Alternating Decision Forests

Samuel Schulter[1], Paul Wohlhart[1], Christian Leistner[2],
Amir Saffari[3], Peter M. Roth[1], Horst Bischof[1]

[1] Institute for Computer Graphics and Vision, Graz University of Technology, Austria

[2] Microsoft, Austria [3] Oxford Brookes University and Sony Computer Entertainment, UK

{schulter,wohlhart,pmroth,bischof}@icg.tugraz.at

christian.leistner@microsoft.com amir@ymer.org

Abstract

This paper introduces a novel classification method termed Alternating Decision Forests (ADFs)*, which formulates the training of Random Forests* explicitly *as a global loss minimization problem. During training, the losses are minimized via keeping an adaptive weight distribution over the training samples, similar to Boosting methods. In order to keep the method as flexible and general as possible, we adopt the principle of employing gradient descent in function space, which allows to minimize arbitrary losses. Contrary to Boosted Trees, in our method the loss minimization is an inherent part of the tree growing process, thus allowing to keep the benefits of common Random Forests, such as, parallel processing. We derive the new classifier and give a discussion and evaluation on standard machine learning data sets. Furthermore, we show how ADFs can be easily integrated into an object detection application. Compared to both, standard Random Forests and Boosted Trees, ADFs give better performance in our experiments, while yielding more compact models in terms of tree depth.*

1. Introduction

In recent years, Random Forests [1, 5] (RFs) have emerged as very useful classifiers for a large variety of computer vision tasks, including object recognition [16], semantic segmentation [30], or data clustering [28]. Although the method is relatively simple, it has many characteristics that make it particularly interesting for computer vision problems: (i) fast training and evaluation, (ii) robustness to label noise, (iii) inherent multi-class capability, (iv) suitability for parallel processing, and (v) good performance for high-dimensional input data [7]. Finally, RFs are able to yield state-of-the-art performance in various classification and regression tasks and compare favorably with other machine learning algorithms [9].

Interestingly and besides their simplicity, it is yet not fully theoretically understood what makes RFs such a powerful learning method. Existing explanations in the literature point to comparisons with nearest neighbor algorithms, rules of large numbers [5], classifier consistency [3], and large-margin methods [9], among others.

Besides these definitely valid insights, we argue that RFs share one important characteristic with other powerful classifiers like SVMs or Boosting. They all approximate Bayes Decision Rule – known to be the optimal classifier – via minimizing a margin-based loss function. However, in contrast to other methods, RFs minimize this loss greedily and implicitly via recursively reducing the uncertainty of given training samples by using independent base classifiers, *i.e.*, trees. Although these characteristics result in both, fast and parallel training capabilities, there is no control over an overall classifier loss and its proper minimization. While this makes it theoretically hard and somewhat unintuitive to comprehend the success of this learning method, it also unveils several practical disadvantages. First, during training there is no guarantee that all parameters have been learned properly by the entire model. Second, unnecessary emphasis is given on easy to classify training samples, often leading to too complex models. Third, it is hard to extend the learner to special learning tasks, such as, domain adaptation, semi-supervised or multiple instance learning, as this is usually realized via regularizing a global loss function.

In this paper, we propose a novel classifier termed *Alternating Decision Forests (ADFs)* that extends RFs by globally minimizing any given differentiable loss function, however, without losing the main characteristics and benefits of the original RF method as discussed above. We achieve this by borrowing ideas from Boosting methods, where during training a globally tracked weight distribution guides the loss minimization.

In more detail, like Boosting, ADFs assign a weight to each training sample. These weights are iteratively updated, *i.e.*, become higher for hard to classify samples and

lower for easy ones, respectively. While RFs are usually trained in a depth-first manner, ADFs are trained breadth-first. This has the main advantage that after having grown one forest stage, we can check the state of the entire model, *i.e.*, measure its performance against a global loss. To allow for incorporating arbitrary differentiable loss functions, we adopt the idea of employing gradient descent in function space [15] to calculate the weight updates. In order to let each stage of the classifier follow the adaptive weight distribution, we replace standard RFs splitting criteria with the weighted entropy, while features are still randomly subsampled. Thus, ADFs *alternate* between overall weight updates, and parallel randomized tree growing, where the latter one preserves the computational benefits of common Random Forests.

