This CVPR2014 paper is the Open Access version, provided by the Computer Vision Foundation.
The authoritative version of this paper is available in IEEE Xplore.

Neural Decision Forests for Semantic Image Labelling

Samuel Rota Bulò
Fondazione Bruno Kessler
Trento, Italy
rotabulo@fbk.eu

Peter Kontschieder
Microsoft Research
Cambridge, UK
pekontsc@microsoft.com

Abstract

In this work we present Neural Decision Forests, a novel approach to jointly tackle data representation- and discriminative learning within randomized decision trees. Recent advances of deep learning architectures demonstrate the power of embedding representation learning within the classifier – An idea that is intuitively supported by the hierarchical nature of the decision forest model where the input space is typically left unchanged during training and testing. We bridge this gap by introducing randomized Multi-Layer Perceptrons (rMLP) as new split nodes which are capable of learning non-linear, data-specific representations and taking advantage of them by finding optimal predictions for the emerging child nodes. To prevent overfitting, we i) randomly select the image data fed to the input layer, ii) automatically adapt the rMLP topology to meet the complexity of the data arriving at the node and iii) introduce an ℓ_1-norm based regularization that additionally sparsifies the network. The key findings in our experiments on three different semantic image labelling datasets are consistently improved results and significantly compressed trees compared to conventional classification trees.

1. Introduction

Machine learning is one of the strongest driving forces behind modern computer vision systems, giving rise to impressive results on practical tasks like image classification [22] and body part recognition [29]. One of the prerequisites for making such systems work is the availability of large and accurately labelled training data sets. Of similar importance is how to optimally abstract the data, *i.e*. to choose the right *data representation* for the task at hand. The rationale behind finding ideal representations is to make the data more distinguishable which in turn should facilitate its handling in subsequent learning stages. In the computer vision community, much effort has gone into the careful design of appropriate representations (*aka* features), *i.e*. SIFT [24], HOG [11] or Shape Context [1] are examples of ingeniously engineered representations, exploiting prior knowledge about the tasks to be accomplished.

The desire to reduce the dependency on properly designed, hand-crafted features has focussed the attention of the machine learning community on *representation learning* (see [3] for a comprehensive introduction). In particular, recent successes in the field of deep learning [22, 9] confirm the motive for joint learning of classifiers and the representations they operate on. In such architectures, multiple sequences of non-linear processing stages generate data representation hierarchies, that are ultimately able to describe highly complex compositions in the data. However, deep learning architectures require substantial experience for hyper-parameter tuning as *e.g*. highlighted in [6, 2].

In this paper we investigate how to deploy representation learning within the conceptually simple framework of decision forests [27, 7, 10]. Decision forests are ensembles of binary decision trees that have become very popular in computer vision due to their efficiency, flexibility, good generalization capability and inherent ability to handle multi-class problems. They have been applied to various tasks including semantic segmentation [30, 20], object detection [15, 20], and edge detection [12]. However, decision forests have not yet been adopted to account for representation learning in the previously introduced sense. Typically, their original input data representations are left unchanged although recent works [25, 19] have considered enriching the input space with intermediate predictions of the input data, which can be regarded as representation learning in a more general manner. In our opinion, this results in a suboptimal way of exploiting the hierarchical nature of decision trees which naturally allows learning cascaded representations of the data, as we will show in the remainder of this work.

We introduce Neural Decision Forests (NDF) – A novel perspective on classification trees for joint learning of data representations and the decisions taken upon them. The core of our method is a novel, *randomized Multi-Layer Perceptron* (rMLP) that we deploy as substitute for the conventional *split* (or interior) nodes of the trees. Our proposed

Figure 1. Example input RGB image and learned representations of our rMLP taken from a hidden layer, visualized using heat-maps. Please note the diverse responses in different areas of the image.

rMLP allows us to jointly learn i) new (and possibly non-linear) data representations by means of its hidden layers, based on the discriminatively routed data it is reached by and ii) optimal predictions for the emerging left and right child nodes to which the output of our rMLP routes the data of the parent in a soft way, respectively. An illustration in Fig. 1 shows an input RGB image with heat map visualizations of four obtained representations, automatically learned by our rMLP. In connection to the rMLP we provide a probabilistic model to describe the splitting process and design a new split function quality measure, which also guides the optimization of the network parameters.

Standard MLPs show a strong tendency to overfit to data [4], consequently giving poor generalization accuracy and therefore low performance on unseen test data. We prevent this effect by taking a number of countermeasures. First, the topology of our rMLP is determined by the distribution of the labels arriving at a node: we impose a higher complexity when many different classes are present. Second, we apply a randomized selection step for choosing the signals to the input layer of the rMLP. Therefore, we have no fixed 'wiring' of image pixel positions to the network, which drastically reduces the number of parameters to be learned but also allows us to interpret our rMLP as representation generator that learns weights for non-local kernels - A desideratum that was highlighted in [3] to exploit the principle of non-local generalization. Finally, we introduce an ℓ_1-norm based regularization strategy to further escape the risk of overfitting and to obtain more concise networks with sparsified connections.

Experiments on semantic segmentation show significant improvements over standard decision forests but also demonstrate comparable or better results to forests trained with more complex data representations. Another impressive property of our NDF is that our trained trees consist only of a fraction of nodes compared to standard forests, *i.e.* we obtain compression rates of up to a factor of 50 in our experiments. These findings encourage and support our initial claim to exploit the hierarchical nature of decision trees to provide a principled and joint approach of representation and discriminative learning.

