Deep Neural Decision Forests

Peter Kontschieder[†], **Madalina Fiterau**[◇], **Antonio Criminisi**[†], **Samuel Rota Bulò**[⋆]
[†] Microsoft Research Cambridge, UK
[◇] Stanford University California, USA
[⋆] Fondazione Bruno Kessler Trento, Italy

Abstract

We present a novel approach to enrich classification trees with the representation learning ability of deep (neural) networks within an end-to-end trainable architecture. We combine these two worlds via a stochastic and differentiable decision tree model, which steers the formation of latent representations within the hidden layers of a deep network. The proposed model differs from conventional deep networks in that a decision forest provides the final predictions and it differs from conventional decision forests by introducing a principled, joint and global optimization of split and leaf node parameters. Our approach compares favourably to other state-of-the-art deep models on a large-scale image classification task like ImageNet.

1 Introduction

Random forests [Amit and Geman, 1997; Breiman, 2001; Criminisi and Shotton, 2013] have found successfull application in machine learning in general and the computer vision community in particular. They empirically outperform most state-of-the-art learners on high dimensional data problems [Caruana *et al.*, 2008], they are inherently able to deal with multi-class problems and are easily distributable on parallel hardware architectures. Moreover, they are considered to be close to an ideal learner [Hastie *et al.*, 2009]. These facts and many (computationally) appealing properties make them attractive for various research areas and commercial products [Bosch *et al.*, 2007; Brostow *et al.*, 2008; Shotton *et al.*, 2013]. On the downside, random forests lack a machanism to *efficiently* learn internal representations that help to capture the main factors of variation in the data [Bengio *et al.*, 2010].

One of the consolidated findings of modern, deep learning approaches [Krizhevsky *et al.*, 2012; Lin *et al.*, 2013; Szegedy *et al.*, 2014] is that their joint and unified way of learning internal data representations along with the classifiers greatly outperforms conventional feature descriptor & classifier pipelines on different tasks, given enough training data and computation capabilities (see *e.g.* [He *et al.*, 2015] for image classification, [Yu and Deng, 2014] for speech recognition, [Fei-Fei, 2015] for image description).

An interesting open question, which has received little attention in the literature so far, is how to endow random forests with the ability of learning proper internal representations of the input data in order to improve on the generalization capacity of the final classifier. Notable but limited exceptions are [Kontschieder *et al.*, 2013; Montillo *et al.*, 2013] where random forests were trained in an entangled setting, stacking intermediate classifier outputs with the original input data. The approach in [Rota Bulò and Kontschieder, 2014] introduced a way to integrate multi-layer perceptrons as split functions, however, representations were learned only locally at split node level and independently among split nodes. While these attempts can be considered early forms of representation learning in random forests, their prediction accuracies remained below the state-of-the-art.

In this work we present *Deep Neural Decision Forests* – a novel approach to unify appealing properties from representation learning as known from deep architectures with the divide-and-conquer principle of decision trees. We introduce a stochastic, differentiable, and therefore back-propagation compatible version of decision trees, guiding the representation learning in hidden layers of deep networks. Thus, the task for representation learning is to reduce the uncertainty on the routing decisions of a sample taken at the split nodes, such that a globally-defined loss function is minimized. Additionally, for given split node parameters we obtain optimal predictions for all leaves of our trees by minimizing a convex objective, and we provide an optimization algorithm for it that does not depend on tedious step-size selection. Consequently, we can take the optimal decision for a test sample ending up in the leaves, with respect to all the training data and the current state of the network. We show the efficacy of our approach on the challenging ImageNet dataset for large-scale image classification, where we obtain state-of-the-art results with no data augmentation.

