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

Abstract

We consider the task of pixel-wise semantic segmentation given a small set of labeled training images. Among two of the most popular techniques to address this task are Random Forests (RF) and Neural Networks (NN). In this work, we explore the relationship between two special forms of these techniques: stacked RFs (namely Auto-context) and deep Convolutional Neural Networks (ConvNet). Our main contribution is to show that Auto-context can be mapped to a deep ConvNet with novel architecture, and thereby trained end-to-end. This mapping can be viewed as an intelligent initialization of a deep ConvNet, enabling training even in the face of very limited amounts of training data. We also demonstrate an approximate mapping back from the refined ConvNet to a second stacked RF, with improved performance over the original. We experimentally verify that these mappings outperform stacked RFs for two different applications in computer vision and biology: Kinect-based body part labeling from depth images, and somite segmentation in microscopy images of developing zebrafish.

1 Introduction

Deep learning has transformed the field of computer vision, and now rivals human-level performance in tasks such as image recognition [16] and object detection [9]. These advances have been fuelled by large labeled data sets, such as ImageNet [25], that can be used to train deep ConvNets. Once trained, these models serve as generic feature extractors, and can be applied to a wide range of problems using simple refinement techniques [9]. An example of this is semantic segmentation by a Fully Convolutional Network that was pre-trained for image classification on ImageNet [19].

Despite the overwhelming success of the prescribed approach, there are still many specialized tasks which cannot be easily addressed by refining pre-trained networks, and for which there does not exist a sufficiently large data set to train a high capacity ConvNet from scratch. An important example of this is in biomedical imaging, where no AlexNet exists, and there is often a dearth of publicly available data.

A common strategy when training data is limited, is to use ensemble approaches, such as Random Forest classifiers (RF). The use of stacked classifiers, such as Auto-context [31],

*Shared first authors

Pages 144.1-144.12
DOI: https://dx.doi.org/10.5244/C.30.144

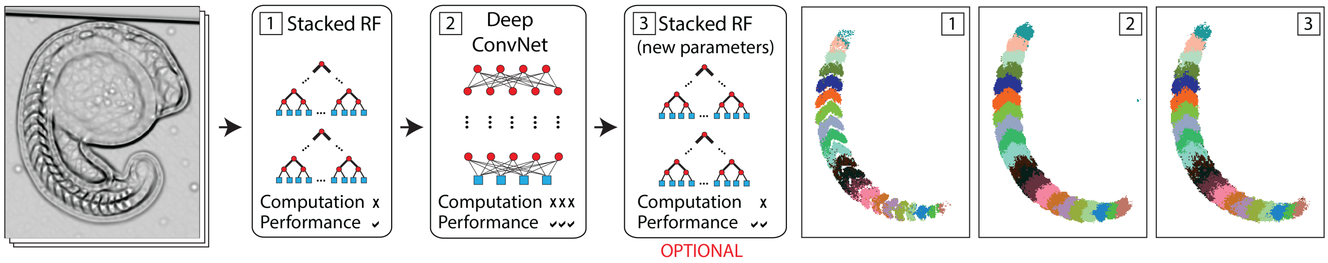


Figure 1: **Overview.** Our method (left) and corresponding results (right) for semantic segmentation of somites in microscopy images of developing zebrafish. (1) A stacked RF is trained to predict dense semantic labels from an input feature stack. (2) The stacked RF is then mapped to a deep ConvNet and further trained by back-propagation to improve performance. (3) Optionally, the ConvNet is mapped back to a stacked RF with updated parameters. The new stacked RF performs worse than the ConvNet but requires much less computation, and is better than the original RF.

creates "deep" classifiers that have been shown to learn contextual information and thereby improve performance on many tasks such as object-class segmentation [28], facade segmentation [13], and brain segmentation [31]. However, this strategy has the limitation that the stack of classifiers is trained greedily; in contrast to the end-to-end training of deep ConvNets. Thus, there is a need for methods that train stacked classifiers end-to-end. Our work addresses this issue by exploiting the connection between decision trees (DT) and NNs [27]. Figure 1 depicts our proposed pipeline.

Contributions:

1. We show that a stacked RF with contextual features is a special case of a deep ConvNet with sparse convolutional kernels.

2. We describe a mapping from a stacked RF to a sparse, deep ConvNet, and utilize this mapping to initialize the ConvNet from a pre-trained stacked RF. This leads to superior results on semantic segmentation with limited training samples, compared to alternative strategies.

3. We describe an approximate mapping of our sparse, deep ConvNet back to a stacked RF with updated parameters, for more computationally efficient evaluation, *e.g.*, for low power devices. We show that this improves over the performance of the original stacked RF.

4. Due to our special ConvNet architecture we are able to gain new insights of the activation pattern of internal layers, with respect to semantic labels. In particular, we observe that the common smoothing strategy in stacked RFs is naturally learned by our ConvNet.