Related Works. We focus on related approaches altering the split models in decision trees as well as previous works aiming to combine them with simple neural perceptrons. Classification trees are typically restricted to use axis-aligned split models (selecting only one or two dimensions of the input space where decisions are based on, also known as decision stumps [34]), motivated by the fact that a hierarchy of such splits is able produce non-linear decisions and is fast to evaluate. In turn, using generally oriented hyperplanes [5] (also introduced as Linear Combination-, multivariate decision-, oblique- [17] or perceptron decision trees [33, 16]) or quadratic surface models [10] are able to capture more complex data. Other works [18, 25] investigated the use of differentiable, softened linear discriminant functions in the interior nodes. More complex split models were also using boosting [31, 37] or support vector machines (SVM) in the splits [36], relying on input representations provided from a pre-processing pipeline. Despite favorable properties for modeling complex training data, they may be negatively affecting the generalization accuracy [10] and cannot generate new data representations. Early works on hybrid models of (multi-layer) perceptrons and decision trees (as those in [33, 16]), lack to address problems of overfitting or principled ways to jointly learn splits and resulting child posteriors and therefore cannot be applied to computer vision tasks in a straightforward way.

2. Neural Decision Forests

A *Neural Decision Forest* (NDF) is an ensemble of neural decision trees, where split functions in each node are *randomized Multi-Layer Perceptrons* (rMLP). This change introduces significant, positive impacts on the final classifier as already mentioned in the introduction. The inference procedure in NDF, which is described in the next section, substantially coincides with the one of RF. The learning phase however, is different and will be addressed in Sect. 3.

2.1. Inference

Inference in NDF takes place like in RF, except for the way the split function is computed. Let $\mathcal{X}$ and $\mathcal{Y}$ denote the input and output space of a classification task. Each tree $t \in \mathcal{F}$ of a NDF $\mathcal{F}$ classifies a sample $\boldsymbol{x} \in \mathcal{X}$ by routing it from the root to a leaf node, where the actual classification takes place. Each *leaf* is associated with a probability distribution defined over $\mathcal{Y}$. Decisions about the routing of samples are taken in the internal nodes, where a *neural split function* $\psi : \mathcal{X} \rightarrow \{\mathtt{L}, \mathtt{R}\}$ (see, Sect. 2.2) is evaluated to decide

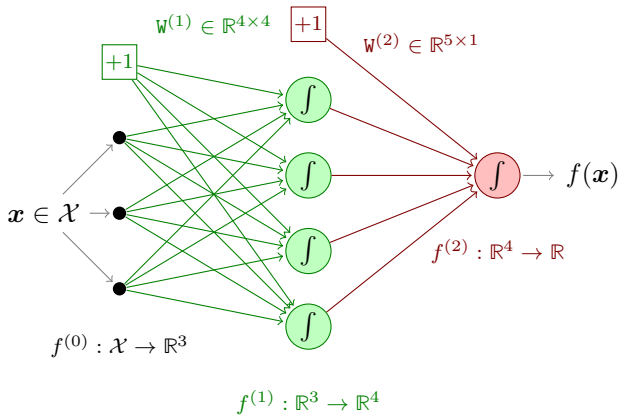

Figure 2. Example of multi-layer perceptron with 1 hidden layer.

whether a sample $\boldsymbol{x}$ has to be forwarded to the left child ($\psi(\boldsymbol{x}) = \mathtt{L}$) or to the right one ($\psi(\boldsymbol{x}) = \mathtt{R}$).

Finally, the classification at the forest level is obtained by combining the predictions delivered by the single trees. If $\mathbb{P}[y|\boldsymbol{x}, t]$ is the probability of sample $\boldsymbol{x}$ to take on class y according to tree $t \in \mathcal{F}$, then the same probability according to the whole forest is given by

$$\mathbb{P}[y|\boldsymbol{x}, \mathcal{F}] = \frac{1}{|\mathcal{F}|} \sum_{t \in \mathcal{F}} \mathbb{P}[y|\boldsymbol{x}, t] \,. \tag{1}$$

2.2. Neural Split Function

A split function in NDF is a randomized multi-layer perceptron consisting of an *input layer* that will be connected to the samples reaching the node, k *hidden layers* and a single *output unit* (see Fig. 2 for an example MLP with 1 hidden layer). We regard the jth neuron of the ith layer ($i > 0$) of the network as a function $f^{(ij)}(\cdot; \boldsymbol{w}^{(ij)}) : \mathbb{R}^{\mathsf{m}_{i-1}} \to \mathbb{R}$. We assume the neural unit to be fully connected to the $(i-1)$th layer consisting of m_{i-1} units and to an input unit clamped at $+1$ (or bias term). Here, $\boldsymbol{w}^{(ij)}$ denotes the $(1 + \mathsf{m}_{i-1})$-dimensional vector of synaptic weights of the neuron and we assume the first entry $w_0^{(ij)}$ to be the synaptic weight of the connection to the $+1$ unit. Each unit is a linear model composed with a non-linear activation function σ, *i.e.* $f^{(ij)}(\boldsymbol{x}; \boldsymbol{w}^{(ij)}) = \sigma\left({\boldsymbol{w}^{(ij)}}^\top \tilde{\boldsymbol{x}}\right)$ where $\tilde{\boldsymbol{x}} = \begin{bmatrix} 1 & \boldsymbol{x}^\top \end{bmatrix}^\top$. Similarly, we abstract the ith neural layer ($i > 0$) consisting of m_i units as a vector-valued function $f^{(i)}(\cdot; \mathtt{W}^{(i)}) : \mathbb{R}^{\mathsf{m}_{i-1}} \to \mathbb{R}^{\mathsf{m}_i}$ defined as

$$f^{(i)}(\boldsymbol{x}; \mathtt{W}^{(i)}) = \begin{bmatrix} f^{(i1)}\left(\boldsymbol{x}; \boldsymbol{w}^{(ij)}\right) \\ \vdots \\ f^{(i\,\mathsf{m}_i)}\left(\boldsymbol{x}; \boldsymbol{w}^{(i\,\mathsf{m}_i)}\right) \end{bmatrix} . \tag{2}$$

$\mathtt{W}^{(i)} = [\boldsymbol{w}^{(i1)}, \ldots, \boldsymbol{w}^{(i\,\mathsf{m}_i)}]$ is a matrix holding the synaptic weights of each neuron of the ith layer, column-wise.