2 Decision Trees with Stochastic Routing

Decision trees can be used to tackle a wide range of learning problems [Criminisi and Shotton, 2013] including classification, which will be the focus of this work. Consider a classification problem with input and (finite) output spaces given by $\mathcal{X}$ and $\mathcal{Y}$, respectively. A *decision tree* is a classifier consisting of decision (or split) nodes and prediction (or leaf) nodes organized into a tree structure. Decision nodes indexed

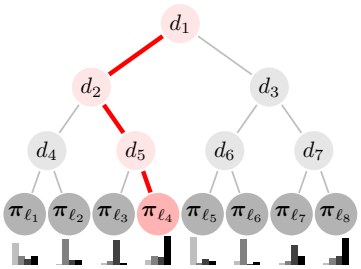

Figure 1: Routing of sample $\boldsymbol{x}$ along a decision tree to leaf ℓ_4. Each node $n \in \mathcal{N}$ along the path steers the sample to the left with probability $d_n(\boldsymbol{x})$ and to the right with probability $\bar{d}_n(\boldsymbol{x}) = 1 - d_n(\boldsymbol{x})$. So, the probability of reaching ℓ_4 is $\mu_{\ell_4}(\boldsymbol{x}) = d_1(\boldsymbol{x})\bar{d}_2(\boldsymbol{x})\bar{d}_5(\boldsymbol{x})$ and the related prediction is $\boldsymbol{\pi}_{\ell_4}$.

by $\mathcal{N}$ are internal nodes of the tree, while prediction nodes indexed by $\mathcal{L}$ are the terminal nodes of the tree. A decision tree classifies a sample by routing it through the tree to a leaf node $\ell \in \mathcal{L}$, where the prediction takes place via a probability distribution $\boldsymbol{\pi}_\ell$ over $\mathcal{Y}$. The routing along the tree is determined through decision functions $d_n(\cdot;\Theta) : \mathcal{X} \to [0,1]$ parametrized by Θ and located in each decision node $n \in \mathcal{N}$, which locally indicate the path to follow. When a sample $\boldsymbol{x} \in \mathcal{X}$ reaches a decision node n it will be sent to the left or right subtree based on the output of $d_n(\boldsymbol{x};\Theta)$. In standard decision forests, d_n is binary and the routing is deterministic. In this paper we will consider instead a probabilistic routing, *i.e.* the routing direction is the output of a Bernoulli random variable with mean $d_n(\boldsymbol{x};\Theta)$. Once a sample ends in a leaf node ℓ, the related tree prediction is given by the class-label distribution $\boldsymbol{\pi}_\ell$ (see Fig. 1 for an illustration). In the case of stochastic routings, the leaf predictions will be weighted by the probability of reaching the leaf. Accordingly, the final prediction for sample $\boldsymbol{x}$ from tree T with decision nodes parametrized by Θ is given by

$$\mathbb{P}_T[y|\boldsymbol{x},\Theta,\boldsymbol{\pi}] = \sum_{\ell\in\mathcal{L}} \pi_{\ell y}\mu_\ell(\boldsymbol{x}|\Theta)\,, \tag{1}$$

where $\boldsymbol{\pi} = (\boldsymbol{\pi}_\ell)_{\ell\in\mathcal{L}}$ and $\pi_{\ell y}$ denotes the probability of a sample reaching leaf ℓ to take on class y, while $\mu_\ell(\boldsymbol{x}|\Theta)$ is regarded as the *routing function* providing the probability that sample $\boldsymbol{x}$ will reach leaf ℓ. Clearly, $\sum_\ell \mu_\ell(\boldsymbol{x}|\Theta) = 1$ for all $\boldsymbol{x} \in \mathcal{X}$. To provide an explicit form for the routing function we introduce the following binary relations that depend on the tree's structure: $\ell \swarrow n$, which is true if ℓ belongs to the left subtree of node n, and $n \searrow \ell$, which is true if ℓ belongs to the right subtree of node n. These relations allow us to express μ_ℓ as follows:

$$\mu_\ell(\boldsymbol{x}|\Theta) = \prod_{n\in\mathcal{N}} d_n(\boldsymbol{x};\Theta)^{\mathbb{1}_{\ell\swarrow n}}\bar{d}_n(\boldsymbol{x};\Theta)^{\mathbb{1}_{n\searrow\ell}}\,, \tag{2}$$

where $\bar{d}_n(\boldsymbol{x};\Theta) = 1 - d_n(\boldsymbol{x};\Theta)$, and $\mathbb{1}_P$ is an indicator function conditioned on the argument P. Although the product in (2) runs over all nodes, only decision nodes along the path from the root to the leaf ℓ contribute to μ_ℓ (assuming $0^0 = 1$), because for all other nodes $\mathbb{1}_{\ell\swarrow n}$ and $\mathbb{1}_{n\searrow\ell}$ will be both 0.

