Semantic Image Segmentation with Task-Specific Edge Detection Using CNNs and a Discriminatively Trained Domain Transform

Liang-Chieh Chen* lcchen@cs.ucla.edu
Jonathan T. Barron, George Papandreou, Kevin Murphy {barron, gpapan, kpmurphy}@google.com
Alan L. Yuille yuille@stat.ucla.edu alan.yuille@jhu.edu

Abstract

Deep convolutional neural networks (CNNs) are the backbone of state-of-art semantic image segmentation systems. Recent work has shown that complementing CNNs with fully-connected conditional random fields (CRFs) can significantly enhance their object localization accuracy, yet dense CRF inference is computationally expensive. We propose replacing the fully-connected CRF with domain transform (DT), a modern edge-preserving filtering method in which the amount of smoothing is controlled by a reference edge map. Domain transform filtering is several times faster than dense CRF inference and we show that it yields comparable semantic segmentation results, accurately capturing object boundaries. Importantly, our formulation allows learning the reference edge map from intermediate CNN features instead of using the image gradient magnitude as in standard DT filtering. This produces task-specific edges in an end-to-end trainable system optimizing the target semantic segmentation quality.

1. Introduction

Deep convolutional neural networks (CNNs) are very effective in semantic image segmentation, the task of assigning a semantic label to every pixel in an image. Recently, it has been demonstrated that post-processing the output of a CNN with a fully-connected CRF can significantly increase segmentation accuracy near object boundaries [5].

As explained in [26], mean-field inference in the fully-connected CRF model amounts to iterated application of the bilateral filter, a popular technique for edge-aware filtering. This encourages pixels which are nearby in position and in color to be assigned the same semantic label. In practice, this produces semantic segmentation results which are well aligned with object boundaries in the image.

One key impediment in adopting the fully-connected CRF is the rather high computational cost of the underlying bilateral filtering step. Bilateral filtering amounts to high-dimensional Gaussian filtering in the 5-D bilateral (2-D position, 3-D color) space and is expensive in terms of both memory and CPU time, even when advanced algorithmic techniques are used.

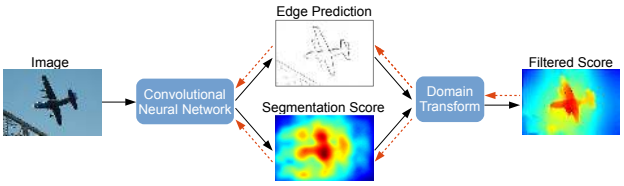


Figure 1. A single unified CNN produces both coarse semantic segmentation scores and an edge map, which respectively serve as input multi-channel image and reference edge to a domain transform edge-preserving filter. The resulting filtered semantic segmentation scores are well-aligned with the object boundaries. The full architecture is discriminatively trained by backpropagation (red dashed arrows) to optimize the target semantic segmentation.

In this paper, we propose replacing the fully-connected CRF and its associated bilateral filtering with the domain transform (DT) [16], an alternative edge-aware filter. The recursive formulation of the domain transform amounts to adaptive recursive filtering of a signal, where information is not allowed to propagate across edges in some reference signal. This results in an extremely efficient scheme which is an order of magnitude faster than the fastest algorithms for a bilateral filter of equivalent quality.

The domain transform can equivalently be seen as a recurrent neural network (RNN). In particular, we show that the domain transform is a special case of the recently proposed RNN with gated recurrent units. This connection allows us to share insights, better understanding two seemingly different methods, as we explain in Section 3.4.

The amount of smoothing in a DT is spatially modulated by a reference edge map, which in the standard DT corresponds to image gradient magnitude. Instead, we will learn the reference edge map from intermediate layer features of the same CNN that produces the semantic segmentation scores, as illustrated in Fig. 1. Crucially, this allows us to learn a task-specific edge detector tuned for semantic image

*Work done in part during an internship at Google Inc.

segmentation in an end-to-end trainable system.

We evaluate the performance of the proposed method on the challenging PASCAL VOC 2012 semantic segmentation task. In this task, domain transform filtering is several times faster than dense CRF inference, while performing almost as well in terms of the mean intersection-over-union (mIOU) metric. In addition, although we only trained for semantic segmentation, the learned edge map performs competitively on the BSDS500 edge detection benchmark.

2. Related Work

Semantic image segmentation Deep Convolutional Neural Networks (CNNs) [27] have demonstrated excellent performance on the task of semantic image segmentation [10, 28, 30]. However, due to the employment of max-pooling layers and downsampling, the output of these networks tend to have poorly localized object boundaries. Several approaches have been adopted to handle this problem. [31, 19, 5] proposed to extract features from the intermediate layers of a deep network to better estimate the object boundaries. Networks employing deconvolutional layers and unpooling layers to recover the "spatial invariance" effect of max-pooling layers have been proposed by [45, 33]. [14, 32] used super-pixel representation, which essentially appeals to low-level segmentation methods for the task of localization. The fully connected Conditional Random Field (CRF) [26] has been applied to capture long range dependencies between pixels in [5, 28, 30, 34]. Further improvements have been shown in [46, 38] when backpropagating through the CRF to refine the segmentation CNN. In contrary, we adopt another approach based on the domain transform [16] and show that beyond refining the segmentation CNN, we can also jointly learn to detect object boundaries, embedding task-specific edge detection into the proposed model.