2 Related Work

Our work relates to (i) global optimization of RF classifiers, (ii) feature learning in stacked RF models, and (iii) applying ConvNets to the task of semantic segmentation.

Global Optimization of RFs. The limitations of traditional greedy RF construction [2] have been addressed by numerous works. In [30], the authors learn DTs by the standard method (see [6] for a detailed description of this method), followed by a process called "fuzzification", replacing all threshold split decisions with smooth sigmoid functions, and re-optimizing the split parameters by back-propagation. In [22], they learn to combine the predictions from each DT so that the complementary information between multiple trees is optimally exploited with respect to a final loss function. After training a standard RF, they retrain the distributions stored in the leaves, and prune the DTs to accomplish compression and avoid overfitting. However, [22] does not retrain the parameters of the internal split nodes of individual DTs, whereas [30] does not retrain the combination of trees in the forest.

A related approach is to train an RF, and then map to a shallow NN with two hidden layers and refine the parameters by back-propagation. This was originally demonstrated for classification [27, 32], and more recently for regression [1]. This effectively "fuzzifies" threshold split decisions, and simultaneously enables training with respect to a final loss function. Hence as opposed to [30] and [22], all model parameters are learned simultaneously in an end-to-end fashion. Our work builds upon [27, 32]: We extend their approach to a deep ConvNet, inspired by the Auto-context algorithm [31], and apply it to semantic segmentation.

Feature Learning in a RF Framework. Auto-context introduces new contextual features during the learning process, and thus is a form of feature learning. Numerous works have generalized this approach. In Entangled Random Forests (ERFs) [20], spatial dependencies are learned using "entanglement features" in each DT, without the need for stacking. Geodesic Forests [14] apply image-aware geodesic smoothing to the class distributions, to generate features for deeper nodes in the DT. However, these approaches are still limited by greedy parameter optimization.

In a more traditional approach to feature learning, Neural Decision Forests [3] mix RFs and NNs by using multi-layer perceptrons (MLP) as soft split functions, to jointly tackle the problem of data representation and discriminative learning. This approach can obtain superior results with smaller trees, at the cost of more complicated split functions; however, the MLPs in each split node are trained independently of each other. The authors in [15] address this limitation by training the entire system end-to-end, and this is the most closely related to our work; however, they only apply their model to the task of image classification. This work could be adapted to pixel-wise segmentation using a sliding-window approach; however, without the use of stacked classifiers their model would not capture contextual information beyond the input window. Furthermore, our work generates a different ConvNet architecture with large, sparse convolutional kernels, and interpretable internal layers.

ConvNets for Semantic Segmentation. ConvNets can be applied to semantic segmentation either in a tile-based manner [5], or using "whole-image-at-a-time" processing in a Fully Convolutional Network (FCN) [19]. A challenge of these approaches is that the built-in spatial invariance of ConvNets trained for image classification leads to coarse-graining effects on the output. A variant of FCN called U-Net was recently proposed in [24], and uses skip layers to combat coarse-graining. In [33], they address coarse-graining by expressing mean-field inference in a dense CRF as a Recurrent Neural Network (RNN), and concatenating this RNN behind a FCN, for end-to-end training of all parameters. Notably, they demonstrate a significant boost in performance on the Pascal VOC 2012 segmentation benchmark; however, this model is trained on large scale data and has not been applied to scenarios, such as biomedical images, with limited training examples.

Unsupervised pre-training has been used successfully to leverage smaller labeled training sets [10, 21, 26]; however, fully supervised training on large data sets still gives higher performance. A common practice is to train a ConvNet on a large training set, and then fine tune the parameters on the target data [9]; however, this requires a closely related task with a large labeled data set, such as ImageNet. Another strategy to address the dependency on training data, is to expand a small labeled training set through data augmentation [24].

We propose a novel strategy for semantic segmentation with limited training data. Similar to [8, 17], we employ supervised pre-training, but in a complementary model, namely the popular Auto-context model [31]. Our approach avoids coarse-graining by generating a

ConvNet architecture with no striding or pooling layers. Our method achieves a large receptive field with few parameters by using sparse convolutional kernels, similar to [4]; however, we learn the optimal position of the non-zero kernel element(s) during construction of the RF stack. There has been recent interest in the use of sparse convolutional kernels for reducing computation in ConvNets [11, 12, 18], indeed ConvNets are known to be highly redundant and the number of parameters can be reduced by up to 90% with only a 1% loss in accuracy [18].

3 Method

In Section 3.1, we describe our main contribution, namely how to map a stack of RFs onto a deep ConvNet. In Section 3.2, we describe our second contribution, an algorithm for mapping our deep ConvNet back onto the original RF stack, with updated parameters. Our contributions build upon the mapping of a RF to a NN with two hidden layers as proposed in [27, 32]. In the following we briefly review this mapping, adopting the notation of [32] for consistency.