Decision nodes. We consider decision trees having decision functions of the following form to deliver stochastic routings:

$$d_n(\boldsymbol{x};\Theta) = \sigma(f_n(\boldsymbol{x};\Theta))\,, \tag{3}$$

where $\sigma(x) = (1+e^{-x})^{-1}$ is the sigmoid function, and $f_n(\cdot;\Theta) : \mathcal{X} \to \mathbb{R}$ is a real-valued function depending on the sample and the parametrization Θ. The outcome of $d_n(\boldsymbol{x};\Theta) \in [0,1]$ represents the probability of sample $\boldsymbol{x}$ to be steered left in node $n \in \mathcal{N}$. Further details about the functions f_n can be found in Sec. 4, but intuitively depending on how we choose these functions we can model trees having shallow decisions (*e.g.* such as in oblique forests [Heath *et al.*, 1993]) as well as deep ones.

Forests of decision trees. A forest is an ensemble of decision trees $\mathcal{F} = \{T_1,\dots,T_{\mathsf{k}}\}$, which delivers a prediction for sample $\boldsymbol{x}$ by averaging the output of each tree, *i.e.*

$$\mathbb{P}_{\mathcal{F}}[y|\boldsymbol{x}] = \frac{1}{\mathsf{k}}\sum_{h=1}^{\mathsf{k}} \mathbb{P}_{T_h}[y|\boldsymbol{x}]\,, \tag{4}$$

omitting the tree parameters for notational convenience.

3 Learning Trees by Back-Propagation

In order to learn a decision tree defined as per Sec. 2, we need to estimate both, the decision node parametrization Θ and the leaf predictions $\boldsymbol{\pi}$. To carry out the estimation we follow the minimum empirical risk principle with respect to a given data set $\mathcal{T} \subset \mathcal{X}\times\mathcal{Y}$ under log-loss, *i.e.* we pursue minimizers of the following risk term:

$$R(\Theta,\boldsymbol{\pi};\mathcal{T}) = \frac{1}{|\mathcal{T}|}\sum_{(\boldsymbol{x},y)\in\mathcal{T}} L(\Theta,\boldsymbol{\pi};\boldsymbol{x},y)\,, \tag{5}$$

where $L(\Theta,\boldsymbol{\pi};\boldsymbol{x},y)$ is the log-loss term for the training sample $(\boldsymbol{x},y) \in \mathcal{T}$, which is given by

$$L(\Theta,\boldsymbol{\pi};\boldsymbol{x},y) = -\log(\mathbb{P}_T[y|\boldsymbol{x},\Theta,\boldsymbol{\pi}])\,, \tag{6}$$

and $\mathbb{P}_T$ is defined as in (1). We use a two-step optimization strategy, described in the rest of this section, alternating updates of Θ with updates of $\boldsymbol{\pi}$ in a way to minimize (5).

Learning Decision Nodes. All decision functions depend on a common parameter Θ, which in turn parametrizes each function f_n in (3). The minimization of the empirical risk with respect to Θ for a given $\boldsymbol{\pi}$ is, in general, a difficult and large-scale optimization problem, since no assumption has been made about the structure of f_n. As an example, Θ could absorb all the parameters of a deep neural network having f_n as one of its output units. For this reason, we will employ a Stochastic Gradient Descent (SGD) approach to minimize (5) with respect to Θ, as commonly done in the context of deep neural networks:

$$\begin{aligned}\Theta^{(t+1)} &= \Theta^{(t)} - \eta\frac{\partial R}{\partial\Theta}(\Theta^{(t)},\boldsymbol{\pi};\mathcal{B}) \\ &= \Theta^{(t)} - \frac{\eta}{|\mathcal{B}|}\sum_{(\boldsymbol{x},y)\in\mathcal{B}}\frac{\partial L}{\partial\Theta}(\Theta^{(t)},\boldsymbol{\pi};\boldsymbol{x},y)\,.\end{aligned} \tag{7}$$