Edge detection The edge/contour detection task has a long history [25, 1, 11], which we will only briefly review. Recently, several works have achieved outstanding performance on the edge detection task by employing CNNs [2, 3, 15, 21, 39, 44]. Our work is most related to the ones by [44, 3, 24]. While Xie and Tu [44] also exploited features from the intermediate layers of a deep network [40] for edge detection, they did not apply the learned edges for high-level tasks, such as semantic image segmentation. On the other hand, Bertasius *et al*. [3] and Kokkinos [24] made use of their learned boundaries to improve the performance of semantic image segmentation. However, the boundary detection and semantic image segmentation are considered as two separate tasks. They optimized the performance of boundary detection instead of the performance of high level tasks. On the contrary, we learn object boundaries in order to directly optimize the performance of semantic image segmentation.

Long range dependency Recurrent neural networks (RNNs) [12] with long short-term memory (LSTM) units [20] or gated recurrent units (GRUs) [8, 9] have proven successful to model the long term dependencies in sequential data (*e.g*., text and speech). Sainath *et al*. [37] have combined CNNs and RNNs into one unified architecture for speech recognition. Some recent work has attempted to model *spatial* long range dependency with recurrent networks for computer vision tasks [17, 41, 35, 4, 43]. Our work, integrating CNNs and Domain Transform (DT) with recursive filtering [16], bears a similarity to ReNet [43], which also performs recursive operations both horizontally and vertically to capture long range dependency within whole image. In this work, we show the relationship between DT and GRU, and we also demonstrate the effectiveness of exploiting long range dependency by DT for semantic image segmentation. While [42] has previously employed the DT (for joint object-stereo labeling), we propose to backpropagate through both of the DT inputs to jointly learn segmentation scores and edge maps in an end-to-end trainable system. We show that these learned edge maps bring significant improvements compared to standard image gradient magnitude used by [42] or earlier DT literature [16].

3. Proposed Model

3.1. Model overview

Our proposed model consists of three components, illustrated in Fig. 2. They are jointly trained end-to-end to optimize the output semantic segmentation quality.

The first component that produces coarse semantic segmentation score predictions is based on the publicly available DeepLab model, [5], which modifies VGG-16 net [40] to be FCN [31]. The model is initialized from the VGG-16 ImageNet [36] pretrained model. We employ the DeepLab-LargeFOV variant of [5], which introduces zeros into the filters to enlarge its Field-Of-View, which we will simply denote by DeepLab in the sequel.

We add a second component, which we refer to as EdgeNet. The EdgeNet predicts edges by exploiting features from intermediate layers of DeepLab. The features are resized to have the same spatial resolution by bilinear interpolation before concatenation. A convolutional layer with kernel size 1×1 and one output channel is applied to yield edge prediction. ReLU is used so that the edge prediction is in the range of zero to infinity.

The third component in our system is the domain transform (DT), which is is an edge-preserving filter that lends itself to very efficient implementation by separable 1-D recursive filtering across rows and columns. Though DT is traditionally used for graphics applications [16], we use it to filter the raw CNN semantic segmentation scores to be better aligned with object boundaries, guided by the EdgeNet produced edge map.

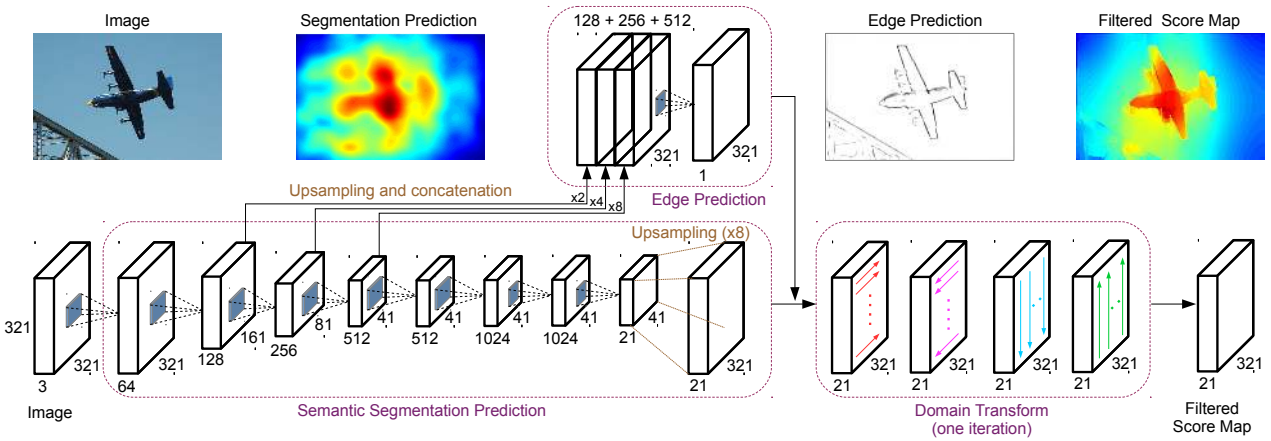


Figure 2. Our proposed model has three components: (1) DeepLab for semantic segmentation prediction, (2) EdgeNet for edge prediction, and (3) Domain Transform to accurately align segmentation scores with object boundaries. EdgeNet reuses features from intermediate DeepLab layers, resized and concatenated before edge prediction. Domain transform takes as input the raw segmentation scores and edge map, and recursively filters across rows and columns to produce the final filtered segmentation scores.

We review the standard DT in Sec. 3.2, we extend it to a fully trainable system with learned edge detection in Sec. 3.3, and we discuss connections with the recently proposed gated recurrent unit networks in Sec. 3.4.

3.2. Domain transform with recursive filtering

The domain transform takes two inputs: (1) The raw input signal $\boldsymbol{x}$ to be filtered, which in our case corresponds to the coarse DCNN semantic segmentation scores, and (2) a positive "domain transform density" signal $\boldsymbol{d}$, whose choice we discuss in detail in the following section. The output of the DT is a filtered signal $\boldsymbol{y}$. We will use the recursive formulation of the DT due to its speed and efficiency, though the filter can be applied via other techniques [16].

