Mapping Auto-context Decision Forests to Deep ConvNets for Semantic Segmentation

David L. Richmond*[1]
Dagmar Kainmueller*[1]
Michael Y. Yang[2]
Eugene W. Myers[1]
Carsten Rother[2]

[1] Max Planck Institute of Molecular Cell Biology and Genetics
Dresden, DE
[2] Technical University of Dresden
Dresden, DE

1 Supplemental Materials

In Section 1.1.1, we describe the training parameters used to train the stacked RF and deep ConvNet for the Kinect example (Section 4.1 in the paper). In Section 1.1.2, we describe the training parameters used to train the stacked RF and deep ConvNet for the zebrafish example (Section 4.2 in the paper). We also describe the parameters used for training the equivalent deep ConvNet with random weight initialization. In Section 1.2, we present the algorithm developed to map the parameters from a RF-initialized ConvNet back to the original stacked RF, after training by back-propagation (Section 3.2 in the paper). Additionally, we include Supplemental Figures 1 and 2, which are referenced from the paper.

1.1 Training Parameters

1.1.1 Kinect

Stacked RF. We trained a two-level stacked RF, with the following parameters at every level: 10 trees, maximum depth 12, stop node splitting if less than 25 samples. We selected 20 samples per class per image for training, and used the scale invariant offset features from [3], with standard deviation, $\sigma = 50$ in each dimension. Each split node selected the best from a random sample of 100 such features.

ConvNet. We mapped the RF stack to a deep ConvNet with 5 hidden layers, as described in Section 3.1. For efficient training, the global parameters influencing the sharpness of the *tanh* activation functions were reduced such that the network could transmit a strong gradient via back-propagation. However, softening these parameters moves the deep ConvNet further from its initialization by the equivalent stacked RF. We evaluated a range of initialization parameters and found $str_{01} = str_{34} = str_{67} = 100$, $str_{12} = str_{45} = str_{78} = 1$, $str_{23} = str_{56} = str_{89} = 0.1$ to be a good compromise, where str_{ij} is the multiplicative factor applied to weights, w_{H_i,H_j}, and bias, b_{H_j}.

We trained the ConvNet using back-propagation and stochastic gradient descent (SGD), with a cross-entropy loss function. During back-propagation, we maintained the sparse connectivity from RF initialization, allowing only the weights on pre-existing edges to change, corresponding to the *sparse* training scheme from [5].

Since the network is designed for whole-image inputs, we first cropped the training images around the region of foreground pixels, and then down-sampled them by 25x. Learning rate, r, was set such that for the i^{th} iteration of SGD, $r(i) = a(1+i/b)^{-1}$ with hyper-parameters $a = 0.01$ and $b = 400$ iterations. Momentum, μ, was set according to the following schedule: $\mu = \mathtt{min}\{\mu_{max}, 1-3/(i+5)\}$, where $\mu_{max} = 0.95$ [4].

1.1.2 Zebrafish

Stacked RF. We trained a three-level RF stack, with the following forest parameters at every level: 16 trees, maximum depth 12, stop node splitting if less than 25 samples. Features were extracted from the images using a standard filter bank, and then normalized to zero mean, unit variance. The number of random features tested in each node was set to the square root of the total number of input features. For each randomly selected feature, 10 additional contextual features were also considered, with X and Y offsets within a 129x129 pixel window. Training samples were generated by sub-sampling the training images 3x in each dimension and then randomly selecting 25% of these samples for training.

ConvNet. We mapped the RF stack to a deep ConvNet with 8 hidden layers. The ConvNet was initialized and trained exactly as for the Kinect example, with the following exeptions: (i) We used a class-balanced cross-entropy loss function, (ii) Training samples were generated by sub-sampling the training images 9x in each dimension. (iii) Learning rate parameters were as follows: $a = 0.01$ and $b = 96$ iterations. (iv) Momentum was initialized to $\mu = 0.4$, and increased to 0.7 after 96 iterations. We observed convergence after only 1-2 passes through the training data, similar to what was reported by [1].

ConvNet from Random Initialization. As discussed in Section 4.2 of the paper, for comparison to the RF-initialized weights described above, we also trained ConvNets with the same architecture, but with random weight initialization. Weights were initialized according to a Gaussian distribution with zero mean and standard deviation, $\sigma = 0.01$. We applied a similar SGD training routine, and re-tuned the hyper-parameters as follows: $a = 3 * 10^{-5}$, $b = 96$ iterations, momentum was initialized to 0.4 and increased to 0.99 after 96 iterations. Larger step-sizes failed to train. Networks were trained for 2500 iterations.

