Fast Residual Forests: Rapid Ensemble Learning for Semantic Segmentation

Yan Zuo, Tom Drummond
Department of Electrical and Computer Systems Engineering
Monash University, Australia
yan.zuo@monash.edu, tom.drummond@monash.edu

Abstract: In recent times, Convolutional Neural Network (CNN) based approaches have performed exceptionally well in many computer vision related tasks, including classification and segmentation. These approaches have shown that given enough training data and time, they can often perform at a level significantly higher than the alternative methods. However, in the context of robotic learning, it is commonly the case that both time and training data are limited. In this work, we propose a learning approach that is more suitable for robotic learning; it substantially reduces the time required to learn and provides much higher performance when training data is limited. Our method combines random forests with deep convolution networks, leveraging the strengths of both frameworks. We develop a method for generating derivatives from our highly non-linear forest classifier which in turn enables training of the CNN. Furthermore, our method allows leaf distributions in the ensemble classifier to be trained jointly with one another using Stochastic Gradient Descent (SGD), allowing for parallel training of a large number of tree classifiers at once. This results in a drastic increase in training speed. Our model demonstrates significant performance improvements over pure deep learning methods, notably on datasets with limited training data. We apply our method to the outdoor and indoor segmentation datasets of KITTI and NYUv2-40, outperforming multiple pure deep learning methods whilst using a fraction of training time normally required.

Keywords: Fast Learning, Decision Forests, Segmentation

1 Introduction

Deep learning approaches have shown success in learning both feature representations together with their classifiers, yielding large performance gains over classical methods that rely on conventional feature descriptor and classifier frameworks [8, 13]. However, these models are often difficult to train, both in terms of the training time required as well as amount of data necessary to ensure generalisation in the trained model [28]. In robotic learning, situations arise where training time is limited or accurate training data is hard to obtain. Hence, whilst effective, pure deep learning approaches can come up short for these types of robotic applications.

Deep learning has been successfully applied to semantic segmentation tasks [15, 19] - a challenge which plays a vital role in scene understanding for robots where learning the spatial information in an environment is as important as understanding the semantic information within it. Robotic segmentation datasets are often cluttered with many classes and are considered challenging due to the high ratio of classes to training data [6, 27]. Furthermore, due to the nature of segmentation tasks, the amount of training data available is limited by the difficulty of providing accurate labeled data to learn from [3, 27]. Despite this, research towards investigating approaches that generalise well when there is insufficient training data available has been limited. The popular approach for training deep neural networks for semantic segmentation tasks with limited training data involves fine-tuning from network weights trained on a much larger, more general dataset [1, 15, 20]. However, even this does not properly address the situation when the training data is too limited to sufficiently fine-tune the network to perform adequately on a given task.

1st Conference on Robot Learning (CoRL 2017), Mountain View, United States.

In this paper, we present a learning approach which not only dramatically reduces time required to learn, but significantly outperforms many pure deep learning methods on datasets with limited training data. Our method learns feature representations from a CNN which are used to train a decision forest framework. We formulate a method which enables the joint training of both features from the CNN together with a decision forest classifier, unifying the two different frameworks and training the model end-to-end. To this end, we present Fast Residual Forests (FRF) which provides the following benefits:

- It jointly optimises information across all the leaf node predictions of trees in the ensemble, allowing for large numbers of tree classifiers to be trained in a parallel (Section 3.2), drastically reducing training time.
- It combines highly non-linear random decision forests with the convolutional weights of a CNN and uses a novel approach to train the whole system via backpropagation (Section 3.3).
- It demonstrates a significant increase in performance on datasets with limited training data, such as the KITTI 6-class and NYUv2 40-class segmentation datasets, when compared to multiple pure deep learning methods (Sections 4.1 & 4.2).