In our experiments on both, machine learning and image data, we show that ADFs yield better classification errors compared to standard Random Forests and Boosted Trees. Furthermore, we empirically demonstrate that ADFs give more compact models in terms of tree depth as the training of the decision trees is guided by the sample weights, *i.e.*, the global loss function.

2. Alternating Decision Forests

Before introducing and discussing Alternating Decision Forests, we shortly review standard Random Forests.

Random Forests [1, 5] are ensembles of T binary decision trees $\mathcal{T}_t(\mathbf{x}) : \mathcal{X} \rightarrow \mathcal{R}^K$, where $\mathcal{X} = \mathbb{R}^M$ is the M-dimensional feature space and $\mathcal{R}^K = [0,1]^K$ describes the space of class probability distributions over the label space $\mathcal{Y} = \{1, \dots, K\}$. During testing, each decision tree thus returns a class probability distribution $p_t(y|\mathbf{x})$ for a given test sample $\mathbf{x} \in \mathbb{R}^M$, and the final class label y^* is then obtained via averaging:

$$y^* = \arg\max_y \frac{1}{T} \sum_{t=1}^{T} p_t(y|\mathbf{x}) \ . \tag{1}$$

During training of a RF, the decision trees are provided with a random subset of the training data (*i.e.*, bagging [5]) and are trained independently from each other. Training a single decision tree involves recursively splitting each node such that the training data in the newly created child nodes is pure according to class labels. Each tree is grown until some stopping criterion, *e.g.*, the maximum tree depth, is reached and the class probability distributions are estimated in the leaf nodes.

A splitting function $s(\mathbf{x}; \Theta)$ is typically parameterized by two values: (i) a feature dimension $\Theta_1 \in \{1, \dots, M\}$ and (ii) a threshold $\Theta_2 \in \mathbb{R}$. The splitting function is then defined as

$$s(\mathbf{x}; \Theta) = \begin{cases} 0 & \text{if } \mathbf{x}(\Theta_1) < \Theta_2 \\ 1 & \text{otherwise} \end{cases} \ , \tag{2}$$

where the outcome defines to which child node the sample $\mathbf{x}$ is routed.

Each node chooses the best splitting function Θ^* out of a randomly sampled set $\{\Theta^i\}$ by optimizing

$$I = \frac{|L|}{|L|+|R|} H(L) + \frac{|R|}{|L|+|R|} H(R) \ , \tag{3}$$

where L and R are the sets of data samples that are routed to the left and right child nodes, according to $s(\mathbf{x}; \Theta^i)$; $H(S)$ is the local score of a set S of data samples (L or R), which is either the entropy or the Gini index [5]. Throughout this paper, we always use the entropy, which is defined as

$$H(S) = -\sum_{k=1}^{K} [p(k|S) \cdot \log(p(k|S))] \ , \tag{4}$$

where K is the number of classes, and $p(k|S)$ is the probability for class k, estimated from the set S.