The architecture of the two layer NN is as follows: The first hidden layer has one neuron, $H_1(n)$, for every split node in the tree (Figure 2b). We use n to denote the split node index. This neuron fires $-/+1$ to encode if the input goes left or right at that node in the tree. The weights in this layer, $w_{f(n),H_1(n)}$, are initialized to 1 for the input feature, $f(n)$, used by the corresponding node in the DT, and 0 otherwise. This can model axis-aligned split functions with a single non-zero weight per neuron, or oblique split functions with multiple non-zero weights per neuron. The weights associated with the bias node, $b_{H_1(n)}$, store the threshold for each split node in the RF. A hyperbolic tangent activation function is used to approximate the binary split function, and finally all weights are multiplied by a global constant that tunes the sharpness of the split decision.

The second hidden layer has one neuron, $H_2(l)$, per leaf node (Figure 2b), and fires $0/1$ to encode if the input ends up in this leaf or not. We use l to denote the leaf node index, and $\text{leaf}(\mathbf{x})$ to denote the unique leaf that sample $\mathbf{x}$ ends up in for each tree. Each neuron is connected to all of the split nodes on the path to that leaf, with weights $w_{H_1(n),H_2(l)}$ equal to $-/+1$ depending on whether the leaf is part of the left or right sub-tree of that split node. The weights to the bias node, $b_{H_2(l)}$, store the length of the path. Again, a hyperbolic tangent activation function is used, and all weights are multiplied by a global constant tuning the sharpness of its response.

The output layer has one neuron per class. The weights, $w_{H_2(l),c}$, are fully connected and store the histogram, y_c^l, over classes, c, from the respective leaf of the tree (Figure 2b). There are no bias nodes in this layer, and softmax activation is used to ensure a probabilistic interpretation. To extend from a tree to a forest, simply connect hidden layer 2 from each tree directly to the output layer (Figure 2c). The inner product sums the result from each tree, and the softmax activation normalizes the output.

We now discuss the relationship between RFs with contextual features and ConvNets. In many applications such as body-pose estimation [22], medical image labeling [20], and scene labeling [31], contextual information is included in the form of "offset features" that are selected from within a window defined by a maximum offset. Such an RF can be viewed as a special case of a ConvNet, with sparse convolutional kernels and no max pooling layers (Figure 3). The convolutions in hidden layer 1 have width matching the size of the window, and are very sparse. *E.g.,* it is common to have only a single non-zero element in this convolution kernel, or in the case of medical imaging, to use average intensity over a smaller

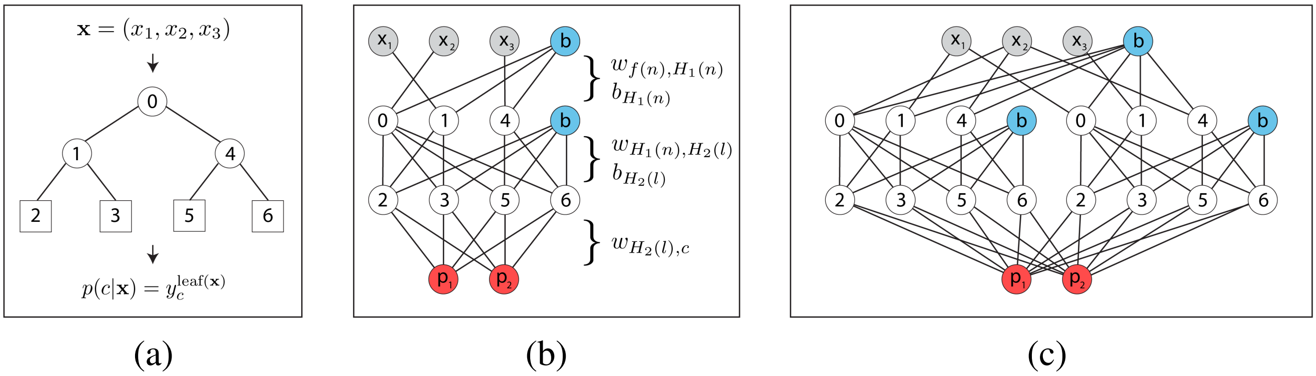


Figure 2: **Mapping from a RF to a NN.** (a) A shallow DT. Nodes are labeled to show mapping to NN. (b) Corresponding NN with two hidden layers. The first hidden layer is connected to the input layer through weights $w_{f(n),H_1(n)}$, where $f(n)$ is the feature used by split node n. This layer outputs the split decision for each split node of the DT (numbered 0,1,4). The weights $w_{H_1(n),H_2(l)}$ between the two hidden layers encode the structure of the tree. In particular, the split nodes along the path to leaf l are connected to $H_2(l)$. For example, leaf node 5 is connected to split nodes 0 and 4, but not split node 1. The second hidden layer encodes leaf membership for each leaf node (numbered 2,3,5,6). The final weights $w_{H_2(l),c}$ are fully connected and store the votes y_c^l for each leaf l and class c. Gray: Input feature nodes. Blue: Bias nodes. Red: Prediction nodes, $p(c|\mathbf{x})$. (c) NN corresponding to a RF with two DTs, each with the same architecture as in (a). Note that, while the two DTs have the same architecture, they use different input features at each split node, and do not share weights.