2 Related Work

Random Forests in Segmentation Random forests have found a wide use of applications in vision-related problems, including segmentation. The early work of [26] used random forests to efficiently generate local pixel-wise priors which were then refined by graph-based methods to infuse them with global contextual information. In [24], single-histogram class models were mapped within a random forest classifier for segmentation tasks. [10] used random forests to learn structured class labels which incorporated joint statistics around a small neighbourhood and used this to perform semantic labelling. [11] enriched the input space with intermediate predictions from a random forest and used this feature space to perform semantic segmentation. [16] used random forests to search for boundaries in a high-dimensional feature space, using these boundaries to perform segmentation on tumors in medical images.

Deep Learning with Decision Forests As deep learning rose to prominence, it quickly became the de facto standard for several vision related tasks, including semantic segmentation. Initial work used the strong features learned by neural networks and leveraged their rich information to train conventional, off-the-shelf classifiers [4, 6, 7]. Following this, end-to-end deep methods which jointly trained CNN features in parallel with a classifier emerged as a natural progression from the typical decoupled frameworks of CNN extractors and classifier pairs [15, 19, 20, 25]. Since then, works in the literature have sought to train conventional CNNs with methods inspired by decision trees [9, 18, 22]. In contrast, our work seeks to directly combine deep learning with decision forest classifiers; most notably, it shares similarities with [12] and [23]. [23] replaced the decision nodes in decision trees with multi-layer perceptrons and used these modified forests to perform semantic segmentation. [12] reformulated split node functions as a soft, differentiable stochastic function, enabling backpropagation to learn both weights in the network and a decision forest classifier. In contrast to [12] and [23], our method does not modify the hard split function found in conventional decision trees and reformulates the problem of learning leaf node distributions to allow for parallelised training of thousands of tree classifiers at once.

3 Theoretical Framework

3.1 Decision Forest Solver

Random Trees Decision trees consist of a set decision nodes and leaf nodes; decision nodes dictate how data is routed down the tree, by splitting data to their corresponding left or right child decision nodes, whilst leaf nodes hold prediction distributions. We define the set of decision nodes as $D = \{d_0, \cdots, d_{N-1}\}$, each holding a decision function $d(x;\theta)$, where θ are the parameters of the decision node. In binary decision trees, a decision function is defined as: $d(x;\theta) : \mathcal{X} \rightarrow [0, 1]$, which routes an instance of data, x, to its respective left or right child decision node. This process

is repeated from the start root decision node until the instance reaches a terminating leaf prediction node $\ell = D(x, \Theta)$. This is illustrated in Fig.1. The leaf nodes in the tree hold class label probability distributions, $q = Q(\ell)$, which serve to classify unseen incoming instances of data. These probability distributions are formed from the training set of observed ground truth class labels of instances routed to them:

$$Q(\ell)_j = \frac{\sum_i \delta(D(x_i), \ell) L_{ij}}{n_\ell} \tag{1}$$

where $n_\ell = \sum_i \delta(D(x_i), \ell)$ is the number of samples routed into leaf node ℓ, and L_{ij} is the observed class label for instance i in class j such that:

$$L_{ij} = \begin{cases} 1, & \text{if instance } i \text{ has class label } j \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

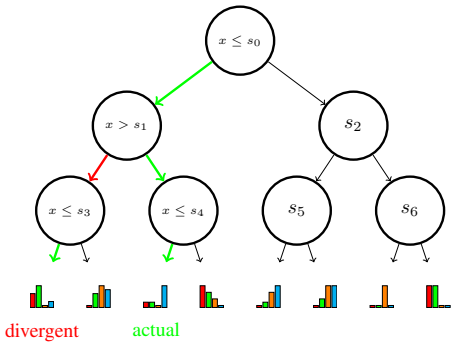


Figure 1: Decision tree routing of training instances. Actual instance routing path (green) and divergent instance path routing (red) highlighted is discussed in Section 3.3

Random Forests In a Random Forest (RF), an ensemble of $\mathcal{T}$ random tree classifiers are combined to give a single classification output. A commonly adopted approach is to average the result across trees in the ensemble:

$$P(\text{class } j | x, \Theta, Q) = \frac{1}{\mathcal{T}} \sum_{t=1}^{\mathcal{T}} Q^t(D^t(x, \Theta^t))_j \tag{3}$$

where Q^t, D^t and Θ^t are the respective distributions, decisions and parameters of tree t, while Θ and Q are the collected parameters of *all* trees' decisions and distributions.