The input layer is represented as a vector-valued function $f^{(0)} : \mathcal{X} \to \mathbb{R}^{\mathsf{m}_0}$ which maps samples in $\mathcal{X}$ to m_0-dimensional feature vectors that will be fed to the subsequent layers of the network. We will discuss how we model the input layer for semantic image labelling in Sect. 4. The output layer indexed $\mathsf{k}+1$ is defined according to (2) and it consists of a singe neuron, *i.e.* $\mathsf{m}_{\mathsf{k}+1} = 1$.

The whole MLP network is defined as a function $f(\cdot; \Theta) : \mathcal{X} \to \mathbb{R}$ obtained as the composition of the layer functions $f^{(i)}$:

$$f(\cdot; \Theta) = f^{(\mathsf{k}+1)} \circ \cdots \circ f^{(0)} \,, \tag{3}$$

where Θ holds all the network's parameters and we wrote $f^{(i)}$ in place of $f^{(i)}(\cdot; \mathtt{W}^{(i)})$. For convenience, we introduce an auxiliary function $g^{(i)}$, which is defined inductively as

$$g^{(i)} = \begin{cases} f^{(i)} \circ g^{(i-1)} & \text{if } i > 0 \,, \\ f^{(0)} & \text{if } i = 0 \,. \end{cases}$$

The neural network can now be compactly expressed in terms of g as $f = g^{(\mathsf{k}+1)}$.

In the literature, different activation functions have been proposed. In this paper we will adopt the *logistic sigmoid function* defined as $\sigma(z) = [1 + \exp(-z)]^{-1}$, and we interpret the output of f as the probability that the input sample is routed left rather than right, *i.e.* $f(\boldsymbol{x}) \approx \mathbb{P}[\psi(\boldsymbol{x}) = \mathtt{L}]$. It will become useful later to set $f_{\mathtt{L}} = f$ and $f_{\mathtt{R}} = 1 - f$. Once the neural network is trained (as detailed in Sect. 3), the split function, which returns a hard decision, can be evaluated by thresholding the MLP's output, *i.e.* $\psi(\boldsymbol{x}) = \mathtt{L}$ if $f(\boldsymbol{x}; \Theta) \geq 0.5$ and $\psi(\boldsymbol{x}) = \mathtt{R}$, otherwise.

3. Learning in NDF

The standard approach to training a random decision tree of a RF consists in a recursive procedure that starts from the root and iteratively builds the tree by splitting the actual terminal nodes. During this process each sample of the training set is routed through the tree in such a way that the whole training set is partitioned across the terminal nodes. We call *node sample* a sample of the training set that reached a particular node of the tree. The decision whether a terminal node should be further split and how the splitting should be done is taken exclusively based on its node samples. The splitting condition depends typically on the depth of the node, the number of node samples or the entropy of the node sample class distribution. If the node splitting takes place, a binary decision function defined on the samples space (*split function*) is determined in a way to maximize some class purity measure such as *information gain* or *Gini impurity*, evaluated with respect to the node samples. This maximization is typically carried out in an approximate way by randomly creating a pool of split functions among which the best one in terms of class purity is

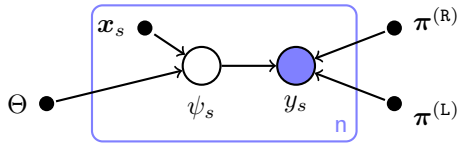

Figure 3. Graphical model for the splitting process. Sample $\boldsymbol{x}_s$ takes on the posterior probability in $\boldsymbol{\pi}^{(\mathtt{L})}$ and $\boldsymbol{\pi}^{(\mathtt{R})}$ according to the outcome of the split function ψ_s parametrized by Θ.

retained. When no node is further grown, the node sample class distributions are stored in the respective tree leaves.

We introduce the graphical model depicted in Fig. 3 to give a probabilistic perspective to the local decision that is taken in the RF at each terminal node to select the best split function and the best predictions for the children. For each node sample $\boldsymbol{x}_s$, $1 \leq s \leq \mathtt{n}$, we regard $\psi_s \in \{\mathtt{L}, \mathtt{R}\}$ as a split variable determining whether sample $\boldsymbol{x}_s$ should be routed left or right. This random variable follows a Bernoulli distribution having as parameter the output of a *routing function* $f(\boldsymbol{x}_s|\Theta) \in [0, 1]$, which depends on the sample and a set of parameters Θ. The routing function in standard RF comes from a random pool of candidate functions taking on binary values, *i.e.* $f(\boldsymbol{x}_s|\Theta) \in \{0, 1\}$, so that the split decision ψ_s has a deterministic outcome, and Θ could represent, *e.g.* its index within the pool. In the case of NDF, instead, $f(\boldsymbol{x}_s|\Theta)$ can be regarded as the output of the rMLP given sample $\boldsymbol{x}_s$, where Θ encodes the neural network's parameters. The class prediction of sample $\boldsymbol{x}_s$, modeled by variable y_s, depends on the outcome of the split decision ψ_s. Depending on whether sample $\boldsymbol{x}_s$ is routed left ($\psi_s = \mathtt{L}$) or right ($\psi_s = \mathtt{R}$), we assume y_s to be distributed as $\boldsymbol{\pi}^{(\mathtt{L})}$ or $\boldsymbol{\pi}^{(\mathtt{R})}$. In other terms, y_s can be regarded as a mixture of two categorical distributions parametrized by $\boldsymbol{\pi}^{(\mathtt{L})}$ and $\boldsymbol{\pi}^{(\mathtt{R})}$, respectively, where the mixing factor is determined by the splitting function ψ_s.