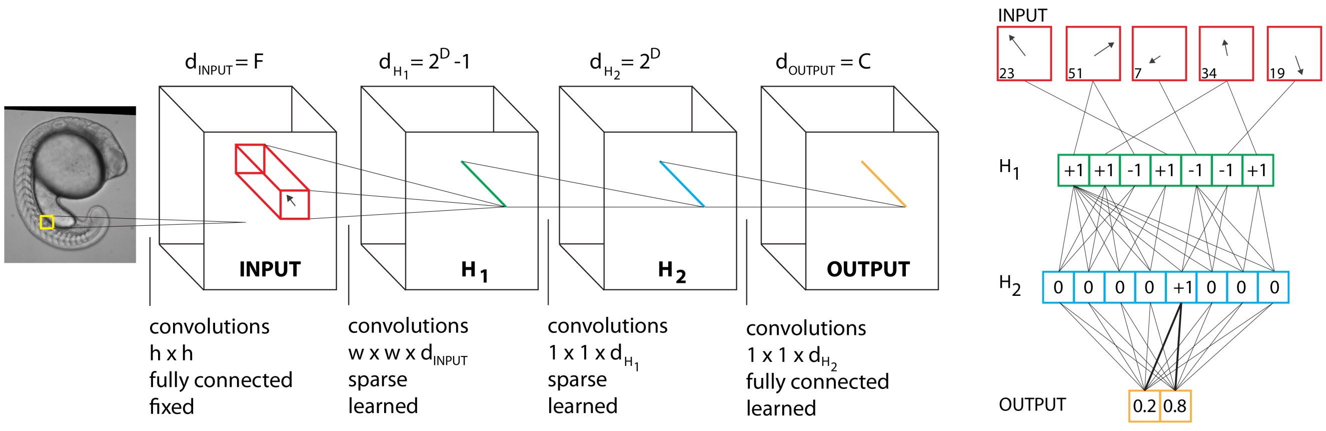
offset window [20]. The second layer convolutions are similarly very sparse, with the number of non-zero elements equal to the depth of the tree.

3.1 Mapping a RF Stack to a Deep ConvNet

A stacked RF consists of multiple RF classifiers in a sequence, such that subsequent RF classifiers in the stack can use the predictions from the previous RF as input features (see Figure 4a). Training is done iteratively, from the first to the last RF in the stack (see [31] for more details). It was noted in the original Auto-context algorithm that it is important to allow the later RF classifiers to select features, not only from the output of the previous classifier, but also from the input feature stack. Finally, to capture contextual information, these features are sampled with a learned offset.

We map this architecture onto a deep ConvNet as follows: each RF is individually mapped to a ConvNet, and then concatenated such that the layers corresponding to intermediate RF predictions become hidden layers, used as input to the next ConvNet in the sequence (Figure 4b). We also connect the input feature stack as bias nodes in hidden layers H_{3k}, $k = 1 \ldots K-1$, and introduce contextual (*i.e.* offset) features with the sparse convolution kernels discussed above. Due to the use of these contextual features, individual pixels cannot be processed independently, but rather the complete image must be run through one level at a time (similar to the Auto-context algorithm), such that all features are available for the next level. Finally, we remove the softmax activation from internal layers H_{3k}, $k = 1 \ldots K-1$, and normalize their output, to match the behaviour of the RF stack. For a K-level RF stack, this generates a deep ConvNet with $3K-1$ hidden layers.

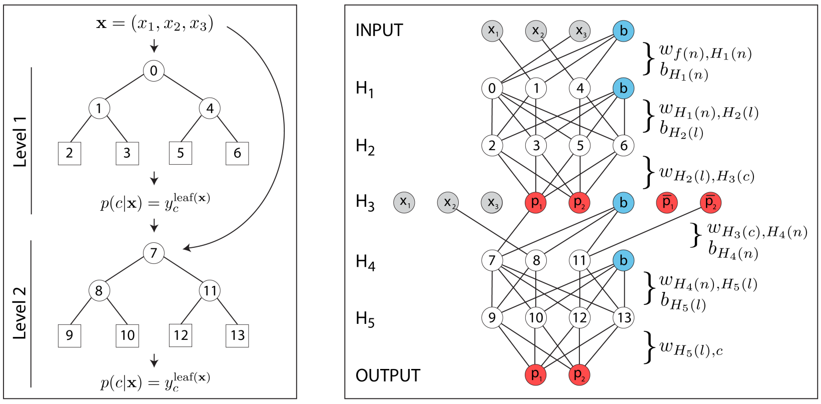
An interesting observation is that addition of trees to the RF and/or growing trees to