For 1-D signals of length N, the output is computed by setting $y_1 = x_1$ and then recursively for $i = 2, \ldots, N$

$$y_i = (1 - w_i)x_i + w_i y_{i-1}\,. \tag{1}$$

The weight w_i depends on the domain transform density d_i

$$w_i = \exp\left(-\sqrt{2}d_i/\sigma_s\right)\,, \tag{2}$$

where σ_s is the standard deviation of the filter kernel over the input's spatial domain.

Intuitively, the strength of the domain transform density $d_i \geq 0$ determines the amount of diffusion/smoothing by controlling the relative contribution of the raw input signal x_i to the filtered signal value at the previous position y_{i-1} when computing the filtered signal at the current position y_i. The value of $w_i \in (0, 1)$ acts like a gate, which controls how much information is propagated from pixel $i-1$ to i. We have full diffusion when d_i is very small, resulting into $w_i = 1$ and $y_i = y_{i-1}$. On the other extreme, if d_i is very large, then $w_i = 0$ and diffusion stops, resulting in $y_i = x_i$.

Filtering by Eq. (1) is asymmetric, since the current output only depends on previous outputs. To overcome this asymmetry, we filter 1-D signals twice, first left-to-right, then right-to-left on the output of the left-to-right pass.

Domain transform filtering for 2-D signals works in a separable fashion, employing 1-D filtering sequentially along each signal dimension. That is, a horizontal pass (left-to-right and right-to-left) is performed along each row, followed by a vertical pass (top-to-bottom and bottom-to-top) along each column. In practice, $K > 1$ iterations of the two-pass 1-D filtering process can suppress "striping" artifacts resulting from 1-D filtering on 2-D signals [16, Fig. 4]. We reduce the standard deviation of the DT filtering kernel at each iteration, requiring that the sum of total variances equals the desired variance σ_s^2, following [16, Eq. 14]

$$\sigma_k = \sigma_s \sqrt{3} \frac{2^{K-k}}{\sqrt{4^K - 1}}, \quad k = 1, \ldots, K\,, \tag{3}$$

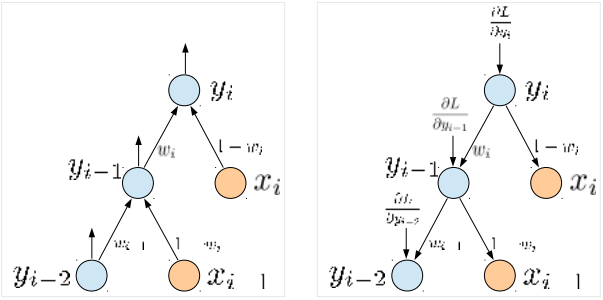
plugging σ_k in place of σ_s to compute the weights w_i by Eq. (2) at the k-th iteration.

The domain transform density values d_i are defined as

$$d_i = 1 + g_i \frac{\sigma_s}{\sigma_r}\,, \tag{4}$$

where $g_i \geq 0$ is the "reference edge", and σ_r is the standard deviation of the filter kernel over the reference edge map's range. Note that the larger the value of g_i is, the more confident the model thinks there is a strong edge at pixel i, thus inhibiting diffusion (*i.e.*, $d_i \to \infty$ and $w_i = 0$). The standard DT [16] usually employs the color image gradient

$$g_i = \sum_{c=1}^{3} \|\nabla I_i^{(c)}\| \tag{5}$$



(a) (b)

Figure 3. Computation tree for domain transform recursive filtering: (a) Forward pass. Upward arrows from y_i nodes denote feeds to subsequent layers. (b) Backward pass, including contributions $\frac{\partial L}{\partial y_i}$ from subsequent layers.

but we show next that better results can be obtained by computing the reference edge map by a learned DCNN.

3.3. Trainable domain transform filtering

One novel aspect of our proposed approach is to back-propagate the segmentation errors at the DT output $\boldsymbol{y}$ through the DT onto its two inputs. This allows us to use the DT as a layer in a CNN, thereby allowing us to jointly learn DCNNs that compute the coarse segmentation score maps in $\boldsymbol{x}$ and the reference edge map in $\boldsymbol{g}$.

We demonstrate how DT backpropagation works for the 1-D filtering process of Eq. (1), whose forward pass is illustrated as computation tree in Fig. 3(a). We assume that each node y_i not only influences the following node y_{i+1} but also feeds a subsequent layer, thus also receiving gradient contributions $\frac{\partial L}{\partial y_i}$ from that layer during back-propagation. Similar to standard back-propagation in time, we unroll the recursion of Eq. (1) in reverse for $i = N, \dots, 2$ as illustrated in Fig. 3(b) to update the derivatives with respect to $\boldsymbol{y}$, and to also compute derivatives with respect to $\boldsymbol{x}$ and $\boldsymbol{w}$,

$$\frac{\partial L}{\partial x_i} \leftarrow (1 - w_i) \frac{\partial L}{\partial y_i} \tag{6}$$

$$\frac{\partial L}{\partial w_i} \leftarrow \frac{\partial L}{\partial w_i} + (y_{i-1} - x_i) \frac{\partial L}{\partial y_i} \tag{7}$$

$$\frac{\partial L}{\partial y_{i-1}} \leftarrow \frac{\partial L}{\partial y_{i-1}} + w_i \frac{\partial L}{\partial y_i}, \tag{8}$$

where $\frac{\partial L}{\partial x_i}$ and $\frac{\partial L}{\partial w_i}$ are initialized to 0 and $\frac{\partial L}{\partial y_i}$ is initially set to the value sent by the subsequent layer. Note that the weight w_i is shared across all filtering stages (*i.e.*, left-to-right/right-to-left within horizontal pass and top-to-bottom/bottom-to-top within vertical pass) and K iterations, with each pass contributing to the partial derivative.