Now, given the graphical model in Fig. 3, we determine the quality $Q(\Theta)$ of a split function by evaluating the likelihood of the corresponding parameters Θ under the best possible choice of $\boldsymbol{\pi} = (\boldsymbol{\pi}^{(\mathtt{L})}, \boldsymbol{\pi}^{(\mathtt{R})})$, *i.e.*

$$Q(\Theta) = \max_{\boldsymbol{\pi}} \mathbb{P}[\boldsymbol{y}|\mathtt{X}, \boldsymbol{\pi}, \Theta] \,, \tag{4}$$

where $\boldsymbol{y} = (y_1, \ldots, y_{\mathtt{n}})$ denotes the vector of class labels for the node samples in $\mathtt{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{\mathtt{n}})$. The likelihood of the observed class labels under i.i.d. assumption is

$$\mathbb{P}[\boldsymbol{y}|\mathtt{X}, \boldsymbol{\pi}, \Theta] = \prod_{s=1}^{\mathtt{n}} \mathbb{P}[y_s|\boldsymbol{x}_s, \boldsymbol{\pi}, \Theta] \,, \tag{5}$$

where the per-sample likelihood term is given by

$$\begin{aligned} \mathbb{P}[y_s|\boldsymbol{x}_s, \boldsymbol{\pi}, \Theta] &= \sum_{d \in \{\mathtt{L},\mathtt{R}\}} \mathbb{P}[y_s|\psi_s = d, \boldsymbol{\pi}] \mathbb{P}[\psi_s = d|\boldsymbol{x}_s, \Theta] \\ &= \sum_{d \in \{\mathtt{L},\mathtt{R}\}} \pi_{y_s}^{(d)} f_d(\boldsymbol{x}_s|\Theta) \,. \end{aligned}$$

An optimal split function can finally be determined by finding a parametrization Θ that maximizes the quality measure in (4), which in turn corresponds to a maximum likelihood estimate. In the supplementary material, we show that our quality measure is substantially equivalent to the usual information gain criterion used to train RF only if the routing function is binary. However, the MLP-based routing function is non-binary and in this case information gain does not deliver an optimal decision in terms of log-loss. This is the reason why we propose a different quality measure. In Sect. 3.1 we will provide also an alternative to (4), which allows us to introduce priors and thus regularization terms in the model.

3.1. Regularization

The graphical model introduced in the previous section, describing the node splitting process, can be further generalized by imposing prior distributions on the model's parameters, namely the neural network's weights Θ and the child posteriors $\boldsymbol{\pi}$. By doing so, we can introduce additional, explicit regularization factors in our NDF. In this work, we introduce a centered Laplace prior $\mathbb{P}_{\text{Lap}}[\Theta|\lambda^{-1}]$ for the neural network's weights with scale (hyper-) parameter λ^{-1} in order to both regularize and sparsify the network's weights. As for the left and right child posteriors, a natural choice is typically a Dirichlet prior with (hyper-) parameter $\boldsymbol{\alpha}$, *i.e.* $\mathbb{P}[\boldsymbol{\pi}|\boldsymbol{\alpha}] = \mathbb{P}_{\text{Dir}}[\boldsymbol{\pi}^{(\mathtt{L})}|\boldsymbol{\alpha}]\mathbb{P}_{\text{Dir}}[\boldsymbol{\pi}^{(\mathtt{R})}|\boldsymbol{\alpha}]$. However, if on one hand having a regularization on the rMLP's weights is beneficial as we will show in the experimental section, having a simple Dirichlet prior for the child posteriors might have a negative impact on the results. This is due to an excessive dominance of the prior with respect to the likelihood term when few samples are considered [26]. Therefore, we decided to keep the Dirichlet's hyperparameter to 1, which coincides with a uniform prior distribution.

In order to account for the Laplace prior during the split function selection, we consider a posterior-based quality function as opposed to (4), which was likelihood-based:

$$Q_{\text{Reg}}(\Theta) = \max_{\boldsymbol{\pi}} \mathbb{P}[\boldsymbol{\pi}, \Theta|\boldsymbol{y}, \mathtt{X}, \lambda^{-1}] \,, \tag{6}$$

where the posterior probability is given by

$$\mathbb{P}[\boldsymbol{\pi}, \Theta|\boldsymbol{y}, \mathtt{X}, \lambda^{-1}] \propto \mathbb{P}[\boldsymbol{y}|\mathtt{X}, \boldsymbol{\pi}, \Theta]\mathbb{P}_{\text{Lap}}[\Theta|\lambda^{-1}] \,.$$

In line with the aforementioned arguments, the uniform prior on the child posteriors has been absorbed into the constant of proportionality. We can finally find the optimal Θ^* for the routing function by maximizing Q_{Reg}, *i.e.*

$$\Theta^* \in \arg\max_{\Theta} Q_{\text{Reg}}(\Theta) \,. \tag{7}$$

The maximization algorithm, starting from an initial configuration of the rMLP network, alternates between updating the child posteriors (see, Sect. 3.2) and the neural network's weights (see, Sect. 3.3).

3.2. Estimation of the Child Posteriors π

In this section we propose a multiplicative update rule for solving the concave maximization in (6). To our knowledge this is the first time that child posteriors are optimized in RF in place of taking the node sample's class distribution, which would be sub-optimal in our case as mentioned before, because we are in presence of non-binary routing functions in the nodes.

In order to compute a global solution to the maximization in (6), we make use of the following multiplicative rule:

$$\pi_c^{(d)} \leftarrow \frac{\pi_c^{(d)}}{Z^{(d)}} \sum_{s=1}^{\mathsf{n}} \frac{\mathbb{1}_{y_s=c}\, f_d^s}{\pi_c^{(\mathtt{L})} f_{\mathtt{L}}^s + \pi_c^{(\mathtt{R})} f_{\mathtt{R}}^s}\,, \tag{8}$$

where $d \in \{\mathtt{L}, \mathtt{R}\}$, $Z^{(d)}$ is the normalizing factor ensuring that $\boldsymbol{\pi}^{(d)}$ remains a probability distribution, and we abbreviated $f_d(\boldsymbol{x}_s|\Theta)$ with f_d^s. The rule in (8) is particularly interesting as it does not require a troublesome step-size to be specified, still guaranteeing a strict increase of the posterior probability at each update until a fixed-point is reached (see proof in supplementary material). The same update rule can be used for the maximization problem in (4) since we have imposed uniform prior on the child posteriors.