Random Residual Trees For each decision node, we randomly index into the input feature's set of channel values. Positive channel values route the feature right; otherwise the feature is routed to the left. For leaf nodes, we generate residual distributions in our leaf prediction nodes which are designed to combine multiplicatively across the ensemble. Hence, our ensemble gives a final prediction for an instance x as:

$$P(\text{class } j | x, \Theta, Q) = \frac{\prod_{t=1}^{\mathcal{T}} Q^t(D^t(x, \Theta^t))_j}{\sum_k \prod_{t=1}^{\mathcal{T}} Q^t(D^t(x, \Theta^t))_k} \tag{4}$$

3.2 Learning Prediction Nodes

Each residual distribution is optimised so that it combines with other residuals in other trees in the ensemble to form an approximation of the underlying data. Given an instance i and a decision tree d from the ensemble that i is to be routed through, we define the following terms:

- Our stored residual distributions in each leaf prediction node for each class j is given by q_j.
- P_{ij}^- is the logistic value of class j for instance i from Equation (4) before the residual distribution value q_j from tree d is added.

- P_{ij}^{+} is the normalised probability of class j for instance x, which comprises of the combined residual distributions from all trees in the ensemble. This includes the residual distribution value q_j where:

$$P_{ij}^{+} = \frac{P_{ij}^{-} q_j}{\sum_k P_{ik}^{-} q_k} \tag{5}$$

Hence, for each residual $\{q_j\}$ contributing to the final prediction, we look to minimise the following log-loss function:

$$\mathbb{L} = -\sum_{ij} L_{ij} \log\left(P_{ij}^{+}\right) \tag{6}$$

Subsequently, learning of our prediction nodes in the ensemble of trees can be constructed as a convex optimisation problem. We use Stochastic Gradient Descent (SGD) to jointly optimise all residuals distributions in all leaves, across all trees in the ensemble. Thus, for the class l of a particular instance i, we can generate its first-order derivatives of the log-loss with respect to the residual, q_l:

$$\begin{aligned} \frac{\partial \mathbb{L}}{\partial q_l} &= -L_{il} \frac{P_{il}^{-}}{P_{il}^{-} q_l} + \frac{P_{il}^{-}}{\sum_k P_{ik}^{-} q_k} \\ &= -\frac{p_{il}}{q_l} + \frac{P_{il}^{-}}{\sum_k P_{ik}^{-} q_k} \\ &= -(p_{il} - P_{il}^{+}) \end{aligned} \tag{7}$$

Where p_{il} is the ground truth probability of instance i for a given class l. We use Stochastic Gradient Descent (SGD) to minimise our log-loss with respect to the residual distributions stored in our ensemble of trees:

$$q_l^{(t+1)} = q_l^{(t)} + \mu\phi \tag{8}$$

Where μ is the momentum and ϕ is an update term given by:

$$\phi = -\eta \frac{\partial \mathbb{L}}{\partial q_l} - \eta\omega q_l^{(t)} \tag{9}$$

Where $\eta > 0$ is the learning rate and $\omega > 0$ is the weight decay. This allows us to build all residual distributions in all trees in the decision forest in *parallel*, where each residual contribution from each tree is jointly optimised in the context of contributions from all other trees in the forest.

3.3 Learning CNN Weights

Conventionally, weights in a CNN are learned via backpropagation using a softmax layer which provides a loss to drive the training. Here, we replace the softmax layer with a decision forest which serves to provide the loss function to enable learning of the weights by backpropagation.