With these partial derivatives we can produce derivatives with respect to the reference edge g_i. Plugging Eq. (4) into Eq. (2) yields

$$w_i = \exp\left(-\frac{\sqrt{2}}{\sigma_k}\left(1 + g_i \frac{\sigma_s}{\sigma_r}\right)\right). \tag{9}$$

Then, by the chain rule, the derivative with respect to g_i is

$$\frac{\partial L}{\partial g_i} = -\frac{\sqrt{2}}{\sigma_k} \frac{\sigma_s}{\sigma_r} w_i \frac{\partial L}{\partial w_i}. \tag{10}$$

This gradient is then further propagated onto the deep convolutional neural network that generated the edge predictions that were used as input to the DT.

3.4. Relation to gated recurrent unit networks

Equation 1 defines DT filtering as a recursive operation. It is interesting to draw connections with other recent RNN formulations. Here we establish a precise connection with the gated recurrent unit (GRU) RNN architecture [8] recently proposed for modeling sequential text data. The GRU employs the update rule

$$y_i = z_i \tilde{y}_i + (1 - z_i) y_{i-1}. \tag{11}$$

Comparing with Eq. (1), we can relate the GRU's "update gate" z_i and "candidate activation" $\tilde{y}_i$ with DT's weight and raw input signal as follows: $z_i = 1 - w_i$ and $\tilde{y}_i = x_i$.

The GRU update gate z_i is defined as $z_i = \sigma(f_i)$, where f_i is an activation signal and $\sigma(t) = 1/(1+e^{-t})$. Comparing with Eq. (9) yields a direct correspondence between the DT reference edge map g_i and the GRU activation f_i:

$$g_i = \frac{\sigma_r}{\sigma_s}\left(\frac{\sigma_k}{\sqrt{2}} \log(1 + e^{f_i}) - 1\right). \tag{12}$$

4. Experimental Evaluation

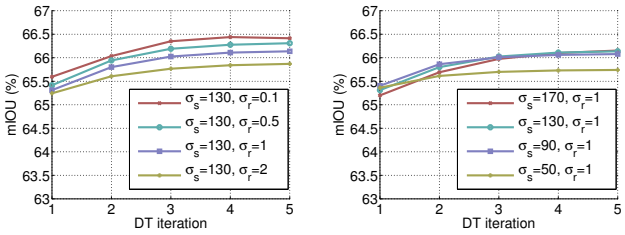
4.1. Experimental Protocol

Dataset We evaluate the proposed method on the PASCAL VOC 2012 segmentation benchmark [13], consisting of 20 foreground object classes and one background class. We augment the training set from the annotations by [18]. The performance is measured in terms of pixel intersection-over-union (IOU) averaged across the 21 classes.

Training A two-step training process is employed. We first train the DeepLab component and then we jointly fine-tune the whole model. Specifically, we employ exactly the same setting as [5] to train DeepLab in the first stage. In the second stage, we employ a small learning rate of 10^{-8} for fine-tuning. The added convolutional layer of EdgeNet is initialized with Gaussian variables with zero mean and standard deviation of 10^{-5} so that in the beginning the EdgeNet predicts no edges and it starts to gradually learn edges for semantic segmentation. Total training time is 11.5 hours (10.5 and 1 hours for each stage).

Method	mIOU (%)
Baseline: DeepLab	62.25
conv3_3	65.64
conv2_2 + conv3_3	65.75
conv2_2 + conv3_3 + conv4_3	66.03
conv2_2 + conv3_3 + conv4_3 + conv5_3	65.94
conv1_2 + conv2_2 + conv3_3 + conv4_3	65.89

Table 1. VOC 2012 val set. Effect of using features from different convolutinal layers for EdgeNet ($\sigma_s = 100$ and $\sigma_r = 1$ for DT).



(a) (b)

Figure 4. VOC 2012 val set. Effect of varying number of iterations for domain transform: (a) Fix σ_s and vary both σ_r and K iterations. (b) Fix σ_r and vary both σ_s and K iterations.

Reproducibility The proposed methods are implemented by extending the Caffe framework [22]. The code and models are available at `http://liangchiehchen.com/projects/DeepLab.html`.

4.2. Experimental Results

We first explore on the validation set the hyper-parameters in the proposed model, including (1) features for EdgeNet, (2) hyper-parameters for domain transform (*i.e.*, number of iterations, σ_s, and σ_r). We also experiment with different methods to generate edge prediction. After that, we analyze our models and evaluate on the official test set.

Features for EdgeNet The EdgeNet we employ exploits intermediate features from DeepLab. We first investigate which VGG-16 [40] layers give better performance with the DT hyper-parameters fixed. As shown in Tab. 1, baseline DeepLab attains 62.25% mIOU on PASCAL VOC 2012 validation set. We start to exploit the features from conv3_3, which has receptive field size 40. The size is similar to the patch size typically used for edge detection [11]. The resulting model achieves performance of 65.64%, 3.4% better than the baseline. When using features from conv2_2, conv3_3, and conv4_3, the performance can be further improved to 66.03%. However, we do not observe any significant improvement if we also exploit the features from conv1_2 or conv5_3. We use features from conv2_2, conv3_3, and conv4_3 in remaining experiments involving EdgeNet.