Here, $\eta > 0$ is the learning rate and $\mathcal{B} \subseteq \mathcal{T}$ is a random subset (*a.k.a.* mini-batch) of samples from the training set. Although not shown explicitly, we additionally consider a momentum term to smooth out the variations of the gradients. The gradient of the loss L with respect to Θ can be decomposed by the chain rule as follows

$$\frac{\partial L}{\partial \Theta}(\Theta, \boldsymbol{\pi}; \boldsymbol{x}, y) = \sum_{n \in \mathcal{N}} \frac{\partial L(\Theta, \boldsymbol{\pi}; \boldsymbol{x}, y)}{\partial f_n(\boldsymbol{x}; \Theta)} \frac{\partial f_n(\boldsymbol{x}; \Theta)}{\partial \Theta} \,. \quad (8)$$

Here, the gradient term that depends on the decision tree is

$$\frac{\partial L(\Theta, \boldsymbol{\pi}; \boldsymbol{x}, y)}{\partial f_n(\boldsymbol{x}; \Theta)} = d_n(\boldsymbol{x}; \Theta) A_{n_r} - \bar{d}_n(\boldsymbol{x}; \Theta) A_{n_l} \,, \quad (9)$$

where n_l and n_r indicate the left and right child of node n, respectively, and we define A_m for a generic node $m \in \mathcal{N}$ as

$$A_m = \frac{\sum_{\ell \in \mathcal{L}_m} \pi_{\ell y} \mu_\ell(\boldsymbol{x}|\Theta)}{\mathbb{P}_T[y|\boldsymbol{x}, \Theta, \boldsymbol{\pi}]} \,,$$

where $\mathcal{L}_m \subseteq \mathcal{L}$ denotes the set of leaves held by the subtree rooted in node m. The actual computation of the gradient term in Eq. (9) can be carried out by traversing the tree twice [Kontschieder *et al.*, 2015].

Learning prediction nodes. We consider now the minimization of (5) with respect to $\boldsymbol{\pi}$ when Θ is fixed, *i.e.*

$$\min_{\boldsymbol{\pi}} R(\Theta, \boldsymbol{\pi}; \mathcal{T}) \,. \quad (10)$$

This is a *convex* optimization problem and a global solution can be easily recovered. A similar problem has been encountered in the context of decision trees in [Rota Bulò and Kontschieder, 2014], but only at the level of a single node. In our case, however, the whole tree is taken into account, and we are jointly estimating *all* the leaf predictions.

In order to compute a global minimizer of (10) we propose the following iterative scheme:

$$\pi_{\ell y}^{(t+1)} = \frac{1}{Z_\ell^{(t)}} \sum_{(\boldsymbol{x}, y') \in \mathcal{T}} \frac{\mathbb{1}_{y=y'} \pi_{\ell y}^{(t)} \mu_\ell(\boldsymbol{x}|\Theta)}{\mathbb{P}_T[y|\boldsymbol{x}, \Theta, \boldsymbol{\pi}^{(t)}]} \,, \quad (11)$$

for all $\ell \in \mathcal{L}$ and $y \in \mathcal{Y}$, where $Z_\ell^{(t)}$ is a normalizing factor ensuring that $\sum_y \pi_{\ell y}^{(t+1)} = 1$. The starting point $\boldsymbol{\pi}^{(0)}$ can be arbitrary as long as every element is positive. A typical choice is to start from the uniform distribution in all leaves, *i.e.* $\pi_{\ell y}^{(0)} = |\mathcal{Y}|^{-1}$. It is interesting to note that the update rule in (11) is step-size free and it guarantees a strict decrease of the risk at each update until a fixed-point is reached [Kontschieder *et al.*, 2015].