Loss Function We generate derivatives to drive training by considering an *actual* and *divergent* loss from each node in each tree in the decision forest. For each instance routed through each tree in the decision forest, we define the actual loss of that instance to be the loss computed if the instance was routed normally through each tree. We define the divergent loss assigned by a node in the tree to be the computed loss if for a node n in tree t, the instance is instead routed the other direction (but proceeding through all other nodes in the tree as normal). This is illustrated in Fig.1. Using this concept, we can form an approximation of the loss function for our decision forest which is differentiable and use this to generate derivatives to drive training of the weights in the CNN. For each input instance, we approximate the loss function of our forest solver as a blend of sigmoids of the actual and divergent loss, for a given decision node n:

$$\mathbb{L} = \frac{1}{1 + e^{\lambda(x - s_n)}} \mathbb{L}_1 + \frac{1}{1 + e^{-\lambda(x - s_n)}} \mathbb{L}_2 \tag{10}$$

Where x is the feature value, s_n is the split threshold which the feature is compared against for a given decision node, d_n. For $x \leq s_n$, $\mathbb{L}_1$ and $\mathbb{L}_2$ are the respective actual and divergent losses; for $x > s_n$, $\mathbb{L}_1$ and $\mathbb{L}_2$ are the respective divergent and actual losses. λ indicates the steepness of the sigmoid function which dictates how close a feature needs to be to its respective threshold to affect the backward derivative generated.

Backward Gradients We can use the loss function defined in Eq. (10) to generate backward derivatives and use SGD to train the weights in the CNN. The first-order derivatives of the loss $\mathbb{L}$, with respect to the feature activations x is given by:

$$\begin{aligned}\frac{\partial \mathbb{L}}{\partial x} &= \delta \left(-\frac{\lambda e^{\lambda(x-s_n)}}{(1+e^{\lambda(x-s_n)})^2}\mathbb{L}_1 + \frac{\lambda e^{-\lambda(x-s_n)}}{(1+e^{-\lambda(x-s_n)})^2}\mathbb{L}_2 \right) \\ &= \delta \frac{\lambda e^{-\lambda(x-s_n)}}{(1+e^{-\lambda(x-s_n)})^2}(\mathbb{L}_1 - \mathbb{L}_2)\end{aligned} \tag{11}$$

Where δ is defined by:

$$\delta = \begin{cases} -1, & \text{if } x > s_n \\ 1, & \text{otherwise} \end{cases} \tag{12}$$

Eq. (11) indicates that for a given feature activation x, the backward gradient generated is dependent on the difference between actual and divergent losses computed at the node n the instance was diverted on. For any divergence point, if the divergent loss is lower than the actual loss, this indicates that the feature activation would be better off on the other side of its threshold value. Hence, a gradient is generated which pulls x towards s_n, such that it may cross the threshold. Likewise, if the actual loss is lower than the divergent loss, the gradient generated would push x away from s_n.

4 Experiments

We use FCN-8s [15] as a feature extractor which we attach our forest classifier as a solver; FCN-8s also serves as a baseline comparison in all experiments. All training experiments were performed on a PASCAL GeForce GTX1080. We use features extracted from the `fc7`, `conv5_3` and `conv4_3` layers of the network in [15] to train our decision forest solver. For all our experiments, we build three forests comprising of 1024, 512 and 256 10-level depth trees for each of the `fc7`, `conv5_3` and `conv4_3` convolutional layers respectively. We train the 'at-once' FCN-8s model of [15] with default specified hyperparameters (high momentum of 0.99, mini-batch size of 1, weight decay of $5e{-}4$ and fixed learning rate of 10^{-6}). For our model, we use the exact same hyperparameter settings, and we set our value of $\lambda = 1$ for all experiments. We initialised both our model and [15] with the same network weights, trained on PASCAL-Context [17].