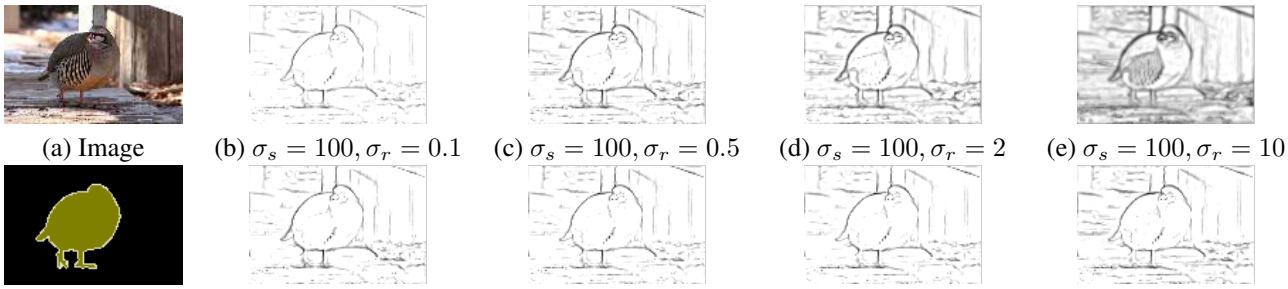
Number of domain transform iterations Domain transform requires multiple iterations of the two-pass 1-D filtering process to avoid the "striping" effect [16, Fig. 4]. We train the proposed model with K iterations for the domain transform, and perform the same K iterations during test. Since there are two more hyper-parameters σ_s and σ_r (see Eq. (9)), we also vary their values to investigate the effect of varying the K iterations for domain transform. As shown in Fig. 4, employing $K = 3$ iterations for domain transform in our proposed model is sufficient to reap most of the gains for several different values of σ_s and σ_r.

Varying domain transform σ_s, σ_r and comparison with other edge detectors We investigate the effect of varying σ_s and σ_r for domain transform. We also compare alternative methods to generate edge prediction for domain transform: (1) DT-Oracle, where groundtruth object boundaries are used, which serves as an upper bound on our method. (2) The proposed DT-EdgeNet, where the edges are produced by EdgeNet. (3) DT-SE, where the edges are found by Structured Edges (SE) [11]. (4) DT-Gradient, where the image (color) gradient magnitude of Eq. (5) is used as in standard domain transform [16]. We search for optimal σ_s and σ_r for those methods. First, we fix $\sigma_s = 100$ and vary σ_r in Fig. 5(a). We found that the performance of DT-Oracle, DT-SE, and DT-Gradient are affected a lot by different values of σ_r, since they are generated by other "plugged-in" modules (*i.e.*, not jointly fine-tuned). We also show the performance of baseline DeepLab and DeepLab-CRF which employs dense CRF. We then fix the found optimal value of σ_r and vary σ_s in Fig. 5 (b). We found that as long as $\sigma_s \geq 90$, the performance of DT-EdgeNet, DT-SE, and DT-Gradient do not vary significantly. After finding optimal values of σ_r and σ_s for each setting, we use them for remaining experiments.

We further visualize the edges learned by our DT-EdgeNet in Fig. 6. As shown in the first row, when σ_r increases, the learned edges start to include not only object boundaries but also background textures, which degrades the performance for semantic segmentation in our method (*i.e.*, noisy edges make it hard to propagate information between neighboring pixels). As shown in the second row, varying σ_s does not change the learned edges a lot, as long as its value is large enough (*i.e.*, ≥ 90).

We show *val* set performance (with the best values of σ_s and σ_r) for each method in Tab. 2. The method DT-Gradient improves over the baseline DeepLab by 1.7%. While DT-SE is 0.9% better than DT-Gradient, DT-EdgeNet further enhances performance (4.1% over baseline). Even though DT-EdgeNet is 1.2% lower than DeepLab-CRF, it is several times faster, as we discuss later. Moreover, we have found that combining DT-EdgeNet and dense CRF yields the best performance (0.8% better than DeepLab-CRF). In this hybrid DT-EdgeNet+DenseCRF scheme we post-process the DT filtered score maps in an extra fully-connected CRF step.

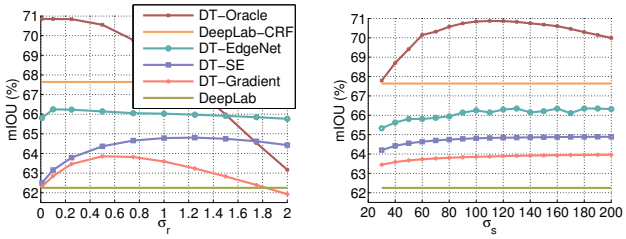
Trimap Similar to [23, 26, 5], we quantify the accuracy of the proposed model near object boundaries. We use the "void" label annotated on PASCAL VOC 2012 validation



(a) Image (b) $\sigma_s = 100, \sigma_r = 0.1$ (c) $\sigma_s = 100, \sigma_r = 0.5$ (d) $\sigma_s = 100, \sigma_r = 2$ (e) $\sigma_s = 100, \sigma_r = 10$

(f) Groundtruth (g) $\sigma_s = 50, \sigma_r = 0.1$ (h) $\sigma_s = 90, \sigma_r = 0.1$ (i) $\sigma_s = 130, \sigma_r = 0.1$ (j) $\sigma_s = 170, \sigma_r = 0.1$

Figure 6. Effect of varying domain transform's σ_s and σ_r. First row: when σ_s is fixed and σ_r increases, the EdgeNet starts to include more background edges. Second row: when σ_r is fixed, varying σ_s has little effect on learned edges.



(a) (b)

Figure 5. VOC 2012 val set. Effect of varying σ_s and σ_r. (a) Fix $\sigma_s = 100$ and vary σ_r. (b) Use the best σ_r from (a) and vary σ_s.