3.3. Estimation of the rMLP Parameters Θ

In order to find the optimal parametrization for the rMLP as per (7), we have to address a non-conventional neural network training, as we do not observe explicitly the desired output of the rMLP. Additionally, due to the ℓ_1-regularization induced by the Laplace prior, we have to cope with a non-differentiable objective. For the optimization we employ a ℓ_1-regularized version of the Resilient Back-Propagation (RProp) algorithm [28] due to its fast convergence properties and dynamic adaptation to the objective's surface. We omit the details of the RProp algorithm due to the limited space, but we report the gradient of the log-likelihood, which is the most important quantity to be computed during the optimization:

$$\frac{\partial \log \mathbb{P}[\boldsymbol{y}|\mathtt{X}, \boldsymbol{\pi}, \Theta]}{\partial \boldsymbol{w}^{(ij)}} = \sum_{s=1}^{\mathsf{n}} \frac{\partial g_s^{(i)}}{\partial \boldsymbol{w}^{(ij)}} \delta_s^{(i)}\,,$$

where $g_s^{(i)} = g^{(i)}(\boldsymbol{x}_s)$ and $\delta_s^{(i)}$ is inductively defined for all $i \geq 0$ as

$$\delta_s^{(i)} = \begin{cases} \dfrac{\partial g_s^{(i+1)}}{\partial g_s^{(i)}} \delta_s^{(i+1)} & \text{if } i \leq \mathsf{k}\,, \\ \dfrac{\pi_{y_s}^{(\mathtt{L})} - \pi_{y_s}^{(\mathtt{R})}}{\mathbb{P}[y_s|\boldsymbol{x}_s, \boldsymbol{\pi}, \Theta]} & \text{if } i = \mathsf{k}+1\,. \end{cases}$$

For the sake of completeness, we provide the formulas of the remaining gradient terms, *i.e.*

$$\frac{\partial g_s^{(i+1)}}{\partial g_s^{(i)}} = [\mathbf{0}|\mathtt{I}]\, \mathtt{W}^{(i+1)} \mathtt{G}_s^{(i+1)} \left(\mathtt{I} - \mathtt{G}_s^{(i+1)}\right)\,,$$

$$\frac{\partial g_s^{(i)}}{\partial \boldsymbol{w}^{(ij)}} = \tilde{g}_s^{(i-1)} \boldsymbol{e}^{j\,\top} \mathtt{G}_s^{(i)} \left(\mathtt{I} - \mathtt{G}_s^{(i)}\right)\,,$$

where $\mathtt{G}_s^{(i)}$ is a diagonal matrix with diagonal entries given by $g_s^{(i)}$, $\mathtt{I}$ is the identity matrix, $\mathbf{0}$ is a vector of zero entries and $\boldsymbol{e}^j$ is the jth column of the identity matrix.

The gradients can be easily computed by performing a forward-backward signal propagation, the forward and backward signals being given by $g_s^{(i)}$ and $\delta_s^{(i)}$ for each layer $i > 0$ and $s \in \{1, \ldots, \mathsf{n}\}$, respectively.

3.4. rMLP's Topology Selection

A thorny problem that we face at every node going to be split is the selection of the neural network's model complexity. The trees, as they develop, decompose an originally complicated classification task into many small, easier ones. For this reason, as a rule of thumb, we should focus on a complex MLP at the root of the tree and gradually take into account simpler models converging towards single perceptrons close to the leaves. Instead of reducing the model complexity as a function of the depth, one should consider rather the number of training samples reaching a node as well as their class distribution, since a tree could in general be highly unbalanced. In this work we opt for a rather simple but effective strategy, which uses only the information about the node sample's label distribution. We create a network having a number of layers proportional to the support size of the label distribution. Moreover, for each hidden layer we sample the number of neural units within a certain range. More details are given in the description of the experimental setting.

4. NDF for Semantic Image Labelling

Semantic image labelling is the problem of assigning a categorical label to each pixel in an image, (a per-pixel classification task) defined over images. A colour image I consists of $\mathsf{h} \times \mathsf{w}$ pixels, each being assigned a 3-dimensional RGB feature vector so the image can be represented in terms of a $(\mathsf{h} \times \mathsf{w} \times 3)$-dimensional tensor, *i.e.* $I \subseteq \mathbb{R}^{\mathsf{h}\times\mathsf{w}\times 3}$. A *sample* $\boldsymbol{x} \in \mathcal{X}$, which is fed to our NDF, is a triplet (u, v, I) identifying a pixel in position (u, v) in image I, while the output space $\mathcal{Y}$ consists in a finite set of categories to be assigned to each pixel.

In Sect. 3 we have presented our neural split function, but we have omitted how the input layer $f^{(0)}$ is actually implemented, as this depends on the type of application. In the context of semantic image labelling, we define the input layer of the rMLP at each internal tree node as a function of m_0 random *box-average* features. A box-average feature provides the mean intensity over an image area having a random size, located on a random position relative to the sample's center, and involving a random RGB channel.

Once the set of features has been determined, each sample can be expressed in terms of a m_0-dimensional vector $\phi(\boldsymbol{x}) \in \mathbb{R}^{\mathsf{m}_0}$. Before feeding the network with this input, we apply a normalizing linear transformation $h(\boldsymbol{z}) = \Sigma^{-\frac{1}{2}}(\boldsymbol{z} - \boldsymbol{\mu})$, where Σ and $\boldsymbol{\mu}$ are the covariance and mean of the box-average features extracted from the node samples used to train the rMLP (this is the variable decorrelation step in [23]). The input layer function can thus be defined as $f^{(0)} = h \circ \phi$.