To demonstrate the usefulness of our method in the context of robotic learning on limited training data, we present results in a small-size outdoor scene segmentation dataset (KITTI) [3] and a medium-size indoor scene segmentation dataset (NVYv2) [27]. Our purpose is to illustrate value of our method in applications to robotic learning; as such, we compare against methods that generate pure unaries (without pre-processing on the input such as multi-scaling and data augmentation, as well as post-processing on the output in the form of CRFs). This allows for a more direct and even comparison across learning approaches and highlights the improvements we gain in speed of learning and segmentation performance over our baseline model. To measure our performance, we use the metrics defined in [15]: mean IU, mean accuracy and global pixel accuracy.

4.1 KITTI

KITTI is a small outdoor scene segmentation dataset with 11 classes, comprising of 146 images in total [3] (100 training and 46 testing images). We follow [30] and exempt under-represented classes like pole and pedestrians for evaluation. This turns the dataset into a 6 class problem and we follow [30], using intersection-over-union (IU) as a measurement of performance.

We compare our results in Table 1, outperforming all other listed methods significantly in 4 out of 6 classes and offering a significant improvement to the mean IU metric. We train the model of [15] for 20,000 iterations after which it shows performance improvements over the graph-based methods of [21, 29] and [30]. Comparatively, our approach further improves on the result of [15], using the same amount of training data and vastly less training iterations (2000 iterations).

Next, we demonstrate improvements towards speed of learning that our method offers. The graph on the left side of Fig.3 shows the relative performance of our method compared to [15] across a range of training iterations. We can see that given the same base learning rate (10^{-6}), our model learns more than an order of magnitude faster than [15]. After approximately 20,000 iterations, the

	sky	building	road	sidewalk	vegetation	car	overall
Ren *et al.* [21]	87.4	78.7	72.6	41.3	80.9	59.5	71.9
Tighe *et al.* [29]	81.41	72.7	51.2	17.3	69.9	52.3	60.7
Wang *et. al* [30]	88.6	80.1	80.9	43.6	81.6	63.5	74.8
FCN-8s-heavy (20k iterations) [15]	78.9	84.4	87.3	68.3	86.6	80.4	81.0
Ours (2k iterations)	84.5	**85.9**	**92.3**	**78.8**	**87.8**	80.3	**84.9**

Table 1: KITTI test data performance results: intersection-over-union

model of [15] begins to converge at a performance point of around 81% mean IU. In contrast, our model reaches this performance point a lot earlier (within 500 iterations of training) and continues to improve in performance for another 1,500 iterations before converging at the significantly higher point of approximately 85% mean IU. This represents a reduction by a factor of 40 in the number of training iterations required by our model compared to [15].

Table 4 shows the timings for both training and inference between our model and the model in [15]. We drastically reduce the training time required, requiring only **4.81** minutes compared to [15], which requires 150.11 minutes to reach the same mean IU performance point of 81%. This represents an increase in training speed by a factor of approximately 31. Inference across both models is approximately the same; we gain a small reduction of 2.67 milliseconds in inference time per inference iteration compared to the model in [15].

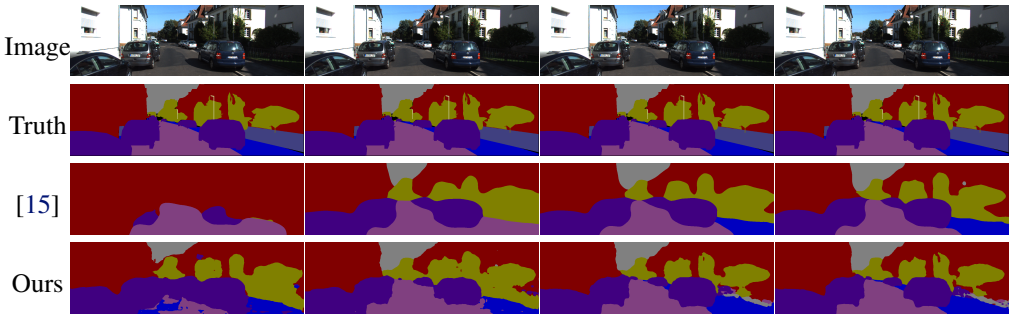
Ablation Study for Tree Depth Additionally, we perform an ablation study on performance of our model versus tree depth selection. Table 2 shows the mean IU performance from shallow trees (1 depth) up to the relatively deep trees used in our final evaluation (10 depth). This shows some limitations of our model - performance converges to approximately the same mean IU (84.5%), regardless of tree depth up to 4 depth. However, for any depth shallower than 4 depth, mean IU performance begins to suffer, indicating that our trees cannot be too shallow for proper learning to occur. The results also seem to indicate deeper trees learn faster (iterations 100 to 500), possibly due to more non-linearity in the classifier and higher modelling capacity.