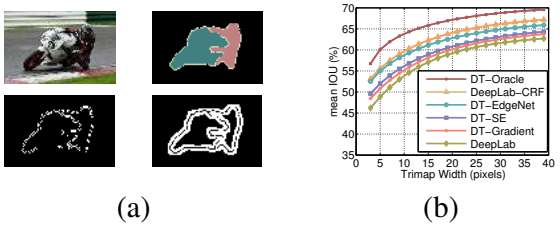
Method	mIOU (%)
DeepLab	62.25
DeepLab-CRF	67.64
DT-Gradient	63.96
DT-SE	64.89
DT-EdgeNet	66.35
DT-EdgeNet + DenseCRF	68.44
DT-Oracle	70.88

Table 2. Performance on PASCAL VOC 2012 val set.

set. The annotations usually correspond to object boundaries. We compute the mean IOU for the pixels that lie within a narrow band (called trimap) of "void" labels, and vary the width of the band, as shown in Fig. 7.

Qualitative results We show some semantic segmentation results on PASCAL VOC 2012 val set in Fig. 9. DT-EdgeNet visually improves over the baseline DeepLab and DT-SE. Besides, when comparing the edges learned by Structured Edges and our EdgeNet, we found that EdgeNet better captures the object exterior boundaries and responds less than SE to interior edges. We also show failure cases in the bottom two rows of Fig. 9. The first is due to the wrong predictions from DeepLab, and the second due to the difficulty in localizing object boundaries with cluttered background.

Test set results After finding the best hyper-parameters, we evaluate our models on the *test* set. As shown in the top of Tab. 4, DT-SE improves 2.7% over the baseline DeepLab, and DT-EdgeNet can further enhance the performance to 69.0% (3.9% better than baseline), which is 1.3% behind



(a) (b)

Figure 7. (a) Some trimap examples (top-left: image. top-right: ground-truth. bottom-left: trimap of 2 pixels. bottom-right: trimap of 10 pixels). (b) Segmentation result within a band around the object boundaries for the proposed methods (mean IOU).

employing a fully-connected CRF as post-processing (*i.e.*, DeepLab-CRF) to smooth the results. However, if we also incorporate a fully-connected CRF as post-processing to our model, we can further increase performance to 71.2%.

Models pretrained with MS-COCO We perform another experiment with the stronger baseline of [34], where DeepLab is pretrained with the MS-COCO 2014 dataset [29]. Our goal is to test if we can still obtain improvements with the proposed methods over that stronger baseline. We use the same optimal values of hyper-parameters as before, and report the results on validation set in Tab. 3. We still observe 1.6% and 2.7% improvement over the baseline by DT-SE and DT-EdgeNet, respectively. Besides, adding a fully-connected CRF to DT-EdgeNet can bring another 1.8% improvement. We then evaluate the models on test set in the bottom of Tab. 4. Our best model, DT-EdgeNet, improves the baseline DeepLab by 2.8%, while it is 1.0% lower than DeepLab-CRF. When combining DT-EdgeNet and a fully-connected CRF, we achieve 73.6% on the test set. Note the gap between DT-EdgeNet and DeepLab-CRF becomes smaller when stronger baseline is used.

Incorporating multi-scale inputs State-of-art models on the PASCAL VOC 2012 leaderboard usually employ multi-scale features (either multi-scale inputs [10, 28, 7] or features from intermediate layers of DCNN [31, 19, 5]). Motivated by this, we further combine our proposed discriminatively trained domain transform and the model of [7], yielding 76.3% performance on test set, 1.5% behind current best models [28] which jointly train CRF and DCNN [6]

Method	mIOU (%)
DeepLab	67.31
DeepLab-CRF	71.01
DT-SE	68.94
DT-EdgeNet	69.96
DT-EdgeNet + DenseCRF	71.77

Table 3. Performance on PASCAL VOC 2012 val set. The models have been pretrained on *MS-COCO 2014* dataset.

EdgeNet on BSDS500 We further evaluate the edge detection performance of our learned EdgeNet on the test set of BSDS500 [1]. We employ the standard metrics to evaluate edge detection accuracy: fixed contour threshold (ODS F-score), per-image best threshold (OIS F-score), and average precision (AP). We also apply a standard non-maximal suppression technique to the edge maps produced by EdgeNet for evaluation. Our method attains ODS=0.718, OIS=0.731, and AP=0.685. As shown in Fig. 8, interestingly, our EdgeNet yields a reasonably good performance (only 3% worse than Structured Edges [11] in terms of ODS F-score), while our EdgeNet is not trained on BSDS500 and there is no edge supervision during training on PASCAL VOC 2012.

Comparison with dense CRF Employing a fully-connected CRF is an effective method to improve the segmentation performance. Our best model (DT-EdgeNet) is 1.3% and 1.0% lower than DeepLab-CRF on PASCAL VOC 2012 test set when the models are pretrained with ImageNet or MS-COCO, respectively. However, our method is many times faster in terms of computation time. To quantify this, we time the inference computation on 50 PASCAL VOC 2012 validation images. As shown in Tab. 5, for CPU timing, on a machine with Intel i7-4790K CPU, the well-optimized dense CRF implementation [26] with 10 mean-field iterations takes 830 ms/image, while our implementation of domain transform with $K = 3$ iterations (each iteration consists of separable two-pass filterings across rows and columns) takes 180 ms/image (4.6 times faster). On a NVIDIA Tesla K40 GPU, our GPU implementation of domain transform further reduces the average computation time to 25 ms/image. In our GPU implementation, the total computational cost of the proposed method (EdgeNet+DT) is 26.2 ms/image, which amounts to a modest overhead (about 18%) compared to the 145 ms/image required by DeepLab. Note there is no publicly available GPU implementation of dense CRF inference yet.