(a) (b)

Figure 3: **ConvNet architecture of a RF.** (a) ConvNet architecture for dense semantic segmentation, corresponding to a RF with contextual features. The variables are: h - size of input convolution kernels; F - number of input convolution kernels; w - window size for offset features; d - number of feature maps in each layer; D - depth of corresponding decision tree; C - number of classes. (b) An example, where the RF is a single DT with depth D = 3, and 2 output classes. One pixel is classified in (b), corresponding to the region in (a) with similar color-coding. The input layer (red) extracts features with a fixed offset (shown by arrows) and filter type (index into filter stack, shown at bottom left of each node). Activation values are shown for nodes in hidden layers 1,2 and the output layer. In this example, the sample ends up in leaf #5. Bias nodes are not shown for simplicity.

a greater depth simply increases the width of the ConvNet, but is always restricted to a shallow architecture with only two hidden layers. However, stacking RFs naturally increases the depth of the ConvNet architecture.



(a) (b)

Figure 4: **Mapping from a stacked RF to a deep ConvNet.** (a) A stacked RF consisting of two shallow decision trees. The second RF takes as input the original stack of convolutional filter responses, and the output of the previous RF across the entire window over which contextual features can be sampled. (b) Corresponding ConvNet with 5 hidden layers. Same color coding and node labeling as in Figure 2. In this example, the second DT learned to use filter response x_2, the RF output for class 1 at that same pixel (*i.e.*, p_1), and the RF output for class 2 at some different offset pixel, denoted $\bar{p}_2$. Note that $\bar{p}_2$ is not a bias node; it is a contextual feature and its value depends on weights in previous layers.

3.2 Mapping the Deep ConvNet back to a RF Stack

ConvNets are highly redundant [18] and thus require a lot of additional computation, which may limit their applications *e.g.* on low power devices. We explore the possibility of mapping our deep ConvNet back to the more computationally efficient architecture of a stacked RF. Given a ConvNet constructed from a K-level RF stack as described above, the weights $w_{H_{3k-2},H_{3k-1}}$, $k = 1...K$ manifest the original tree structure. Thus, keeping these weights and the corresponding biases, $b_{H_{3k-1}}$, fixed during training allows the ConvNet to be mapped back onto the structure of the original RF stack. For a single level stack, the mapping is: (i) $\theta(n) = -b_{H_1(n)}/w_{f(n),H_1(n)}$, where $\theta(n)$ is the threshold for split node n, and (ii) $y_c^l = w_{H_2(l),c}$. We refer to this as "Map Back #1". When evaluating this RF, a softmax activation function needs to be applied to the output distribution to mimic inference in the ConvNet. For deeper stacks, the output of *each* RF must be post-processed with the corresponding activation function in the ConvNet, which in this paper is simple class normalization.

The above approach is appropriate if only a single leaf neuron fires in hidden layer 2 for each sample. However, after training by back-propagation, this activation pattern will likely be distributed, with many neurons contributing to the prediction, and our mapping may not make optimal use of the learned parameter refinement. Here, we propose a strategy to capture the distributed activation of the ConvNet. For input $\mathbf{x}$ and class c, we would ideally like to store in leaf($\mathbf{x}$) of each DT, the following inner product: $y_c^{\text{leaf}(\mathbf{x})} = z_{\mathbf{x}}(c) := \sum_l a_{\mathbf{x}}(H_2(l)) \cdot w_{H_2(l),c}$, where $a_{\mathbf{x}}(H_2(l))$ is the activation of neuron $H_2(l)$.

This mapping would elicit the identical output from the RF as from the ConvNet for input $\mathbf{x}$. However, the activation pattern will vary for different training samples that end up in the same leaf, so this mapping cannot be satisfied simultaneously for the whole training set. This results from the fact that DTs store distributions in their leaves that represent a piecewise-constant function on the input feature space, while the re-trained ConvNet allows for a non-piecewise-constant function (Supplemental Figure 1). As a compromise, we seek new vote distributions $\hat{y}_c^l$, for each c, l, to minimise the following error, averaged over the finite set of training samples, X^{train}.

$$\sum_{\mathbf{x}\in X^{train}:\ \text{leaf}(\mathbf{x})=l} \left(z_{\mathbf{x}}(c) - \hat{y}_c^l\right)^2 \tag{1}$$

Equation 1 can be solved analytically, yielding the following result:

$$\hat{y}_c^l = \frac{1}{|\{\mathbf{x} \in X^{train} :\ \text{leaf}(\mathbf{x}) = l\}|} \sum_{\mathbf{x}\in X^{train}:\ \text{leaf}(\mathbf{x})=l} z_{\mathbf{x}}(c) \tag{2}$$

This is a simple average of $z_{\mathbf{x}}(c)$ over all samples that end up in the same leaf of the corresponding DT. We refer to this as "Map Back #2". To implement this algorithm in a stack, we start by determining leaf membership for every sample and every tree in the first level of the RF stack. We then update the votes according to Equation 2. This is then repeated for all subsequent levels of the stack. See Algorithm 1 in the Supplemental Materials for more details. In the trivial case where, for every sample, a single neuron fires with unit activation in layers H_{3k-1}, $k = 1...K$, this is equivalent to "Map Back #1".