2.1. RFs Approximate the Bayes Optimal Classifier

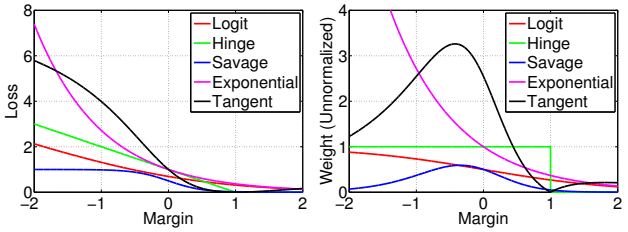
In the literature [4, 26], it is well known that the Bayes optimal classifier is the best classification technique. In practice, however, it is mostly infeasible to estimate such a classifier and one has to rely on approximations. Lin [23] showed that minimizing so called Fisher-consistent margin-based loss functions automatically leads to good approximations of the unknown Bayes decision rule. Zou *et al.* [31] extended this concept to multi-class problems. Prominent Fisher-consistent loss functions are, *e.g.*, hinge (SVM), exponential (Boosting) or logit (logistic regression), which deliver one reasonable explanation for the success of the corresponding learners. Due to lack of space, we refer the reader to the corresponding literature for more details.

As reviewed above, training in RFs is performed via recursively splitting training data into child nodes, where each split locally tries to optimize Eq. (3). As shown in [22], the local score $H(S)$ can generally be defined as

$$H(S) = \sum_{k=1}^{K} \left[p(k|S) \cdot l_{mm} \left(p(k|S) - \frac{1}{K} \right) \right] \ , \tag{5}$$

where $l_{mm}(\cdot)$ is a margin maximizing loss function. Hence, it is easy to see that minimizing the entropy is equivalent to minimizing the log likelihood, a typical Fisher-consistent loss. Thus, via greedily minimizing such a local score, a decision tree aims at minimizing a Fisher-consistent loss function, and is hence approximating Bayes optimal learners.

However, unlike other learners like Boosting or SVM approaches, RFs do not have an explicit global loss function. In turn, the nodes operate locally and trees are grown isolated, disregarding the current state of the entire classifier. While in practice this approach performs remarkably well, there is no global control over the entire ensemble, thus leading to sub-optimal approximations.



(a) (b)

Figure 1: Loss functions (a) and their corresponding weight update functions (b), both with respect to the margin.

2.2. Introducing a Global Loss

To allow for integrating different, global loss functions into the Random Forests training procedure, we adopt ideas from Boosting [29, 13]. A Boosting classifier $F(\mathbf{x})$ consists of T weak learners $f_t(\mathbf{x}) : \mathcal{X} \rightarrow \mathcal{R}^K$, where each weak learner gives a prediction $p_t(y|\mathbf{x})$ about the class confidences for a sample $\mathbf{x}$. The final output of a Boosting classifier is the weighted sum of the class confidences, such as, $F(\mathbf{x}) = \sum_{t=1}^{T} \nu_t f_t(\mathbf{x})$, where ν_t steers the influence of each individual weak learner.

The training procedure of Boosting runs in T iterations, where in each iteration a new classifier $f_t(\mathbf{x})$ is trained, its contribution rate ν_t determined, and both added to the model F. Boosting also keeps a weight distribution w_i^t for all training samples $\mathbf{x}_i$, $i = 1 \dots N$, in each iteration t. This weight distribution gets updated, such that "easy" samples are downweighted and "hard" samples are assigned higher weights, respectively. This allows the model to subsequently put its emphasis on hard samples (*i.e.*, those that have been misclassified).

There exist several different Boosting variants, most of them differing by the loss function they use, which in turn determines the shape of the weight distribution [17]. In Fig. 1a we illustrate prominent loss functions, such as, Logit, Hinge, Exponential, Savage [25], and Tangent [24]. While the first three are well known and often used in SVMs, Boosting (*e.g.*, AdaBoost [14]), or logistic regression, the last two are non-convex and more robust to outliers.