5. Conclusions

We have presented an approach to learn edge maps useful for semantic image segmentation in a unified system that is trained discriminatively in an end-to-end fashion. The proposed method builds on the domain transform, an edge-preserving filter traditionally used for graphics applications. We show that backpropagating through the domain transform allows us to learn an task-specific edge map optimized for semantic segmentation. Filtering the raw semantic segmentation maps produced by deep fully convolutional networks with our learned domain transform leads to improved localization accuracy near object boundaries. The resulting

Method	ImageNet	COCO
DeepLab [5, 34]	65.1	68.9
DeepLab-CRF [5, 34]	70.3	72.7
DT-SE	67.8	70.7
DT-EdgeNet	69.0	71.7
DT-EdgeNet + DenseCRF	71.2	73.6
DeepLab-CRF-Attention [7]	-	75.7
DeepLab-CRF-Attention-DT	-	76.3
CRF-RNN [46]	72.0	74.7
BoxSup [10]	-	75.2
CentraleSuperBoundaries++ [24]	-	76.0
DPN [30]	74.1	77.5
Adelaide_Context [28]	75.3	77.8

Table 4. mIOU (%) on PASCAL VOC 2012 *test* set. We evaluate our models with two settings: the models are (1) pretrained with ImageNet, and (2) further pretrained with MS-COCO.

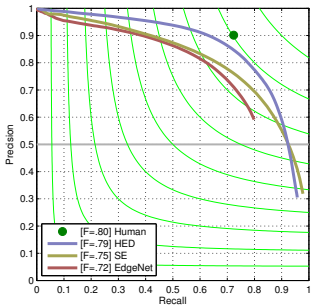
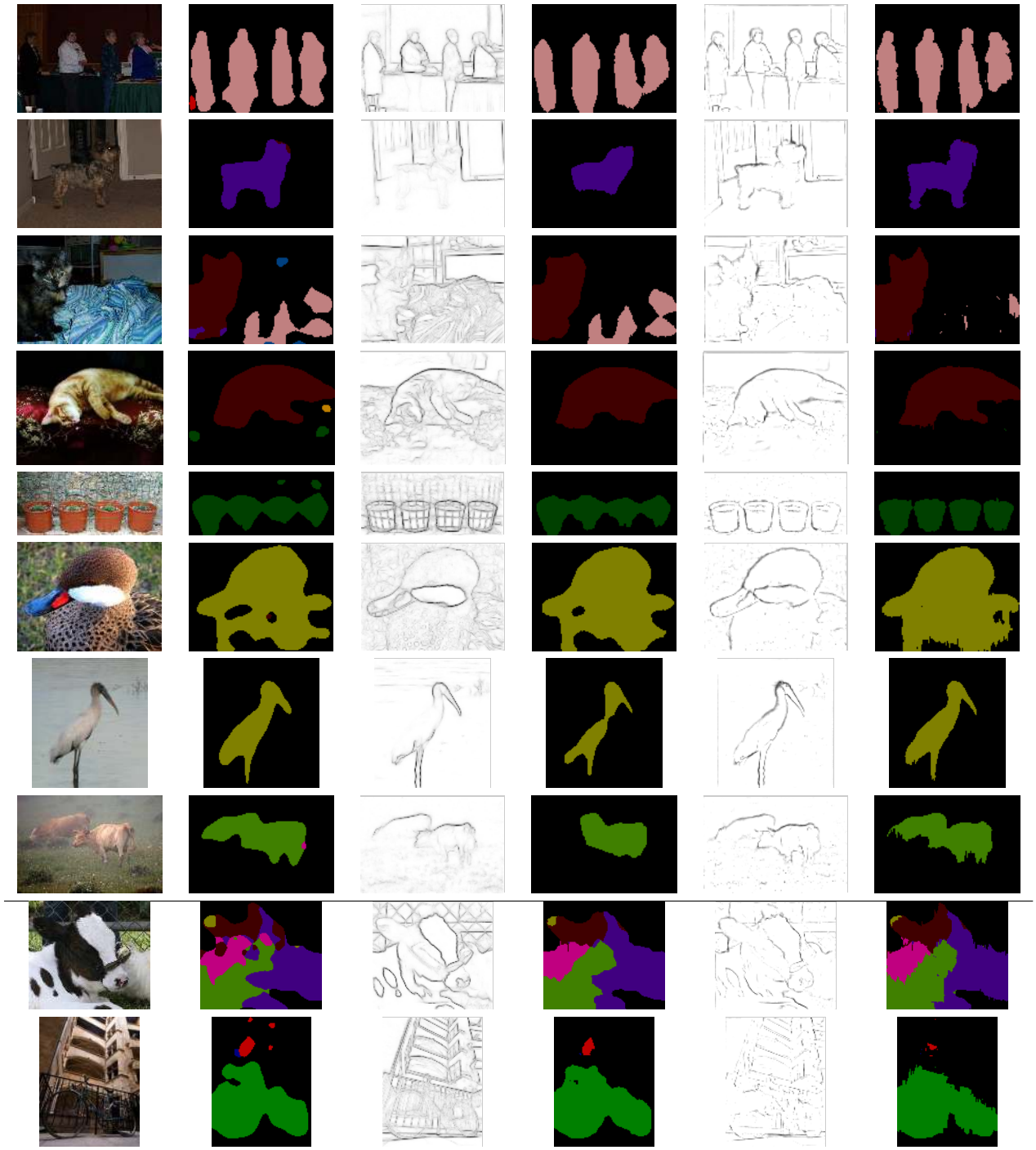


Figure 8. Evaluation of our learned EdgeNet on the test set of BSDS500. Note that our EdgeNet is only trained on PASCAL VOC 2012 semantic segmentation task without edge supervision.