4 Results

The forward and backward mappings described above were implemented in Matlab, and tested on two different applications: Kinect-based body part labeling from depth images, and somite segmentation in microscopy images of developing zebrafish.

4.1 Kinect Body Part Classification

Experimental Setup. We applied our method to human body part classification from Kinect depth images, a domain where Random Forests have been highly successful [29]. We use the recently provided data set in [7], since there is no publicly available data set from the original paper [29]. It contains 2000 training images, and 500 testing images, each 320x240 pixels, containing 19 foreground classes and 1 background class (Figure 5(a,b) for an example). We evaluate the pixel accuracy, averaged over all foreground classes, as was done by [7]. Note that background is trivially classified.

Training Parameters. We first trained a two-level stacked RF, and then mapped it to a deep ConvNet with 5 hidden layers, as described in Section 3. We trained the ConvNet using back-propagation and stochastic gradient descent (SGD) with momentum. SGD training is applied by passing images through the network one at a time, and computing the gradient averaged over all pixels (*i.e.*, batch size = 1 image). Thus, we do "whole-image-at-a-time" training, as in [19]. We trained for 8000 iterations, which takes approximately 10 hours in our CPU-based Matlab implementation. See Supplemental Materials for training parameters.

Results. With our initial two-level stacked RF, we achieved a pixel accuracy of 0.82, comparable to the original result of 0.79 [7] (Figure 5(c), Table 1(RF)). After mapping to a deep ConvNet and re-training, we achieved an accuracy of 0.91, corresponding to an 11% relative improvement over the RF stack (Figure 5(d), Table 1(ConvNet)). This final result is comparable to the state-of-the-art result on this data set which aims to compress RFs by learning a better combination of their constituent trees [22]. They achieve a class-balanced pixel accuracy of 0.92 over all classes, including the background class, for a model size of 6.8MB. Our model is smaller, at 3.3MB, due to our use of fewer and shallower trees. Furthermore, they report that their optimization procedure takes multiple days, compared to our overnight refinement by back-propagation. However, due to the different error metric, and their evaluation on a selected subset of pixels, the results are not directly comparable.

We also tried mapping the ConvNet back to the initial stacked RF architecture with updated parameters. We first employed the trivial approach of mapping weights directly onto votes, similar to what was done in the RF to NN forward mapping; however, this reduced the Dice score to 0.74 (Table 1(MB1)), worse than the performance of the initial RF. Next we applied Algorithm 1 (Supplemental Materials), which yielded a final Dice score of 0.85 (Table 1(MB2)). Thus, we achieve a 4% relative improvement of our RF stack, which retains its exact tree structure, by mapping to a deep ConvNet, training all weights by back-propagation, and mapping back to the original RF stack with updated threshold and leaf distributions. However, note that the performance of the final RF is lower than the ConvNet, due to the approximate nature of the mapping.

Insights. The architecture of the deep ConvNet preserves the intermediate prediction layers of the RF stack, which generates one image for each class at the same resolution as the input image. This enables us to gain insights on internal ConvNet layers. However, due to back-propagation training, these images no longer represent probability distributions. In particular, the pixel values can now be negative. We visualized the internal layers to

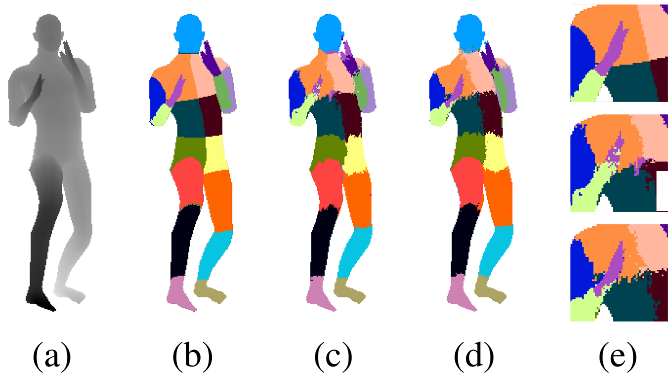


Figure 5: **Example result of Kinect body part classification.** (a) Depth image. (b) Ground truth labels. (c) Result of stacked RF. (d) Result of RF-initialized ConvNet, after re-training. The accuracy for this test image increases from 0.88 to 0.94 on foreground classes. (e) Crop of hands for GT, RF and ConvNet, from top to bottom.

better understand how they changed during additional training in the ConvNet (Figure 6(a)). Interestingly, we noticed that compared to the stacked RF, the internal activation layers in the ConvNet were less thresholded, and fired on adjacent body parts. A common strategy in stacked classification is to introduce smoothing between the layers of the stack (*e.g.* [13, 14, 23]), and it appears that a similar strategy is naturally learned by the deep ConvNet.