Friedman *et al.* [15] showed that Boosting can be understood as performing Gradient Descent in function space, paving the way for a new Boosting method named *GradientBoost*. In more detail, given a labeled training set $\{\mathbf{x}_i, y_i\}_{i=1}^N$, training a single weak learner can be written as the global loss minimization problem

$$\arg\min_{\Theta^t} \sum_{i=1}^{N} l(y_i; \mathcal{F}_{1:t-1}(\mathbf{x}_i; \bar{\Theta}) + f_t(\mathbf{x}_i; \Theta^t)) \ . \tag{6}$$

Here, $l(\cdot)$ is a differentiable loss function, $\mathcal{F}_{1:t-1}(\mathbf{x}; \bar{\Theta}) = \sum_{j=1}^{t-1} \nu \cdot f_j(\mathbf{x}; \Theta^j)$ describes the already trained classifier and $f_t(\mathbf{x}; \Theta^t)$ is the classifier in the current iteration t; $\bar{\Theta}$ is the collection of the parameters of the already fixed weak learners and Θ^t are the parameters to be trained in the current iteration; ν is the so-called shrinkage factor [17].

With a Taylor expansion, we can re-write Eq. (6) as

$$\arg\min_{\Theta^t} \sum_{i=1}^{N} l(y_i; \mathcal{F}_{1:t-1}(\mathbf{x}_i, \bar{\Theta})) - \frac{\partial\, l(y_i, \mathcal{F}_{1:t-1}(\mathbf{x}_i; \bar{\Theta}))}{\partial \mathcal{F}(\mathbf{x})} \cdot f_t(\mathbf{x}_i; \Theta^t) \ , \tag{7}$$

in order to learn the parameters Θ^t of the current weak classifier. As shown in [15], this can be done by training $f_t(\mathbf{x}; \Theta^t)$ to have high correlation with the negative gradient of the loss, which corresponds to updating the weights w_i^t for each training sample $\mathbf{x}_i$ in iteration t as

$$w_i^t = \left| \frac{\partial\, l(y_i, \mathcal{F}_{1:t-1}(\mathbf{x}_i; \bar{\Theta})}{\partial \mathcal{F}(\mathbf{x})} \right| \ . \tag{8}$$

GradientBoost has the advantage that any differentiable loss function can be used. This even allows for incorporating non-convex loss functions, which have been shown to be more robust to label noise [25]. For better understanding, in Fig. 1b, we illustrate the derivatives of frequently used losses. For more details we refer the interested reader to [17] and the references therein.

2.3. Training Alternating Decision Forests

In order to incorporate any differentiable, global loss function into Random Forests, we adopt the idea of the above reviewed Gradient Boosting method, *i.e.*, we perform Gradient Descent in function space. While we could easily train Boosted Trees that respect a global loss function, *i.e.*, Gradient Boosting having decision trees as weak learners, the goal of Alternating Decision Forests is to explicitly integrate the global loss into the tree growing scheme, thus preserving the parallel training of standard RFs. For that purpose, we need (i) a stage-wise tree growing scheme during which we update the weight distribution and (ii) splitting functions taking these weights into account.

To get a *stage-wise classifier*, we let our trees grow in an iterative, breadth-first manner, contrary to a typical depth-first scheme in standard RFs. Each stage in ADFs corresponds to a single depth of the forest, *i.e.*, the iterations are now indexed with $d = 1, \dots, D_{\text{max}}$, where D_{max} is the maximum tree depth of the forest. After training each single iteration d, we have a classification model $\mathcal{T}^d(\mathbf{x}_i) = \frac{1}{T} \sum_{t=1}^{T} p_t^d(y_i|\mathbf{x}_i)$, where $p_t^d(y_i|\mathbf{x}_i)$ is the class probability estimate of sample $\mathbf{x}_i$ returned by tree $\mathcal{T}_t^d$, which denotes tree $\mathcal{T}_t$ grown up to depth d. We can now use this strong classifier and a given loss $l(\cdot)$ to update the weights w_i^{d+1} of all training samples for the next iteration as illustrated in