Method	CPU time	GPU time
DeepLab	5240	145
EdgeNet	20 (0.4%)	1.2 (0.8%)
Dense CRF (10 iterations)	830 (15.8%)	-
DT (3 iterations)	180 (3.4%)	25 (17.2%)
CRF-RNN (CRF part) [46]	1482	-

Table 5. Average inference time (ms/image). Number in parentheses is the percentage w.r.t. the DeepLab computation. Note that EdgeNet computation time is improved by performing convolution first and then upsampling.



(a) Image (b) Baseline (c) SE (d) DT-SE (e) EdgeNet (f) DT-EdgeNet

Figure 9. Visualizing results on VOC 2012 val set. For each row, we show (a) Image, (b) Baseline DeepLab segmentation result, (c) edges produced by Structured Edges, (d) segmentation result with Structured Edges, (e) edges generated by EdgeNet, and (f) segmentation result with EdgeNet. Note that our EdgeNet better captures the object boundaries and responds less to the background or object interior edges. For example, see the legs of left second person in the first image or the dog shapes in the second image. Two failure examples in the bottom.

scheme is several times faster than fully-connected CRFs that have been previously used for this purpose.

Acknowledgments This work wast partly supported by ARO 62250-CS and NIH Grant 5R01EY022247-03.

References

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *PAMI*, 33(5):898–916, May 2011.

[2] G. Bertasius, J. Shi, and L. Torresani. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In *CVPR*, 2015.

[3] G. Bertasius, J. Shi, and L. Torresani. High-for-low and low-for-high: Efficient boundary detection from deep object features and its applications to high-level vision. In *ICCV*, 2015.

[4] W. Byeon, T. M. Breuel, F. Raue, and M. Liwicki. Scene labeling with lstm recurrent neural networks. In *CVPR*, 2015.

[5] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015.

[6] L.-C. Chen, A. Schwing, A. Yuille, and R. Urtasun. Learning deep structured models. In *ICML*, 2015.

[7] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille. Attention to scale: Scale-aware semantic image segmentation. *arXiv:1511.03339*, 2015.

[8] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv:1409.1259*, 2014.

[9] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv:1412.3555*, 2014.

[10] J. Dai, K. He, and J. Sun. Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *ICCV*, 2015.

[11] P. Dollár and C. L. Zitnick. Structured forests for fast edge detection. In *ICCV*, 2013.

[12] J. L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[13] M. Everingham, S. M. A. Eslami, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserma. The pascal visual object classes challenge a retrospective. *IJCV*, 2014.

[14] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *PAMI*, 2013.

[15] Y. Ganin and V. Lempitsky. Nˆ4-fields: Neural network nearest neighbor fields for image transforms. In *ACCV*, 2014.

[16] E. S. L. Gastal and M. M. Oliveira. Domain transform for edge-aware image and video processing. In *SIGGRAPH*, 2011.

[17] A. Graves and J. Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *NIPS*, 2009.

[18] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *ICCV*, 2011.

[19] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015.

[20] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[21] J.-J. Hwang and T.-L. Liu. Pixel-wise deep learning for contour detection. In *ICLR*, 2015.

[22] Y. Jia et al. Caffe: Convolutional architecture for fast feature embedding. *arXiv:1408.5093*, 2014.

[23] P. Kohli, P. H. Torr, et al. Robust higher order potentials for enforcing label consistency. *IJCV*, 82(3):302–324, 2009.

[24] I. Kokkinos. Pushing the boundaries of boundary detection using deep learning. In *ICLR*, 2016.

[25] S. Konishi, A. L. Yuille, J. M. Coughlan, and S. C. Zhu. Statistical edge detection: Learning and evaluating edge cues. *PAMI*, 25(1):57–74, 2003.

[26] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *NIPS*, 2011.

[27] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[28] G. Lin, C. Shen, I. Reid, et al. Efficient piecewise training of deep structured models for semantic segmentation. *arXiv:1504.01013*, 2015.

[29] T.-Y. Lin et al. Microsoft COCO: Common objects in context. In *ECCV*, 2014.

[30] Z. Liu, X. Li, P. Luo, C. C. Loy, and X. Tang. Semantic image segmentation via deep parsing network. In *ICCV*, 2015.

[31] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[32] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich. Feedforward semantic segmentation with zoom-out features. In *CVPR*, 2015.

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

[34] G. Papandreou, L.-C. Chen, K. Murphy, and A. L. Yuille. Weakly- and semi-supervised learning of a dcnn for semantic image segmentation. In *ICCV*, 2015.

[35] P. Pinheiro and R. Collobert. Recurrent convolutional neural networks for scene labeling. In *ICML*, 2014.

[36] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015.

[37] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak. Convolutional, long short-term memory, fully connected deep neural networks. In *ICASSP*, 2015.

[38] A. G. Schwing and R. Urtasun. Fully connected deep structured networks. *arXiv:1503.02351*, 2015.

[39] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang. Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *CVPR*, 2015.

[40] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[41] R. Socher, B. Huval, B. Bath, C. D. Manning, and A. Y. Ng. Convolutional-recursive deep learning for 3d object classification. In *NIPS*, 2012.

[42] V. Vineet, J. Warrell, and P. H. Torr. Filter-based mean-field inference for random fields with higher-order terms and product label-spaces. *IJCV*, 110(3):290–307, 2014.

[43] F. Visin, K. Kastner, K. Cho, M. Matteucci, A. Courville, and Y. Bengio. Renet: A recurrent neural network based alternative to convolutional networks. *arXiv:1505.00393*, 2015.

[44] S. Xie and Z. Tu. Holistically-nested edge detection. In *ICCV*, 2015.

[45] M. D. Zeiler, G. W. Taylor, and R. Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *ICCV*, 2011.

[46] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr. Conditional random fields as recurrent neural networks. In *ICCV*, 2015.