4.2 Zebrafish Somite Classification

Experimental Setup. We next applied our method to semantic segmentation of 21 somites[1] and 1 background class in a data set of 32 images (800x950 pixels) of developing zebrafish. Experts in biology manually created ground truth segmentations of these images[2]. The data set was split into 16 images for training and 16 images for test. Two additional training images were generated from each original training image by random rotation of the originals. We evaluated the resulting segmentation by means of the class-balanced Dice score.

Training Parameters. We first trained a three-level stacked RF, and then mapped it to a deep ConvNet with 8 hidden layers. The ConvNet was initialized and trained exactly as for the Kinect example; however, with different hyper-parameters (Supplemental Materials).

Results. Segmentation of the test data by means of the resulting three-level stacked RF achieved an average Dice score of 0.60 (Supplemental Figure 2(c) and Table 1(RF)). The RF-initialized ConvNet achieved a Dice score of 0.66 after re-training, corresponding to a 10% relative improvement (Supplemental Figure 2(d) and Table 1(ConvNet)). This result matches previous State-of-the-Art results on this data set [23], but without the need for time-consuming model-based inference. It's interesting to note that training a deep ConvNet with 8 hidden layers using hyperbolic tangent activation functions is typically extremely difficult, but works well here, likely due to the good initialization of the network. We discuss insights on the internal activation layers of this network in Figure 6(b). We also mapped the ConvNet back to the initial stacked RF architecture with updated parameters. MB#1 yielded a Dice score of 0.59, and MB#2 a final score of 0.63, a 5% improvement from the original RF model (Table 1 and Supplemental Figure 2).

We also considered the task of training the same ConvNet architecture from a random initialization (Supplemental Materials). We trained the network first maintaining the sparsity of the weight layers, and then fully connecting the layers corresponding to tree connectivity; however, these yielded final Dice scores of only 0.04 and 0.15, respectively. Finally, we compared our method with the Fully Convolutional Network (FCN), a state-of-the-art model for semantic segmentation [19]. The model was downloaded from Caffe's Model Zoo[3], and

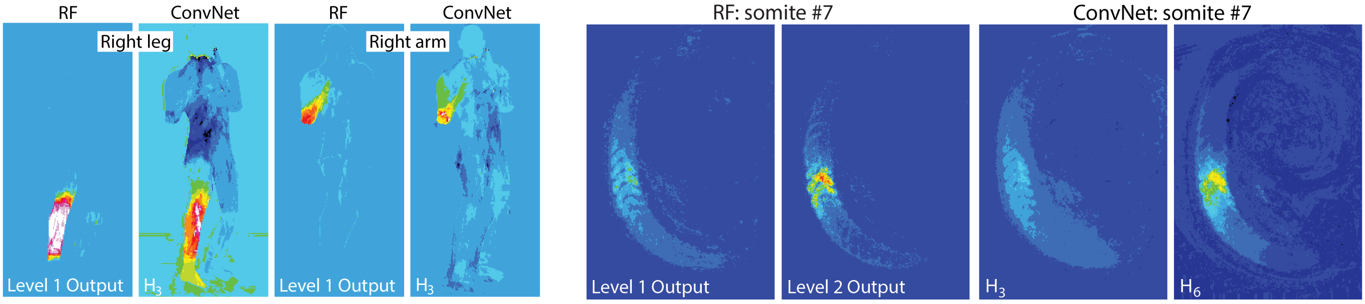
[1] Somites are the metameric units that give rise to muscle and bone, including vertebrae.

[2] The data is publicly available at http://tinyurl.com/zyvc2lu

[3] https://github.com/BVLC/caffe/wiki/Model-Zoo#fcn

	RF	ConvNet	MB1	MB2
Kinect	0.82	**0.91**	0.74	0.85
Zebrafish	0.60	**0.66**	0.59	0.63

Table 1: **Comparison of dense semantic labeling.** Dice score is reported for the initial stacked RF (RF), RF-initialized and re-trained ConvNet (ConvNet), and after mapping the ConvNet back to a stacked RF using Map Back #1 and #2 (MB1 and MB2, respectively. See Section 3.2 for details). Higher Dice score corresponds to a more accurate segmentation.



(a) (b)

Figure 6: **Visualization of internal activation layers.** We visualize the probability maps output by the intermediate output layers of the RF stack (*e.g.*, Level 1,2 Output), and the activation maps from the corresponding hidden layers of the ConvNet (*e.g.*, H_3, H_6) for (a) Kinect body parts, and (b) zebrafish somites. Notice that the activation from the ConvNet appears smoothed along the direction of the foreground classes compared to the noisier output of the stacked RF. Best viewed in colour.

initialized with weights fine-tuned from the ILSVRC-trained VGG-16 model. This model (and others in the Caffe Zoo) take as input an RGB image, and are not directly amenable to grayscale microscopy images. We created 3-channel images by duplicating the grayscale image, and fine-tuned the network for approximately 1 day on a single Nvidia K-40 GPU (Supplemental Materials). The FCN network failed to train successfully, achieving a Dice score of only 0.18, due either to incompatibility of this pre-trained model with 1-channel images, the significant difference in task domain, and/or the limited size of the training set.