Learning a forest. So far we have dealt with a single decision tree setting. Now, we consider an ensemble of trees $\mathcal{F}$, where all trees can possibly share the same parameters in Θ, but each tree can have a different structure with a different set of decision functions (still defined as in (3)), and independent leaf predictions $\boldsymbol{\pi}$. Since each tree in the forest $\mathcal{F}$ has its own set of leaf parameters $\boldsymbol{\pi}$, we can update the prediction nodes of each tree independently via the update rule (11), given the current estimate of Θ. As for Θ, instead, we randomly select a tree in $\mathcal{F}$ for each mini-batch and then we proceed with the SGD update in (7), assuming that its leaf nodes' parameters $\boldsymbol{\pi}$ are fixed. This strategy somewhat resembles the basic idea of Dropout [Srivastava *et al.*, 2014], where each SGD update is potentially applied to a different network topology, which is sampled according to a specific distribution. In addition, updating individual trees instead of the entire forest reduces the computational load during training. At test time we average over tree predictions to produce the final outcome,(4).

4 Deep Decision Nodes

We have defined decision functions d_n in terms of real-valued functions $f_n(\cdot; \Theta)$, which are not necessarily independent, but coupled through the shared parametrization Θ. Our intention is to endow the trees with feature learning capabilities by embedding functions f_n within a deep neural network with parameters Θ. To this end, we regard each function f_n as a linear output unit of a deep network that will be turned into a probabilistic routing decision by the action of d_n, which applies a sigmoid activation to obtain a response in the $[0, 1]$ range. Fig. 2 provides a schematic illustration of this idea. The number of split nodes is determined by the number of output nodes of the preceding fully-connected layer. Under the proposed construction, the output units of the deep network are therefore not directly delivering the final predictions, *e.g.* through a Softmax layer, but each unit is responsible for driving the decision of a node in the forest. Indeed, during the forward pass through the deep network, a data sample $\boldsymbol{x}$ produces soft activations of the routing decisions of the tree that induce via the routing function a mixture of leaf predictions as per (1), which will form the final output.

5 Experiments on ImageNet

ImageNet [Russakovsky *et al.*, 2014] is a benchmark for large-scale image recognition tasks and its images are assigned to one out of 1000 possible ground truth labels. The dataset contains ≈1.2M training images, 50.000 validation images and 100.000 test images with average dimensionality of 482x415 pixels. Training and validation data are publicly available and we followed the commonly agreed protocol by reporting *Top5-Errors* on validation data. The GoogLeNet architecture [Szegedy *et al.*, 2014], which has a reported Top5-Error of 10.07% when used in a single-model, single-crop setting (see first row in Tab. 3 in [Szegedy *et al.*, 2014]), served as basis for our experiments. As opposed to conventional architectures, GoogLeNet uses 3 Softmax layers at different stages of the network to encourage the construction of informative features, due to its very deep architecture.

In order to obtain a *Deep Neural Decision Forest* architecture coined dNDF**.**NET, we have replaced each Softmax layer from GoogLeNet with a forest consisting of 10 trees (each fixed to depth 15), resulting in a total number of 30 trees. We refer to the individual forests as dNDF_0 (closest to raw input), dNDF_1 (replacing middle loss layer in GoogLeNet) and dNDF_2 (as terminal layer). Further details about our dNDF**.**NET architecture are provided in [Kontschieder *et al.*,