Tree Depth	No. of Training Iterations				
	100	200	500	1000	2000
1	36.3	58.9	58.3	54.1	53.9
2	62.5	48.7	57.2	63.6	71.6
3	66.5	66.3	65.0	68.6	70.5
4	41.2	62.9	64.4	72.1	84.3
5	39.5	65.3	73.8	81.3	84.6
6	54.8	60.6	67.5	81.9	84.7
7	51.6	74.4	80.4	81.7	84.9
8	57.1	57.6	75.1	**83.9**	**85.0**
9	53.7	74.2	79.7	79.4	84.5
10	**63.8**	**75.3**	**81.5**	82.0	84.9

Table 2: Ablation study on tree depth for KITTI

4.2 NYUDv2

We now show that our method can be extended to larger, more complex scene segmentation datasets. The NYUDv2 [27] dataset is a challenging 40 class indoor scene segmentation problem with pixel-wise labels provided by [5], considered a medium-sized dataset (1449 total images). We use the standard split of 795 training images and 654 testing images. First, we list our best results comparison in Table 3 - note that our method and [14] use only colour information for training whilst [2, 6, 15] use the method of [6] to utilise additional depth information for training. We show that we outperform all colour only methods by a significant margin across all metrics. Furthermore, we even outperform methods that use both colour and depth information for training across the mean accuracy and mean IU metric, and obtain a competitive result in global accuracy compared to [2].



(a) 100 (b) 500 (c) 2k (d) 10k

Figure 2: Qualitative results on KITTI Test data. The first and second rows show the test image and corresponding ground truth segmentation respectively. The third row shows the output of [15] over training iterations. Our model's output is shown in the fourth row over the same training iterations.

	pixel acc. (%)	mean acc. (%)	mean IU (%)
Gupta *et al.* (RGB-HHA) [6]	60.3	35.1	28.6
FCN-32s-heavy (RGBD) [15]	61.5	42.4	30.5
FCN-32s-heavy (RGB-HHA) [15]	64.3	44.9	32.8
FCN-16s-heavy (RGB-HHA) [15]	65.4	46.1	34.0
Eigen *et al.* (RGB-HHA) [2]	65.6	45.1	34.1
FCN-8s (100k iterations) (RGB) [15]	60.6	41.6	29.0
Lin *et. al* (Basemodel) (RGB) [14]	63.5	45.3	32.4
Ours (10k iterations) (RGB)	**64.6**	**48.3**	**34.3**

Table 3: NYUDv2 test results

We again offer a more in-depth analysis of the performance of our model against our baseline model [15]. The graph on the right side of Fig.3 compares the performance of our method to [15] across a range of training iterations. After approximately 100,000 iterations, the model of [15] begins to converge at a performance point of around 29% mean IU (consistent with the published results in [15]). Our model reaches this performance point much earlier (after approximately 4000 iterations of training) and continues to improve in performance until 10,000 iterations of training, at a significantly higher point of approximately 34% mean IU. This represents a reduction by a factor of 25 in the number of training iterations required by our model compared to [15].