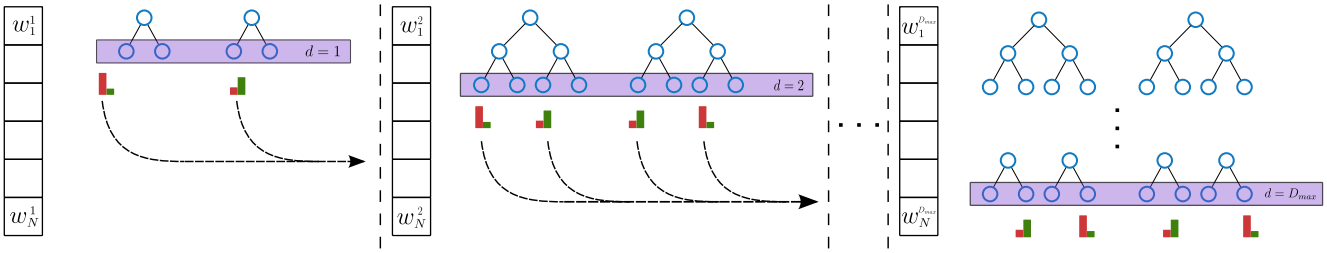


Figure 2: Overview of the proposed tree growing principle of *Alternating Decision Forests*. In the first iteration ($d = 1$), the weights w_i^d are uniform, and the first split functions are trained in a breadth-first manner. This forest with depth $d = 2$ can give predictions on the training samples, which are used to calculate weights, based on a global loss function, for the next iteration $d = 3$. This procedure is repeated until the maximum tree depth $d = D_{\text{max}}$ is reached.

Eq.(8). In the first iteration, all weights are set uniformly. After the weight updates, we train the next stage of the forest in parallel. This is done by converting all leaf nodes in the current iteration to split nodes.

To let the *splitting functions consider the sample weights*, we change the standard entropy calculation from Eq. (4) to become a weighted entropy by changing the estimation of the class distributions $p(k|S)$ for the set S:

$$p(k|S) = \frac{\sum_{i=1}^{|S|} \mathbb{1}[y_i = k] \cdot w_i^d}{\sum_{i=1}^{|S|} w_i^d} \,. \tag{9}$$

Here, $\mathbb{1}[y_i = k]$ is the indicator function returning 1 if the label y_i of the sample $\mathbf{x}_i \in S$ is equal to k, and 0 otherwise.

This process of *alternating* between training a single stage d and updating the weights w_i^{d+1} for the next stage is repeated until the same stopping criteria as in standard RFs are reached. Hence, we name this learning method *Alternating Decision Forests*. We give an illustrative overview of this scheme in Fig. 2 and summarize it in Alg. 1. Furthermore, we note that inference in ADFs is exactly the same as in RFs, *i.e.*, ADFs also inherit the properties of low computational costs during the testing phase from RFs. Thus, Alternating Decision Forests can incorporate well-defined losses into the training of Random Forests; however, without violating their most important benefits and characteristics.

Algorithm 1 ADF Training

Require: Labeled training set $\{\mathbf{x}_i, y_i\}_{i=1}^N \in \mathcal{X} \times \mathcal{Y}$
Require: Maximum tree depth D_{max}
1: Init weights $w_i^1 = \frac{1}{N}$
2: Init the root nodes
3: **for** d from 1 to D_{max} **do**
4: Check stopping criteria for all nodes in depth d
5: Split nodes in depth d; Eq. (3), Eq. (4), Eq. (9)
6: Update weights w_i^{d+1}; Eq. (8)
7: **end for**

Please note that, if each tree would correspond to one weak learner $f_t(\mathbf{x}; \Theta)$, each being fully trained in a single iteration, we would get a version of Boosted Trees [17]. In contrast, we train all trees in parallel, by switching from a depth-first to a breadth-first tree growing scheme. In detail, we start the training of our model by creating T trees. Then, we run D_{max} iterations, where in each iteration all trees of the forest are simultaneously grown deeper by one stage.

The weighted tree growing scheme in ADFs can be interpreted as a guided training of each single decision tree, but also as an explicit collaboration between all trees in the model. All nodes in the trees concentrate on "hard" training samples and, thus, don't waste effort on samples that are already learned relatively well by the entire model. However, unlike Boosted Trees, this global loss is now an inherent part of Random Forests. As we show in our experimental evaluations, this property of ADFs leads to more compact models in terms of tree depth.