5 Conclusions and Future Work

We have exploited a new mapping between stacked RFs and deep ConvNets, and demonstrated the benefits of this mapping for semantic segmentation with limited training data. There are many exciting avenues for future research. Our ConvNet architecture produces internal activation images, one for each class, that are directly interpretable. Thus, it is straightforward to incorporate differentiable model layers, *e.g.*, dense convolutions, that operate on the output of these layers. In the midterm, we are excited about extending our architecture and also merging it with existing ConvNet architectures. *E.g.*, we would like to test whether our sparse ConvNet architecture can be trained from scratch on small data sets, by incorporating standard ConvNet tricks such as drop-out regularization and ReLU activations. We are also interested to test whether additional ConvNet architectures, such as a smaller FCN model with single channel input, can be trained on our data. Finally, we would like to compare the performance of our model directly to the recent model from [15], and explore whether our Auto-context ConvNet can also be incorporated after the feature extraction layers in a traditional ConvNet.

References

[1] Gérard Biau, Erwan Scornet, and Johannes Welbl. Neural random forests. *arXiv.org*, 2016.

[2] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[3] Samuel Rota Bulò and Peter Kontschieder. Neural decision forests for semantic image labelling. In *CVPR*, 2014.

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015.

[5] Dan Ciresan, Alessandro Giusti, Luca M Gambardella, and Jürgen Schmidhuber. Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images. In *NIPS*, 2012.

[6] Antonio Criminisi and Jamie Shotton. *Decision Forests for Computer Vision and Medical Image Analysis*. Springer, 2013.

[7] Misha Denil, David Matheson, and Nando de Freitas. Consistency of online random forests. *Journal of Machine Learning Research*, 28(3):1256–1264, 2013.

[8] Scott Fahlman and Christian Lebiere. The cascade-correlation learning architecture. In *NIPS*, 1989.

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

[10] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[11] Yani Ioannou, Duncan Roberston, Jamie Shotton, Roberto Cipolla, and Antonio Criminisi. Training convolutional neural networks with low-rank filters for efficient image classification. In *ICLR*, 2016.

[12] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. In *BMVC*, 2014.

[13] Varun Jampani, Raghudeep Gadde, and Peter Gehler. Efficient facade segmentation using auto-context. In *WACV*, 2015.

[14] Peter Kontschieder, Pushmeet Kohli, Jamie Shotton, and Antonio Criminisi. Geof: Geodesic forests for learning coupled predictors. In *CVPR*, 2013.

[15] Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulo. Deep neural decision forests. In *ICCV*, 2015.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[17] Régis Lengellé and Thierry Denoeux. Training mlps layer by layer using an objective function for internal representations. *Neural Networks*, 9(1):83–97, 1996.

[18] Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Pensky. Sparse Convolutional Neural Networks. In *CVPR*, 2015.

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

[20] Albert Montillo, J Tu, Jamie Shotton, John Winn, Juan Eugenio Iglesias, Dimitris N Metaxas, and Antonio Criminisi. Entanglement and differentiable information gain maximization. In *Decision Forests for Computer Vision and Medical Image Analysis*, 2013.

[21] Marc Aurelio Ranzato, Fu Jie Huang, Y-Lan Boureau, and Yann LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *CVPR*, 2007.

[22] Shaoqing Ren, Xudong Cao, Yichen Wei, and Jian Sun. Global refinement of random forest. In *CVPR*, 2015.

[23] David L Richmond, Dagmar Kainmueller, Ben Glocker, Carsten Rother, and Eugene W Myers. Uncertainty-driven forest predictors for vertebra localization and segmentation. In *MICCAI*, 2015.

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[25] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[26] Pierre Sermanet, Koray Kavukcuoglu, Sumith Chintala, and Yann LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In *CVPR*, 2013.

[27] Ishwar Sethi. Entropy nets: from decision trees to neural networks. *Proceedings of the IEEE*, 78 (10):1605–1613, 1990.

[28] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. In *CVPR*, 2008.

[29] 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.

[30] Alberto Suarez and James Lutsko. Globally optimal fuzzy decision trees for classification and regression. *PAMI*, 21(12):1297–1311, 1999.

[31] Zhuowen Tu and Xiang Bai. Auto-context and its application to high-level vision tasks and 3d brain image segmentation. *PAMI*, 32(10):1744–1757, 2010.

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

[33] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip Torr. Conditional random fields as recurrent neural networks. In *ICCV*, 2015.