Table 4 shows the timings for both training and inference between our model and the model in [15]. We improve on training time, using only **21.08** minutes compared to [15], which requires 533.21 minutes to reach the same mean IU performance point of 29%. This represents an increase in training speed by a factor of approximately 25. We gain a significant reduction in inference time, cutting inference down by more than 20 milliseconds per iteration of inference over the model in [15].

Method	KITTI		NYUv2	
	Total training time (min)	Avg. inference time (ms)	Total training time (min)	Avg. inference time (ms)
FCN-8s-heavy [15]	150.11	109.02	533.21	69.46
Ours	**4.81**	**106.35**	**21.08**	**48.36**

Table 4: Training and inference timings for KITTI and NYUv2

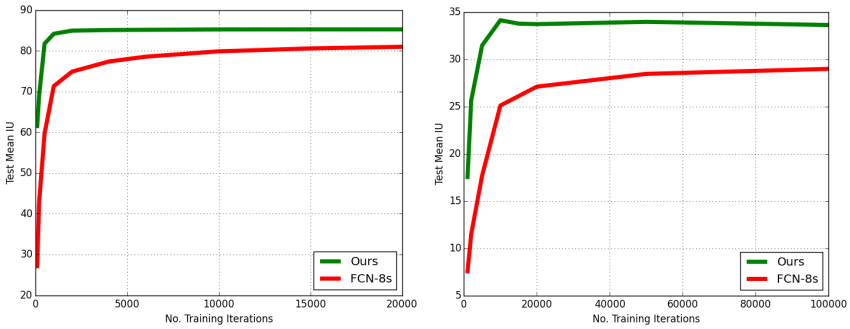
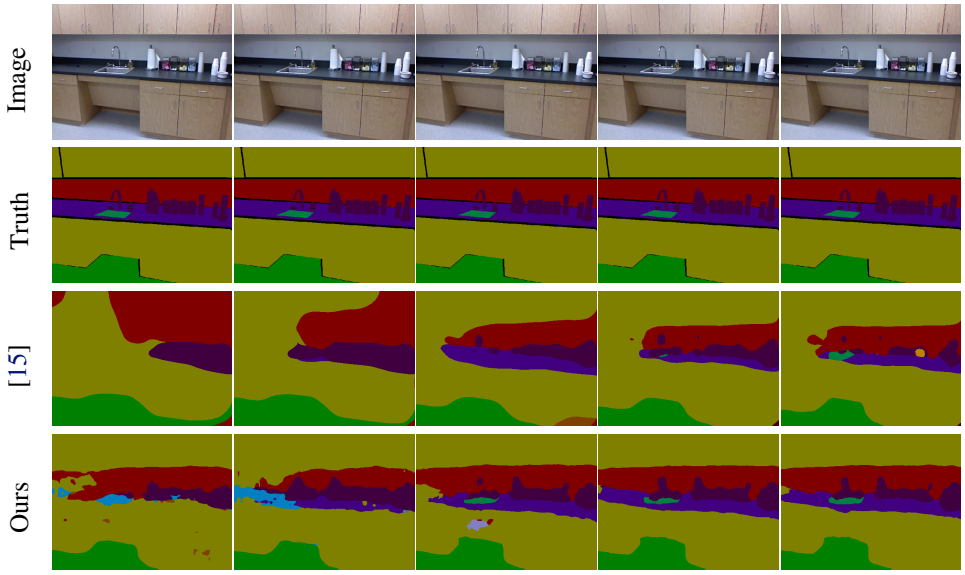


Figure 3: Training iteration timings vs. mean IU performance on KITTI (left) and NYUv2 (right) test data



(a) 1k (b) 2k (c) 10k (d) 25k (e) 50k

Figure 4: Qualitative results on NYUv2-40 Test data. The first and second rows show the test image and corresponding ground truth segmentation respectively. The third row shows the output of [15] over training iterations. Our model's output is shown in the fourth row over the same training iterations.