For RFs, an upper bound on the generalization error GE can be defined as $GE \leq \rho \frac{1-s^2}{s^2}$, where ρ denotes the mean pair-wise correlation between trees and s the strength of individual trees [5]. A low generalization error thus requires strong but decorrelated trees. While ADFs explicitly enforce the collaboration of all trees in the forest, the decorrelation ρ can be preserved as in standard RFs, because (i) the splitting functions $s(\mathbf{x}; \Theta)$ are still drawn completely randomly and (ii) the set of samples falling in common nodes is different for each tree. This is important as the correlation ρ directly affects the generalization error.

3. Relation to Previous Work

In [12], Freund and Mason proposed a formulation to represent AdaBoost as a *single* decision tree. Although the method is termed Alternating Decision Tree or AD Tree, it is quite different to the algorithm presented in this paper. In more detail, an AD Tree consists of two different types of nodes, a split node and a predictor node, which are alternated during training. Several split nodes can be attached

to a single predictor node, thus allowing for multiple paths of a single example to traverse down the tree. This entity makes both training and inference rather slow.

In contrast to standard RFs, the individual trees of ADFs become interdependent or entangled during training. This is similar to the works of [27] and [20] that train decision trees breadth-first or according to a priority queue, in order to incorporate contextual features into the learning process. However, both works only consider the predictions within a single decision tree and use those predictions as additional features for the splitting functions in the next stage. In contrast, ADFs collect the information from the whole classifier, *i.e.*, all decision trees, and uses the predictions in order to minimize a global loss function. While it is beyond the scope of this paper, extending ADFs to incorporate such contextual information is straight-forward.

Another work that optimizes trees according to a differentiable loss function is that of Jancsary *et al.* [19], where the leaves of the trees store parameters for a Gaussian Conditional Random Field with different interaction types. Each factor type is only associated with a single tree but they are connected implicitly via the random field. In contrast, ADFs train a set of trees, which are connected via averaging over the predictions, *i.e.*, a generic Random Forest.

We further would like to point out the differences of ADFs to Boosted Trees [17] (BTs), *i.e.*, standard Boosting with decision trees as weak learners. As mentioned before, in BTs, the decision trees are trained sequentially and the weights are updated after training each single tree. This makes the whole training phase much slower than ADFs, as no parallelization is possible. Furthermore, during training a single decision tree in BTs, the weights cannot be updated. Contrary, in ADFs, it is exactly this property that allows for learning more compact models as the growing of each tree is somehow guided by the weight updates from the previous depth in the trees.

4. Machine Learning Experiments

Our experiments on machine learning benchmarks give a detailed analysis of the proposed classifier. First, we compare ADFs with the most related competing methods, *i.e.*, Random Forests (RFs) and Boosted Trees (BTs) on 5 different data sets. We also investigate different choices of the loss function. Then, we evaluate the influence of two important parameters common to ADFs, RFs, and BTs on the overall classification performance. Finally, we give a dense evaluation of different parameter choices for our classifier.

Data sets: We use 5 standard machine learning data sets to compare ADFs with related approaches and also to investigate different parameter settings. We use the *G50c* data set from [8], the *Letter* and *MNIST* data sets from [11], the *USPS* data set from [18], and the *Char74k* data set from [6]. The properties of these data sets are summarized in Tab. 1.

Dataset	# Train	# Test	# Features	# Classes
G50c	50	500	50	2
Letter	16000	4000	16	26
USPS	7291	2007	256	10
MNIST	60000	10000	784	10
Char74k	66707	7400	64	62

Table 1: Properties of the machine learning data sets used in our evaluation.

