 'DeepLab'. The DeepLab CNN model produces a lower resolution prediction ($\frac{1}{8}\times$) which is then bilinearly interpolated to the input image resolution. The original models have been fine-tuned using both the MSCOCO [45] and the extended VOC [44] datasets. Next, we describe modifications to these models and show performance improvements in terms of both IoU and runtimes.

Model	Training	*IoU*	*Runtime*
DeepLab [3]		68.9	145ms
With BI modules			
$BI_6(2)$	only BI	70.8	+20
$BI_6(2)$	BI+FC	71.5	+20
$BI_6(6)$	BI+FC	72.9	+45
$BI_7(6)$	BI+FC	73.1	+50
$BI_8(10)$	BI+FC	72.0	+30
$BI_6(2)$-$BI_7(6)$	BI+FC	73.6	+35
$BI_7(6)$-$BI_8(10)$	BI+FC	73.4	+55
$BI_6(2)$-$BI_7(6)$	FULL	**74.1**	+35
$BI_6(2)$-$BI_7(6)$-CRF	FULL	**75.1**	+865
DeepLab-CRF [3]		72.7	+830
DeepLab-MSc-CRF [3]		**73.6**	+880
DeepLab-EdgeNet [6]		71.7	+30
DeepLab-EdgeNet-CRF [6]		**73.6**	+860

Table 1. Semantic Segmentation using DeepLab model. IoU scores on Pascal VOC12 segmentation test dataset and average runtimes (ms) corresponding to different models. Also shown are the results corresponding to competitive dense pixel prediction techniques that used the same base DeepLab CNN. Runtimes also include superpixel computation (6ms). In the second column, 'BI', 'FC' and 'FULL' correspond to training 'BI', 'FC' and full model layers respectively.

We add inception modules after different FC layers in the original model and remove the DenseCRF post processing. For this dataset, we use 1000 SLIC superpixels [35,37]. The inception modules after FC_6, FC_7 and FC_8 layers are referred to as $BI_6(H)$, $BI_7(H)$ and $BI_8(H)$ respectively, where H is the number of kernels. All results using the DeepLab model on Pascal VOC12 dataset are summarized in Tab. 1. We report the 'test' numbers without validation numbers, because the released DeepLab model that we adapted was trained using both train and validation sets. The DeepLab network achieves an IoU of 68.9 after bilinear interpolation. Experiments with $BI_6(2)$ module indicate that even only learning the inception module while keeping the remaining network fixed results in an reliable IoU improvement (+1.9). Additional joint training with FC layers significantly improved the performance. The results also show that more kernels improve performance. Next, we add multiple modules to the base DeepLab network at various stages and train them jointly. This results in further improvement of the performance. The $BI_6(2)$-$BI_7(6)$ model with two inception modules shows significant improvement in IoU by 4.7 and 0.9 in comparison to baseline model and DenseCRF application respectively. Finally, finetuning the entire network (FULL in Tab. 1) boosts the performance by 5.2 and 1.4 compared to the baseline and DenseCRF application.

Some visual results are shown in Fig. 5 and more are included in the supplementary. Several other variants of using BI are conceivable. During our experiments, we have observed that more kernels and more modules improve the performance, so we expect that even better results can be achieved. In Tab. 1, the runtime (ms) is included for several models. These numbers have been obtained using a Nvidia Tesla K80 GPU and standard Caffe time benchmarking [38]. DenseCRF timings are taken from [6]. The runtimes indicate that the overhead with BI modules is quite minimal in comparison to using Dense CRF.

In addition, we include the results of some other dense pixel prediction methods that are build on top of the same DeepLab base model. DeepLab-MSc-CRF is a multi-scale version [3] of DeepLab with DenseCRF on top. DeepLab-EdgeNet [6] is a recently proposed fast and discriminatively trained domain transform technique for propagating information across pixels. Comparison with these techniques in terms of performance and runtime indicates that our approach performs on par with latest dense pixel prediction techniques with significantly less time overhead. Several state-of-the-art CNN based systems [15,16] have achieved higher results than DeepLab on Pascal VOC12. These models are not yet publicly available and so we could not test the use of BI models in them. A close variant [26] of our work, which propose to do optimization in the bilateral space also has fast runtimes, but reported lower performance in comparison to the application of DenseCRF.