Fully Convolutional Network. As discussed in Section 4.2 of the paper, we also compared our method with the Fully Convolutional Network (FCN) [2]. This network was downloaded from Caffe's Model Zoo , and initialized with weights fine-tuned from the ILSVRC-trained VGG-16 model. We trained all layers of the network using SGD with a learning rate of 10^{-9}, momentum of 0.99 and weight decay of 0.0005.

1.2 Mapping Back Algorithm

The following algorithm was used to map the parameters from a trained ConvNet back to the original stacked RF architecture, and is referred to as Map Back #2 (see Section 3.2 of the paper). We applied this algorithm to the zebrafish data set (Figure 1: panel 3, and Supplemental Figure 2(f)).

*https://github.com/BVLC/caffe/wiki/Model-Zoo#fcn

Algorithm 1 Algorithm for mapping deep ConvNet back to K-level stacked RF

```
1. Push all training data through ConvNet
2. Store activations $a_{\mathbf{x}}(H_{3k-1}(l))$, for $k = 1...K$
for  $i = 1 : K$ do
    Push all training data through stacked RF to level $i$
    Store leaf($\mathbf{x}$), for every tree and every sample $\mathbf{x}$, at level $i$
    Update votes in $i^{th}$ RF to $\hat{y}_c^l$, according to Equation 2
end for
```

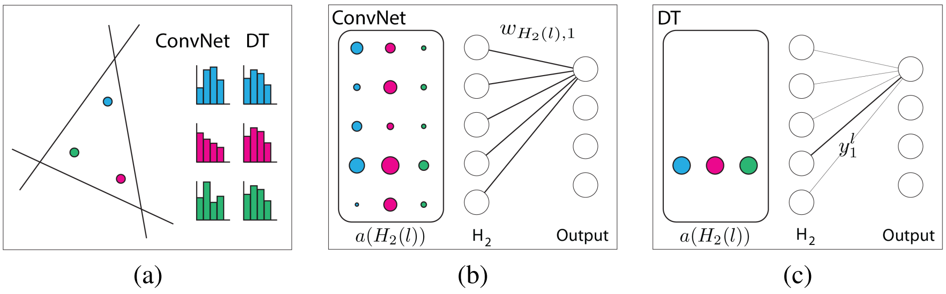

Figure 1: **Mapping ConvNet back to a RF.** (a) Three samples (blue, magenta, green) falling into the leaf of a DT, corresponding to a subset of feature space, have the same posterior distributions; however, in a ConvNet their posteriors can be different. (b) Corresponding activation pattern $a(H_2(l))$ for the three samples shown in (a) at hidden layer 2 of the RF-initialized ConvNet. Radius of circles denotes the strength of the activation. The output layer receives the inner product of the activation pattern with weights $w_{H_2(l),c}$ (only weights to class 1 shown for simplicity). (c) Activation pattern in corresponding DT. Note, the inner product reduces to the value y_1^l for class 1. In Equation 2, we compute the optimal value of y_c^l, namely $\hat{y}_c^l$, to mimize the difference between the output of the DT and the ConvNet.

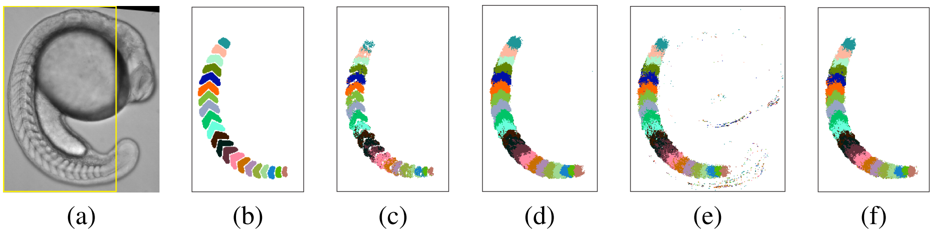

Figure 2: **Comparison of different methods for zebrafish somite labeling.** (a) Raw image of zebrafish. Yellow box denotes crop for b,c,d,f. (b) Ground truth labeling. (c) Prediction of stacked RF. (d) Prediction of corresponding deep ConvNet, after parameter refinement by back-propagation. (e) Prediction of "Map Back #1" stacked RF. (f) Prediction of "Map Back #2" stacked RF. See Section 3.2 for details of map back algorithms.

References

[1] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[2] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[3] Jamie Shotton, Andrew W. Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. In *CVPR*, 2011.

[4] Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, 2013.

[5] Johannes Welbl. Casting random forests as artificial neural networks (and profiting from it). In *GCPR*, 2014.