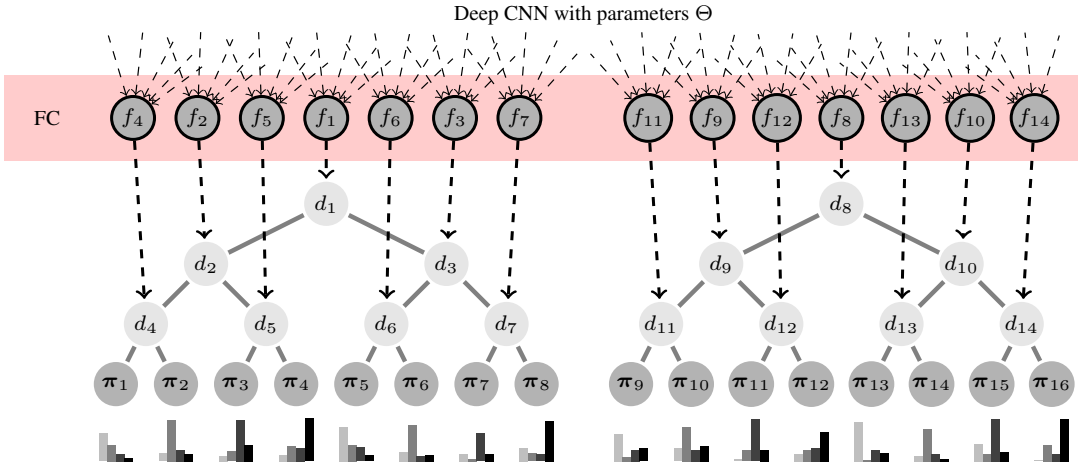

Figure 2: Illustration how to implement a deep neural decision forest (dNDF). Top: Deep neural network with variable number of layers, represented via parameters Θ. FC block: Fully Connected layer used to provide functions $f_n(\cdot;\Theta)$ (here: inner products), described in Eq. (3). Each output of f_n is brought in correspondence with a split node in a tree, eventually producing the routing (split) decisions $d_n(\boldsymbol{x}) = \sigma(f_n(\boldsymbol{x}))$. The order of the assignments of output units to decision nodes can be arbitrary (the one we show allows a simple visualization). The circles at the bottom correspond to leaf nodes, holding probability distributions $\boldsymbol{\pi}_\ell$ as a result of solving the convex optimization problem defined in Eq. (10).

	GoogLeNet [Szegedy *et al.*, 2014]						dNDF**.**NET			dNDF_0	dNDF_1	dNDF_2
#models/#crops	1/1	1/10	1/144	7/1	7/10	7/144	1/1	1/10	7/1	1/1		
Top5-Errors	10.07%	9.15%	7.89%	8.09%	7.62%	6.67%	7.84%	7.08%	**6.38%**	9.26%	8.49%	7.92%

Table 1: Top5-Errors obtained on ImageNet validation data, comparing our dNDF**.**NET to GoogLeNet.

2015]. Following the implementation guideline for decision nodes in Section 4, we randomly selected 500 output dimensions of the respectively preceding layers in GoogLeNet for each decision function f_n. We trained the network for 1000 epochs using (mini-) batches composed of 100.000 images (which was feasible due to distribution of the computational load to a cluster of 52 CPUs and 12 hosts, where each host is equipped with a NVIDIA Tesla K40 GPU).

As for the posterior learning, we only update the leaf node predictions of the tree that also receives split node parameter updates, *i.e.* the randomly selected one as described in Section 3. To improve computational efficiency, we consider only the samples of the current mini-batch for posterior learning, while *all* the training data could be used in principle. However, since we use mini-batches composed of 100.000 samples, we can approximate the training set sufficiently well and, at the same time, introduce a positive, regularizing effect.

Tab. 1 provides a summary of Top5-Errors on validation data for our proposed dNDF**.**NET against GoogLeNet. We ascribe the improvements on the single crop, single model setting (Top5-Error of only 7.84%) to our proposed approach, as the only architectural difference to GoogLeNet is deploying our dNDFs. By using an ensemble of 7 dNDF**.**NETs (still single crop inputs), we can reduce the error further and obtain a Top5-Error of 6.38%, which is better than the best result of 6.67%, obtained with 7 GoogLeNets using 144 crops per image [Szegedy *et al.*, 2014]. In the last three columns of Tab. 1, we report also the performance of each individual forest dNDF_x ($x = 0, 1, 2$) within the architecture. As expected, dNDF_0 (closest to the input layer) performs worse than dNDF_2, which is the final layer of dNDF**.**NET, but only by 1.34%. Averaging over all three dNDF forest outputs yields the lowest Top5-Error of 7.84%.