CRFasRNN As a second architecture, we modified the CNN architecture trained by [5] that produces a result at an even lower resolution ($\frac{1}{16}\times$). Multiple deconvolution steps are employed to obtain the segmentation at input image resolution. This result is then passed onto the DenseCRF recurrent neural network to obtain the final segmentation result. We insert BI modules after score-pool3, score-pool4, FC_6 and FC_7 layers, please see [18,5] for the network architecture details. Instead of combining outputs from the above layers with deconvolution steps, we introduce BI modules after them and linearly combined the outputs to obtain final segmentation result. Note that we entirely removed both the deconvolution and the DenseCRF parts of the original model [5]. See Tab. 2 for results on the DeconvNet model. Without the DenseCRF part and only evaluating the deconvolutional part of this model, one obtains an IoU score of 72.0. Ten steps of mean field inference increase the IoU to 74.7 [5]. Our model, with few additional parameters compared to the base CNN, achieves a IoU performance of 74.9, showing an improvement of 0.2 over the CRFasRNN model. The BI layers lead to better performance than deconvolution and DenseCRF combined while being much faster.

Model	*IoU*	*Runtime*
DeconvNet(CNN+Deconv.)	72.0	190ms
With BI modules $BI_3(2)$-$BI_4(2)$-$BI_6(2)$-$BI_7(2)$	**74.9**	245
CRFasRNN (DeconvNet-CRF)	74.7	2700

Table 2. Semantic Segmentation using CRFasRNN model. IoU scores and runtimes corresponding to different models on Pascal VOC12 test dataset. Note that runtime also includes superpixel computation.

Hierarchical Clustering Analysis We learned the network parameters using 1000 gSLIC superpixels per image, however the inception module allows to change the resolution (a non-square K). To illustrate this, we perform agglomorative clustering of the superpixels, sequentially merging the nearest two superpixels into a single one. We then evaluated the DeepLab-$BI_6(2)$-$BI_7(6)$ network using different levels of the resulting hierarchy re-using all the trained network parameters. Results in Fig. 6 show that the IoU score on the validation set decreases slowly with decreasing number of points and then drops for less than 200 superpixels. This validates that the network generalizes to different superpixel layouts and it is sufficient to represent larger regions of similar color

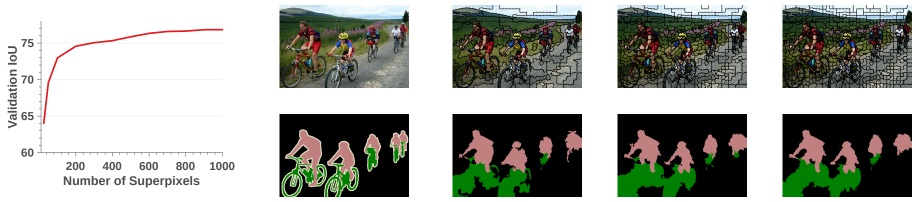

Fig. 6. Hierarchical Clustering Analysis. From left to right: Validation performance when using different super-pixel layouts, visualization of an image with ground truth segmentation, and the $BI_6(2)$-$BI_7(6)$ result with 200, 600, and 1000 superpixels.

by fewer points. In future, we plan to explore different strategies to allocate the representation to those regions that require more resolution and to remove the superpixelization altogether. Fig. 6 shows example image with 200, 600, and 1000 superpixels and their obtained segmentation with BI modules.

4.2 Material Segmentation

Model	Class / Total accuracy	Runtime
Alexnet CNN	55.3 / 58.9	300ms
$BI_7(2)$-$BI_8(6)$	67.7 / 71.3	410
$BI_7(6)$-$BI_8(6)$	**69.4 / 72.8**	470
AlexNet-CRF	65.5 / 71.0	3400

Table 3. Material Segmentation using AlexNet. Pixel accuracies and runtimes (in ms) of different models on MINC material segmentation dataset [4]. Runtimes also include the time for superpixel extraction (15ms).