In Fig. 4 we show an example using single-channel images for better illustration. Given a sample $\boldsymbol{x} = (u, v, I) \in \mathcal{X}$ to be fed to the rMLP, the mean intensities over the sampled areas are computed for image I, relative to the sample's center (u, v). To avoid making a one-shot guess when we determine the set of box-average features, we re-iterate the sampling process a fixed number of times and retain the set of features yielding the best quality according to (4) under a random weight initialization of the rMLP and binary step activation functions.

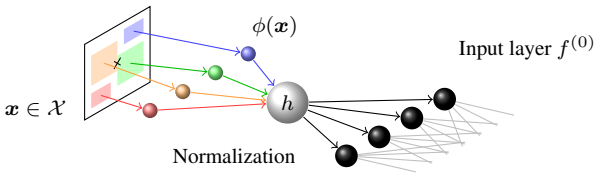

Figure 4. Example of how samples reaching a node of the NDF are mapped to the rMLP's input layer. For each sample $\boldsymbol{x} \in \mathcal{X}$ the mean values over a set of randomly positioned boxes are computed. They will then form the input to $f^{(0)}$ of the rMLP.

5. Experiments

In this section we report experimental results on the following three datasets: Etrims8 [21], CamVid [8] and Labelled Faces in the Wild [32]. In all our experiments we compare to a baseline of conventional classification forests (RF) and to scores reported in literature, where applicable. For all our provided methods, we train forests comprised of 8 trees where the inputs are either taken solely from raw channel RGB intensities or a stack of derived representations (Lab raw channels, first and second order derivatives of L-channel and HOG-like features). We fix the maximum probe offset radius to 30 pixels and only rescale images of the CamVid dataset by a factor of 0.5. For the RF forest, we use 300 node tests. For our NDF we compare the effect of 4 different variants when using RGB inputs: i) $\text{NDF}_{\mathtt{P}}$ is always restricted to a single perceptron, while the topology of the others is selected based on the distribution over labels in the data: ii) $\text{NDF}_{\mathtt{MLP}}$ uses either a single or two-layer perceptron, iii) $\text{NDF}_{\mathtt{MLPC}}$ is at most a 4 layer perceptron and iv) $\text{NDF}_{\mathtt{MLPC}-\ell 1}$ is at most a 4 layer perceptron with additional ℓ_1 regularization on the weights (fixing $\lambda = 1e^{-4}\mathsf{b}$ and $\mathsf{b} = 30000$ is the maximum number of samples per node we use to train the rMLP). For results on the derived representations we compare the baseline forest with $\text{NDF}_{\mathtt{MLPC}-\ell 1}$ only. We use the default parameters for RProp as given in [28]. The actual topology selection is driven by the support of the class label distribution π of the parent node, *i.e.* we gradually simplify the model with descending $\sum_c \mathbb{1}_{\pi_c > 0}$ in order not to overfit (which we found performing better than a selection based on the current depth of the node to be split). For $\text{NDF}_{\mathtt{MLPC}}$ and $\text{NDF}_{\mathtt{MLPC}-\ell 1}$ we use the 4-layer model if more than $2/3$rd of the classes are present, the 3-layer model if between $2/3$rd and $1/3$rd of them are present and the single perceptron instead. If applicable, the number of hidden units is also randomly selected and varies between 40-120% of the input layer size, which we fix to 5 probes per color channel. We report three types of scores, the percentage of all correctly classified pixels (**Global**), the class-average score (**Class-Avg**) defined as $\frac{\text{TP}}{\text{TP+FN}}$ and the Jaccard score (**Jaccard**), defined as $\frac{\text{TP}}{\text{TP+FP+FN}}$ (where TP, FP and FN are the number of true positives, false positives and false negatives, respectively).

5.1. Etrims8 Dataset

To assess the behaviour with respect to overfitting, we deliberately use this small dataset containing only 60 images that are annotated into 8 semantic classes and mostly show views of houses. We have constructed a random, 5-fold split of the data into 40 train and 20 test images and provide the results in the first three columns of Table 5.3.

On average and with RGB input only we improve by 5/3.5/4.3% on Global/Class-Avg/Jaccard scores when using $\text{NDF}_{\mathtt{P}}$, $\text{NDF}_{\mathtt{MLP}}$ or $\text{NDF}_{\mathtt{MLPC}}$, indicating that the more complex node split models cannot improve but instead start to overfit. However, when we evaluate $\text{NDF}_{\mathtt{MLPC}-\ell 1}$, we find improvements of 7.2/5.7/6.6%, indicating the efficacy of our regularization scheme. When comparing our best global score ($\text{NDF}_{\mathtt{MLPC}-\ell 1}\approx$71.7%) to [13] we find similar performance with their forest approach using color and contextual features (70.8%) or colour, context, geometric and image gradients (72.6%). Also, we are about on par with a SIFT-based approach, yielding $\approx 71.2\%$ and also reported in [13]. Using the derived representations (middle part of Table 5.3) we substantially improve over both, our baseline and previous state-of-the-art in [13, 14], using our $\text{NDF}_{\mathtt{MLPC}-\ell 1}$.