Experimental Setup: For a fair comparison between all three classifiers we set the common parameters to the same values. We thus set the number of trees $T = 100$ (for BT, this is equivalent to the number of weak learners), the maximum depth D_{max} of the trees to either 10, 15, or 25 (depending on the size of the training data of the data sets), the number of random splits per node to $\sqrt{M}$ [5], the number of random thresholds to 10 per node and the minimum number of samples for further splitting to 5. We always report the average error and standard deviation over several independent runs as all classifiers are non-deterministic; we have 250 runs for *G50c* and 5 runs for the remaining data sets, as they are much larger.

4.1. Comparison of ADFs with Other Classifiers

First, we directly compare all methods against each other on all 5 data sets. As the common parameters of ADFs, RFs and BTs are set equally, as mentioned before, we directly compare the way the tree structure is built. For ADFs and BTs we also evaluate 5 different loss functions that can be integrated in the Gradient Boosting formulation: Logit, Hinge, Exponential, Savage [25] and Tangent [24].

Tab. 2 depicts our results. As can be seen, the combination of ADFs and the Tangent loss function yields the best results on all 5 data sets. ADFs with Savage loss and Exponential loss are the second best choices on 3, respectively 1 data sets. Only for *G50c*, BTs with Savage loss gives the second best results. Most interesting, however, is the fact that ADFs (Tangent loss) constantly outperform both RFs and BTs, *i.e.*, the main competitors.

It is also worthwhile to investigate the influence of the different loss functions. The Logit, Hinge and Exponential losses are typically outperformed by the Savage and Tangent losses for both methods, ADFs and BTs. This indicates that the non-convexity of the loss, and thus the robustness to outliers (see Sec. 2 and Fig. 1a), plays a crucial role for the weight updates and thus the tree growing. For our further experiments, we fix the loss function to be the Tangent loss, as we can expect the best overall performance.

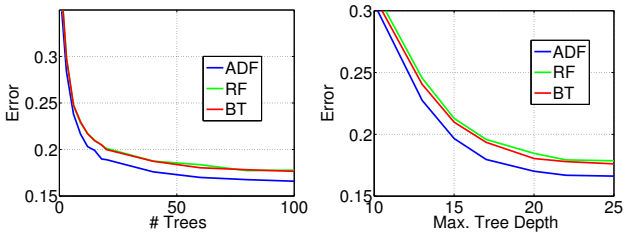
We further evaluate the computational costs of training ADFs and give a relative comparison to RFs and BTs in Tab. 3. As expected, BTs are much slower (around 7 times slower), while ADFs and RFs show similar training time.

Method	Loss	*G50c*	*Letter*	*USPS*	*MNIST*	*Char74k*
Alternating Decision Forests	Logit	19.83 ± 1.34	6.81 ± 0.34	6.31 ± 0.23	3.82 ± 0.08	17.38 ± 0.16
	Hinge	19.47 ± 1.26	4.89 ± 0.16	5.87 ± 0.19	3.20 ± 0.06	17.00 ± 0.10
	Exp	19.09 ± 1.17	4.27 ± 0.13	6.03 ± 0.29	2.96 ± 0.05	16.82 ± 0.15
	Savage [25]	19.00 ± 1.32	3.94 ± 0.14	5.76 ± 0.16	2.78 ± 0.09	16.92 ± 0.15
	Tangent [24]	$\mathbf{18.71 \pm 1.27}$	$\mathbf{3.52 \pm 0.12}$	$\mathbf{5.59 \pm 0.16}$	$\mathbf{2.71 \pm 0.10}$	$\mathbf{16.67 \pm 0.21}$
Boosted Trees [17]	Logit	18.99 ± 1.51	4.98 ± 0.09	5.99 ± 0.21	3.18 ± 0.08	17.98 ± 0.19
	Hinge	18.97 ± 1.33	4.87 ± 0.15	5.90 ± 0.15	3.23 ± 0.05	17.65 ± 0.19
	Exp	18.91 ± 1.30	4.78 ± 0.12	5.83 ± 0.19	3.17 ± 0.07	17.57 ± 0.10
	Savage [25]	18.87 ± 1.31	4.65 ± 0.12	5.92 ± 0.19	3.19 ± 0.07	17.62 ± 0.25
	Tangent [24]	18.90 ± 1.31	4.70 ± 0.18	5.93 ± 0.27	3.15 ± 0.05	17.59 ± 0.29
Random Forests [1, 5]	-	18.91 ± 1.33	4.75 ± 0.10	5.96 ± 0.21	3.21 ± 0.07	17.76 ± 0.13