We also experiment on a different pixel prediction task of material segmentation by adapting a CNN architecture fine-tuned for Materials in Context (MINC) [4] dataset. MINC consists of 23 material classes and is available in three different resolutions with the same aspect ratio: low (550^2), mid (1100^2) and an original higher resolution. The authors of [4] train CNNs on the mid resolution images and then combine with a DenseCRF to predict and evaluate on low resolution images. We build our work based on the Alexnet model [46] released by the authors of [4]. To obtain a per pixel labeling of a given image, there are several processing steps that [4] use for good performance. First, a CNN is applied at several scales with different strides followed by an interpolation of the predictions to reach the input image resolution and is then followed by a DenseCRF. For simplicity, we choose to run the CNN network with single scale and no-sliding. The authors used just one kernel with (u, v, L, a, b) features in the DenseCRF part. We used the same features in our inception modules. We modified the base AlexNet model by inserting BI modules after FC_7 and FC_8 layers. Again, 1000 SLIC superpixels are used for all experiments. Results on the test set are shown in Table 3. When inserting BI modules, the performance improves both in total pixel accuracy as well as in class-averaged accuracy. We observe an improvement of 12% compared to CNN predictions and $2-4\%$ compared to CNN+DenseCRF results. Qualitative examples are shown in Fig. 7 and

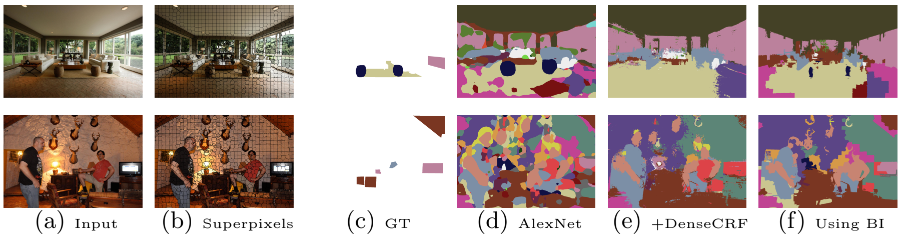

Fig. 7. Material Segmentation. Example results of material segmentation. (d) depicts the AlexNet CNN result, (e) CNN + 10 steps of mean-field inference, (f) results obtained with bilateral inception (BI) modules ($BI_7(2)$+$BI_8(6)$) between FC layers.

more are included in the supplementary. The weights to combine outputs in the BI layers are found by validation on the validation set. For this model we do not provide any learned setup due very limited segment training data.

4.3 Street Scene Segmentation

We further evaluate the use of BI modules on the Cityscapes dataset [43]. Cityscapes contains 20K high-resolution (1024×2048) images of street scenes with coarse pixel annotations and another 5K images with fine annotations, all annotations are from 19 semantic classes. The 5K images are divided into 2975 train, 500 validation and remaining test images. Since there are no publicly available pre-trained models for this dataset yet, we trained a DeepLab model. We trained the base DeepLab model with half resolution images (512×1024) so that the model fits into GPU memory. The result is then interpolated to full-resolution using bilinear interpolation.

Model	IoU (Half-res.)	IoU (Full-res.)	Runtime
DeepLab CNN	62.2	65.7	0.3s
$BI_6(2)$	62.7	66.5	5.7
$BI_6(2)$-$BI_7(6)$	**63.1**	**66.9**	6.1
DeepLab-CRF	63.0	66.6	6.9

Table 4. Street Scene Segmentation using DeepLab model. IoU scores and runtimes (in sec) of different models on Cityscapes segmentation dataset [43], for both half-resolution and full-resolution images. Runtime computations also include superpixel computation time (5.2s).

We experimented with two layouts: only a single $BI_6(2)$ and one with two inception $BI_6(2)$-$BI_7(6)$ modules. We notice that the SLIC superpixels [35] give higher quantization error than on VOC and thus used 6000 superpixels using [36] for our experiments. Quantitative results on the validation set are shown in Tab. 4. In contrast to the findings on the previous datasets, we only observe modest improvements with both DenseCRF and our inception modules in comparison to the base model. Similar to the previous experiments, the inception modules achieve better performance than DenseCRF while being faster. The majority of the computation time in our approach is due to the extraction of

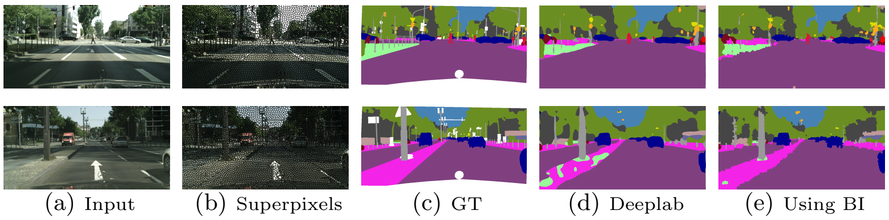

Fig. 8. Street Scene Segmentation. Example results of street scene segmentation. (d) depicts the DeepLab results, (e) result obtained by adding bilateral inception (BI) modules ($BI_6(2)$+$BI_7(6)$) between FC layers. More in supplementary.

superpixels ($5.2s$) using a CPU implementation. Some visual results with $BI_6(2)$-$BI_7(6)$ model are shown in Fig. 8 with more in supplementary.