Another very appealing property of our method is that we can produce much more compressed trees compared to RF: The average number of nodes per tree in the forest gracefully decreases with more sophisticated node splits models. The average numbers of nodes per tree over all folds for this dataset are as follows: 42031 (RF), 7970 ($\text{NDF}_{\mathtt{P}}$), 3842 ($\text{NDF}_{\mathtt{MLP}}$), 2413 ($\text{NDF}_{\mathtt{MLPC}}$) and only 1997 for ($\text{NDF}_{\mathtt{MLPC}-\ell 1}$). We can see that with our proposed methods

we can achieve a compression up to a factor of 21. Another illustration for the quality of the resulting leaves is shown in the leftmost plot of Fig. 5, where we show average entropy vs. average cardinality of the leaf nodes. Clearly, the best models should reside on the top left of the plot which says that many samples are clustered in the leaves but also that the purity of the leaves is high. We find our proposed models at the desired locations with low entropy but high cardinality while the entropy of RF is low, but also the average leaf cardinality is fairly low. In other words, RF provides confident predictions which however stem from few samples while with our proposed NDF we can provide confident predictions based on the statistics of many samples.

5.2. CamVid Dataset

The CamVid dataset [8] contains images from driving videos (captured at daylight and at dusk) and is segmented into 32 semantic categories. We follow the standard protocol as in [8, 20, 35] and evaluate on the 11 most common categories, using a training/testing split of 367/233 images, respectively. We collected samples randomly and homogeneously for each class. The results are listed in the middle columns of Table 5.3, demonstrating again a considerate improvement over the baseline. We can also see that with RGB input only (top part) we can closely approach or even surpass the random-forest based scores reported in [8, 35, 19, 20], although they were using richer image representations. When using these derived image representations (middle part), we significantly outperform all comparable approaches without however applying an explicit spatial regularization as *e.g.* in [19, 19]. In terms of average nodes per tree in the forest we find the following development for RGB input: 328081 (RF), 102588 ($\text{NDF}_{\texttt{P}}$), 58153 ($\text{NDF}_{\texttt{MLP}}$), 42617 ($\text{NDF}_{\texttt{MLPC}}$) and 35992 for ($\text{NDF}_{\texttt{MLPC}-\ell 1}$). As can be seen, the compression rate is ≈ 10 and an impressive ratio of average leaf entropy vs. average leaf cardinality as shown in the middle plot of Fig. 5.

5.3. Labelled Faces in the Wild Dataset

In our final experiment, we take a random subset of 601 images from the Labelled Faces in the Wild dataset (LFW) [32] and added ground truth annotation for 8 classes (BACKGROUND, LEFT/RIGHT EYEBROW, LEFT/RIGHT EYE, NOSE, MOUTH AND FACE). We split the data into 500 training images and 101 test images and adjusted the data acquisition strategy to use all foreground samples but only as many background samples as the cardinality of the most dominant foreground class per image. With this setup, we obtain the scores listed in the final columns of Table 5.3. We can substantially improve over the baseline RF results with all our proposed variants of NDF using RGB inputs. The best performing approach in terms of Class-Avg/Jaccard is again $\text{NDF}_{\texttt{MLPC}-\ell 1}$, yielding impressive gains of 10.4/9.5% with RGB inputs and 18.5/18% with derived representations, respectively. Moreover, we find even stronger decrease in the overall number of nodes, on average over all trees: 501380 (RF), 58790 ($\text{NDF}_{\texttt{P}}$), 35021 ($\text{NDF}_{\texttt{MLP}}$), 21621 ($\text{NDF}_{\texttt{MLPC}}$) and 9456 ($\text{NDF}_{\texttt{MLPC}-\ell 1}$). Here, we can find a compression rate of ≈ 53, while still maintaining relatively low average entropy in the leaves (see the rightmost plot in Fig. 5).

6. Conclusions

In this work we have introduced Neural Decision Forests (NDF), which are a novel approach to jointly learn a classification model and the data representation it is based on, within the decision forest framework. We proposed randomized Multi-Layer Perceptrons (rMLP) to generate node-specific, possibly non-linear data representations but also act as routing instance for samples to the respective child nodes. We counteract overfitting by using randomized inputs, data-complexity dependent network topologies and an ℓ_1-norm based weight regularization in our rMLP. Moreover, we provide an alternating optimization scheme to jointly optimize over the child posterior distributions and the weights in the network. Experimental evaluations showed promising results when using raw data input and comparing to forests using well-designed and hand-crafted data representations. Using derived data representations as input, our approach yields new state-of-the-art scores on ETrims8 and CamVid datasets, compared to previous forest-based methods.

References

[1] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *(PAMI)*, 24:509–522, 2002.

[2] Y. Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade*, LNCS, pages 437–478. 2012.

[3] Y. Bengio, A. C. Courville, and P. Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR*, abs/1206.5538, 2012.

[4] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[5] A. Bosch, A. Zisserman, and X. Muñoz. Image classification using random forests and ferns. In *(ICCV)*, pages 1–8, 2007.

[6] L. Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, LNCS, pages 421–436. 2012.

[7] L. Breiman. Random forests. In *Machine Learning*, volume 45, pages 5–32, 2001.

[8] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla. Segmentation and recognition using structure from motion point clouds. In *(ECCV)*, pages 44–57. 2008.

[9] D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *(CVPR)*, 2012.

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

[11] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *(CVPR)*, 2005.

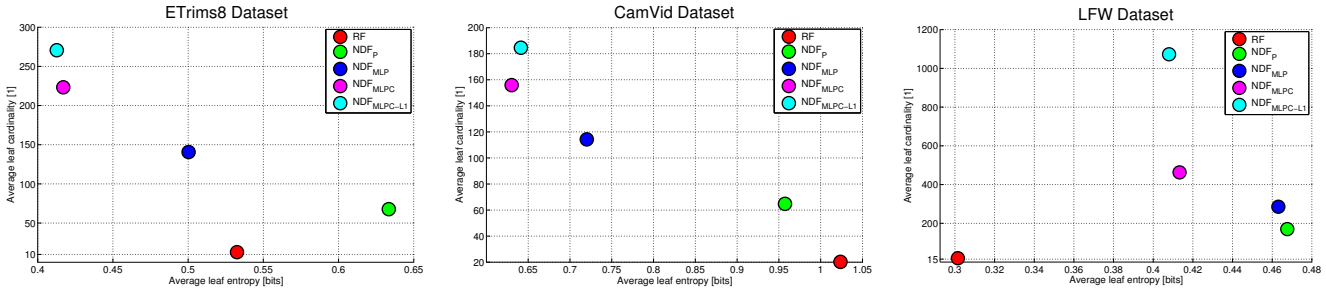

Figure 5. Average entropy vs. average cardinality of leaves for ETrims8, CamVid and LFW Datasets (left to right, best viewed in colour).