5 Conclusion

In this paper, we have presented an ensemble learning method for segmentation which demonstrates vast improvements on speed of training. We introduce a hybrid-model which utilises the representational learning of CNNs together with the discriminating power of decision forests. Moreover, we formulate a method that allows for end-to-end learning of both representations and leaf distributions in our decision forest solver. We use this approach to demonstrate successful segmentation results on the KITTI and NYUv2 datasets, outperforming multiple pure deep learning approaches and cutting down training time by more than an order of magnitude.

Acknowledgments

This work was supported by the Australian Research Council Centre of Excellence for Robotic Vision (project number CE1401000016).

References

[1] J. Dai, K. He, and J. Sun. Convolutional feature masking for joint object and stuff segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3992–4000, 2015.

[2] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.

[3] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361. IEEE, 2012.

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[5] S. Gupta, P. Arbelaez, and J. Malik. Perceptual organization and recognition of indoor scenes from rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 564–571, 2013.

[6] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning rich features from rgb-d images for object detection and segmentation. In *European Conference on Computer Vision*, pages 345–360. Springer, 2014.

[7] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *European Conference on Computer Vision*, pages 297–312. Springer, 2014.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, 2016.

[9] Y. Ioannou, D. Robertson, D. Zikic, P. Kontschieder, J. Shotton, M. Brown, and A. Criminisi. Decision forests, convolutional networks and the models in-between. *arXiv preprint arXiv:1603.01250*, 2016.

[10] P. Kontschieder, S. R. Bulo, H. Bischof, and M. Pelillo. Structured class-labels in random forests for semantic image labelling. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2190–2197. IEEE, 2011.

[11] P. Kontschieder, P. Kohli, J. Shotton, and A. Criminisi. Geof: Geodesic forests for learning coupled predictors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 65–72, 2013.

[12] P. Kontschieder, M. Fiterau, A. Criminisi, and S. Rota Bulo. Deep neural decision forests. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1467–1475, 2015.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[14] G. Lin, C. Shen, A. van den Hengel, and I. Reid. Efficient piecewise training of deep structured models for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3194–3203, 2016.

[15] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[16] A. Montillo, J. Tu, J. Shotton, J. Winn, J. Iglesias, D. Metaxas, and A. Criminisi. Entangled forests and differentiable information gain maximization. *Decision Forests in Computer Vision and Medical Image Analysis*, 5:1, 2013.

[17] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille. The role of context for object detection and semantic segmentation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 891–898, 2014.

[18] V. N. Murthy, V. Singh, T. Chen, R. Manmatha, and D. Comaniciu. Deep decision network for multi-class image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2240–2248, 2016.

[19] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528, 2015.

[20] P. H. Pinheiro and R. Collobert. Recurrent convolutional neural networks for scene labeling. In *ICML*, pages 82–90, 2014.

[21] X. Ren, L. Bo, and D. Fox. Rgb-(d) scene labeling: Features and algorithms. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2759–2766. IEEE, 2012.

[22] D. L. Richmond, D. Kainmueller, M. Yang, E. W. Myers, and C. Rother. Mapping stacked decision forests to deep and sparse convolutional neural networks for semantic segmentation. *arXiv preprint arXiv:1507.07583*, 2015.

[23] S. Rota Bulo and P. Kontschieder. Neural decision forests for semantic image labelling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 81–88, 2014.

[24] F. Schroff, A. Criminisi, and A. Zisserman. Object class segmentation using random forests. In *BMVC*, pages 1–10, 2008.

[25] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *International Conference on Learning Representations (ICLR 2014)*, page 16. CBLS, 2013.

[26] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *Computer vision and pattern recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[27] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pages 746–760. Springer, 2012.

[28] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[29] J. Tighe and S. Lazebnik. Finding things: Image parsing with regions and per-exemplar detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3001–3008, 2013.

[30] S. Wang, S. Fidler, and R. Urtasun. Holistic 3d scene understanding from a single geo-tagged image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3964–3972, 2015.