5 Conclusion

The DenseCRF [2] with mean field inference has been used in many CNN segmentation approaches. Its main ingredient and reason for the improved performance is the use of a bilateral filter applied to the beliefs over labels. We have introduced a CNN approach that uses this key component in a novel way: filtering intermediate representations of higher levels in CNNs while jointly learning the task-specific feature spaces. This propagates information between earlier and more detailed intermediate representations of the classes instead of beliefs over labels. Further we show that image adaptive layouts in the higher levels of CNNs can be used to an advantage in the same spirit as CRF graphs have been constructed using superpixels in previous works on semantic segmentation. The computations in the 1×1 convolution layers scales in the number of superpixels which may be an advantage. Further we have shown that the same representation can be used to interpolate the coarser representations to the full image.

The use of image-adaptive convolutions in between the FC layers retains the appealing effect of producing segmentation masks with sharp edges. This is not a property of the superpixels, using them to represent information in FC layers and their use to interpolate to the full resolution are orthogonal. Different interpolation steps can be used to propagate the label information to the entire image, including bilinear interpolation, up-convolutions and DenseCRFs. We plan to investigate the effect of different sampling strategies to represent information in the higher layers of CNNs and apply similar image-adaptive ideas to videos.

We believe that the Bilateral Inception models are an interesting step that aims to directly include the model structure of CRF factors into the forward architecture of CNNs. The BI modules are easy to implement and are applicable to CNNs that perform structured output prediction.

Acknowledgements We thank the reviewers for their valuable feedback. Raghudeep Gadde is supported by CSTB and ANR-13-CORD-0003.

References

1. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the 18th International Conference on Machine Learning. (2001) 282–289
2. Krähenbühl, P., Koltun, V.: Efficient inference in fully connected CRFs with Gaussian edge potentials. In: Advances in Neural Information Processing Systems. (2011)
3. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Semantic image segmentation with deep convolutional nets and fully connected CRFs. In: International Conference on Learning Representation. (2015)
4. Bell, S., Upchurch, P., Snavely, N., Bala, K.: Material recognition in the wild with the materials in context database. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 3479–3487
5. Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P.H.: Conditional random fields as recurrent neural networks. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 1529–1537
6. Chen, L.C., Barron, J.T., Papandreou, G., Murphy, K., Yuille, A.L.: Semantic image segmentation with task-specific edge detection using CNNs and a discriminatively trained domain transform. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 4545–4554
7. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 1–9
8. Lin, M., Chen, Q., Yan, S.: Network In Network. In: International Conference on Learning Representation. (2014)
9. Domke, J.: Learning graphical model parameters with approximate marginal inference. Pattern Analysis and Machine Intelligence, IEEE Transactions on **35**(10) (2013) 2454–2467
10. Jampani, V., Kiefel, M., Gehler, P.V.: Learning sparse high dimensional filters: Image filtering, dense CRFs and bilateral neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 4452–4461
11. Li, Y., Zemel, R.: Mean-field networks. ICML workshop on Learning Tractable Probabilistic Models (2014)
12. Schwing, A.G., Urtasun, R.: Fully connected deep structured networks. arXiv preprint arXiv:1503.02351 (2015)
13. Kiefel, M., Gehler, P.V.: Human pose estimation with fields of parts. In: European Conference on Computer Vision, Springer (2014) 331–346
14. Chen, L.c., Schwing, A., Yuille, A., Urtasun, R.: Learning deep structured models. In: Proceedings of the 32nd International Conference on Machine Learning. (2015) 1785–1794
15. Lin, G., Shen, C., Reid, I., et al.: Efficient piecewise training of deep structured models for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 3194–3203
16. Liu, Z., Li, X., Luo, P., Loy, C.C., Tang, X.: Semantic image segmentation via deep parsing network. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 1377–1385
17. Zeiler, M.D., Krishnan, D., Taylor, G.W., Fergus, R.: Deconvolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2010) 2528–2535

18. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 3431–3440
19. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. In: European Conference on Computer Vision, Springer (2014) 346–361
20. Ionescu, C., Vantzos, O., Sminchisescu, C.: Matrix backpropagation for deep networks with structured layers. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 2965–2973
21. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. In: International Conference on Learning Representation. (2016)
22. He, S., Lau, R.W., Liu, W., Huang, Z., Yang, Q.: Supercnn: A superpixelwise convolutional neural network for salient object detection. International Journal of Computer Vision (2015) 1–15
23. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. In: International Conference on Learning Representation. (2014)
24. Tenenbaum, J.B., De Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science **290**(5500) (2000) 2319–2323
25. Mostajabi, M., Yadollahpour, P., Shakhnarovich, G.: Feedforward semantic segmentation with zoom-out features. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 3376–3385
26. Barron, J.T., Poole, B.: The fast bilateral solver. In: European Conference on Computer Vision, Springer (2016)
27. Barron, J.T., Adams, A., Shih, Y., Hernández, C.: Fast bilateral-space stereo for synthetic defocus. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 4466–4474
28. Kiefel, M., Jampani, V., Gehler, P.: Permutohedral lattice CNNs. In: International Conference on Learning Representation Workshops. (2015)
29. Adams, A., Baek, J., Davis, M.A.: Fast high-dimensional filtering using the permutohedral lattice. In: Computer Graphics Forum. Volume 29., Wiley Online Library (2010) 753–762
30. Campbell, N., Subr, K., Kautz, J.: Fully-connected crfs with non-parametric pairwise potential. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2013) 1658–1665
31. Gould, S., Zhao, J., He, X., Zhang, Y.: Superpixel graph label transfer with learned distance metric. In: European Conference on Computer Vision, Springer (2014) 632–647
32. Gonfaus, J.M., Boix, X., Van de Weijer, J., Bagdanov, A.D., Serrat, J., Gonzalez, J.: Harmony potentials for joint classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2010) 3280–3287
33. Nowozin, S., Gehler, P.V., Lampert, C.H.: On parameter learning in CRF-based approaches to object class image segmentation. In: European Conference on Computer Vision, Springer (2010) 98–111
34. Everingham, M., Gool, L.V., Williams, C., Winn, J., Zisserman, A.: The PASCAL VOC2012 challenge results. (2012)
35. Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Susstrunk, S.: Slic superpixels compared to state-of-the-art superpixel methods. Pattern Analysis and Machine Intelligence, IEEE Transactions on **34**(11) (2012) 2274–2282

36. Dollár, P., Zitnick, C.L.: Structured forests for fast edge detection. In: Proceedings of the IEEE International Conference on Computer Vision. (2013) 1841–1848
37. Ren, C.Y., Prisacariu, V.A., Reid, I.D.: gslicr: Slic superpixels at over 250hz. ArXiv e-prints (1509.04232) (2015)
38. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. In: Proceedings of the ACM International Conference on Multimedia, ACM (2014) 675–678
39. Paris, S., Durand, F.: A fast approximation of the bilateral filter using a signal processing approach. In: European conference on computer vision, Springer (2006) 568–580
40. Gastal, E.S., Oliveira, M.M.: Domain transform for edge-aware image and video processing. In: ACM Transactions on Graphics (TOG). Volume 30., ACM (2011) 69
41. Adams, A., Gelfand, N., Dolson, J., Levoy, M.: Gaussian kd-trees for fast high-dimensional filtering. In: ACM Transactions on Graphics (TOG). Volume 28., ACM (2009) 21
42. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representation. (2015)
43. Cordts, M., Omran, M., Ramos, S., Scharwächter, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset. In: CVPR Workshop on The Future of Datasets in Vision. (2015)
44. Hariharan, B., Arbeláez, P., Bourdev, L., Maji, S., Malik, J.: Semantic contours from inverse detectors. In: Proceedings of the IEEE International Conference on Computer Vision. (2011) 991–998
45. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European Conference on Computer Vision, Springer (2014) 740–755
46. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems. (2012) 1097–1105