Method	ETRIMS8			CAMVID			LFW		
	Global	**Class-Avg**	**Jaccard**	**Global**	**Class-Avg**	**Jaccard**	**Global**	**Class-Avg**	**Jaccard**
RF Baseline	64.5 ±1.6	59.6 ±1.7	40.3 ±1.1	64.0	41.6	27.2	88.8	49.3	37.0
NDF_P	69.8 ±1.8	64.3 ±2.2	45.0 ±1.9	67.4	46.5	30.8	92.4	59.6	46.1
NDF_{MLP}	68.9 ±2.0	62.4 ±2.3	44.2 ±2.1	67.1	44.4	30.1	92.4	54.1	42.8
NDF_{MLPC}	69.7 ±1.7	62.5 ±2.1	44.7 ±1.9	67.4	44.2	30.2	92.6 (**+3.3**)	53.4	42.8
$NDF_{MLPC-\ell 1}$	71.7 ±2.0 (**+7.2**)	65.3 ±2.3 (**+5.7**)	46.9 ±2.0 (**+6.6**)	69.0 (**+5.0**)	46.8 (**+5.2**)	31.7 (**+4.5**)	91.8	59.7 (**+10.4**)	46.5 (**+9.5**)
RF Baseline	72.2 ±1.9	68.0 ±0.8	47.5 ±1.0	68.5	50.3	32.4	89.2	55.6	41.6
$NDF_{MLPC-\ell 1}$	80.8 ±0.7 (**+8.6**)	74.6 ±0.7 (**+6.6**)	56.9 ±1.2 (**+9.4**)	82.1 (**+13.6**)	56.1 (**+5.8**)	43.3 (**+10.9**)	95.4 (**+6.2**)	74.1 (**+18.5**)	59.6 (**+18.0**)
Best RF in [13]	76.1	72.3	-	-	-	-	-	-	-
Best in [14]	75.1	72.4	-	-	-	-	-	-	-
Best RF in [19]	-	-	-	-	-	38.3	-	-	-
Best RF in [20]	-	-	-	72.5	51.4	36.4	-	-	-
Best in [8]	-	-	-	69.1	53.0	-	-	-	-
Best in [35]	-	-	-	73.7	36.3	29.6	-	-	-

Table 1. Experimental results with comparisons to baselines and prior work, all numbers in [%]. Top: RGB input only. Middle: Derived representations. Bottom: Related works.

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

[13] B. Fröhlich, E. Rodner, and J. Denzler. Semantic segmentation with millions of features: Integrating multiple cues in a combined random forest approach. In *(ACCV)*, 2012.

[14] B. Fröhlich, E. Rodner, M. Kemmler, and J. Denzler. Large-scale gaussian process multi-class classification for semantic segmentation and facade recognition. *Machine Vision and Applications*, 24(5):1043–1053, 2013.

[15] J. Gall and V. Lempitsky. Class-specific hough forests for object detection. In *(CVPR)*, pages 1022–1029, 2009.

[16] H. Guo and S. B. Gelfand. Classification trees with neural network feature extraction. *IEEE Trans. on Neural Networks*, 1992.

[17] D. Heath, S. Kasif, and S. Salzberg. Induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2(2):1–32, 1993.

[18] G. H. John. Robust linear discriminant trees. In *Fifth Int. Workshop on Artificial Intelligence and Statistics*, 1995.

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

[20] P. Kontschieder, S. Rota Bulò, M. Pelillo, and H. Bischof. Structured labels in random forests for semantic labelling and object detection. *(PAMI)*, to appear, 2014.

[21] F. Korč and W. Förstner. eTRIMS Image Database for interpreting images of man-made scenes. Technical Report TR-IGG-P-2009-01, April 2009.

[22] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *(NIPS)*, 2012.

[23] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[24] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, pages 91–110, 2004.

[25] 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*. 2013.

[26] I. Nemenman, F. Shafee, and W. Bialek. Entropy and inference, revisited. In *Advances in Neural Inform. Process. Syst.*, 2003.

[27] J. R. Quinlan. Induction of decision trees. *(ML)*, pages 81–106, 1986.

[28] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *IEEE Conf. on Neural Networks*, 1993.

[29] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *(CVPR)*, pages 1297–1304, 2011.

[30] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *(CVPR)*, pages 1–8, 2008.

[31] Z. Tu. Probabilistic boosting-tree: Learning discriminative models for classification, recognition, and clustering. In *(ICCV)*, 2005.

[32] http://vis-www.cs.umass.edu/lfw.

[33] P. E. Utgoff. Perceptron trees: A case study in hybrid concept representations. In *Proceedings of AAAI*, 1988.

[34] P. Viola and M. Jones. Robust real-time object detection. *(IJCV)*, 2004.

[35] Y. Yang, Z. Li, L. Zhang, C. Murphy, J. Hoeve, and H. Jiang. Local label descriptor for example based semantic image labeling. In *(ECCV)*, volume 7578, pages 361–375. 2012.

[36] B. Yao, A. Khosla, and L. Fei-Fei. Combining randomization and discrimination for fine-grained image categorization. In *(CVPR)*, pages 1577–1584, 2011.

[37] P. Yin, A. Criminisi, J. Winn, and I. Essa. Tree-based classifiers for bilayer video segmentation. In *(CVPR)*, 2007.