6 Conclusions

In this paper we have shown how to model and train stochastic, differentiable decision trees, usable as alternative classifiers for end-to-end learning in (deep) convolutional networks. Prevailing approaches for decision tree training typically operate in a greedy and local manner, making representation learning impossible. To overcome this problem, we introduced stochastic routing for decision trees, enabling split node parameter learning via back-propagation. Moreover, we showed how to populate leaf nodes with their optimal predictors, given the current state of the tree/underlying network. We have successfully validated our new decision forest model on ImageNet and surpassed state-of-the-art when integrating it in the GoogLeNet architecture, without any form of dataset augmentation, further detailed in [Kontschieder *et al.*, 2015].

Acknowledgments Peter and Samuel were partially supported by Novartis Pharmaceuticals and the ASSESS MS project. Madalina was partially supported by NSH Award 1320347, DARPA Contract FA8750-12-2-0324 and the Mobilize Center (NIH U54 EB020405).

References

[Amit and Geman, 1997] Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *(NC)*, 9(7):1545–1588, 1997.

[Bengio *et al.*, 2010] Yoshua Bengio, Olivier Delalleau, and Clarence Simard. Decision trees do not generalize to new variations. *Computational Intelligence*, 26(4):449–467, 2010.

[Bosch *et al.*, 2007] A. Bosch, A. Zisserman, and X. Muñoz. Image classification using random forests and ferns. In *(ICCV)*, 2007.

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

[Brostow *et al.*, 2008] Gabriel J. Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. Segmentation and recognition using structure from motion point clouds. In *(ECCV)*. Springer, 2008.

[Caruana *et al.*, 2008] R. Caruana, N. Karampatziakis, and A. Yessenalina. An empirical evaluation of supervised learning in high dimensions. In *(ICML)*, pages 96–103, 2008.

[Criminisi and Shotton, 2013] A. Criminisi and J. Shotton. *Decision Forests in Computer Vision and Medical Image Analysis*. Springer, 2013.

[Fei-Fei, 2015] Andrej Karpathy Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *(CVPR)*, 2015.

[Hastie *et al.*, 2009] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, 2009.

[He *et al.*, 2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015.

[Heath *et al.*, 1993] David Heath, Simon Kasif, and Steven Salzberg. Induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2(2):1–32, 1993.

[Kontschieder *et al.*, 2013] P. Kontschieder, P. Kohli, J. Shotton, and A. Criminisi. GeoF: Geodesic forests for learning coupled predictors. In *(CVPR)*, pages 65–72, 2013.

[Kontschieder *et al.*, 2015] P. Kontschieder, M. Fiterau, A. Criminisi, and S. Rota Bulò. Deep neural decision forests. In *(ICCV)*, 2015.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. Imagenet classification with deep convolutional neural networks. In *(NIPS)*, 2012.

[Lin *et al.*, 2013] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *CoRR*, abs/1312.4400, 2013.

[Montillo *et al.*, 2013] A. Montillo, J. Tu, J. Shotton, J. Winn, J. E. Iglesias, D. N. Metaxas, and A. Criminisi. Entangled forests and differentiable information gain maximization. In *Decision Forests in Computer Vision and Medical Image Analysis*. Springer, 2013.

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

[Russakovsky *et al.*, 2014] 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 (IJCV)*, 2014.

[Shotton *et al.*, 2013] Jamie Shotton, Ross Girshick, Andrew Fitzgibbon, Toby Sharp, Mat Cook, Mark Finocchio, Richard Moore, Pushmeet Kohli, Antonio Criminisi, Alex Kipman, and Andrew Blake. Efficient human pose estimation from single depth images. *(PAMI)*, 2013.

[Srivastava *et al.*, 2014] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[Szegedy *et al.*, 2014] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[Yu and Deng, 2014] D. Yu and L. Deng. *Automatic Speech Recognition: A Deep Learning Approach*. Springer, 2014.