Table 2: Alternating Decision Forests compared with the two main competitors, Random Forests and Boosted Trees, on 5 data sets. Best performing methods are highlighted and marked **bold-face**, second best methods are highlighted only.

Method	*G50c*	*Letter*	*USPS*	*MNIST*	*Char74k*
ADF	1.0	1.0	1.0	1.0	1.0
BT	3.99	6.55	7.32	7.05	7.09
RF	1.55	0.45	0.70	0.79	1.52

Table 3: Training time comparison between the three main competitors on all 5 data sets, relative to ADFs.



(a) (b)

Figure 3: Influence of (a) the number of trees T and (b) the maximum tree depth D_{max} on the three classifiers ADFs, RFs, and BTs on the *Char74k* data set.

4.2. Comparison of Common Parameters

Next, we evaluate the influence of two important parameters of all evaluated classifiers on the $Char74k$ data set. We investigate the number of trees T and the maximum tree depth D_{max}, which we vary in the ranges $[1, 100]$ and $[10, 25]$, respectively. As mentioned before, we choose the Tangent loss for both ADFs and BTs due to the good overall performance in the last experiment.

It is interesting that ADFs improve over RFs and BTs only for larger number of trees (see Fig. 3a). For a small number of trees, the performance is more or less the same. This could be explained by the weight updates in ADF, which follow the predictions of the current state of the classifier. However, due to the small number of trees and the fact that the trees are not fully grown yet, these predictions are unreliable. This might decrease the performance as the weights take unreasonable values. However, as soon as the number of trees increases, also the performance of ADFs increases and outperforms both RFs and BTs.

The behavior of the second parameter, *i.e.*, the maximum tree depth D_{max}, is different. Here, ADFs consistently outperform RFs and BTs (see Fig. 3b). This can be explained by the fact that the weight updates can rely on more stable predictions, as a larger number of trees is available. This experiment also reflects the guidance of the tree growing mentioned in Sec. 3, as both RFs and BTs need deeper trees to reach the performance of ADFs.

4.3. Parameter Evaluation

Finally, we also give a detailed evaluation of various parameter choices on the *Char74k* data set for our proposed method. Here, we compare several combinations of two different parameters – again, the number of trees T and the maximum tree depth D_{max}. We show our results in Tab. 4.

This evaluation confirms the properties of ADFs, inherited from RFs. The performance steadily increases with the number of trees and also the maximum depth of the trees. Parameter choices beyond those shown in the table do not further improve the results, which indicates that the performance saturates at some point, similar as for RFs and BTs. Tab. 4 also contains our overall best result of 16.18% error on the *Char74k* data set with 300 trees and a maximum depth of 25.

5. Alternating Decision Hough Forests

We now adopt our novel Alternating Decision Forests for one concrete computer vision application, namely for object detection with the popular Hough Forests (HFs) framework [16]. We denote our extension Alternating Decision Hough Forests (ADHF).

Tree Depth	Number of trees T						
$D_{\max}$	*1*	*10*	*25*	*50*	*100*	*200*	*300*
5	83.12 ± 0.68	53.75 ± 0.69	48.15 ± 0.58	46.34 ± 0.54	44.97 ± 0.29	44.20 ± 0.13	44.38 ± 0.14
10	55.63 ± 1.75	34.32 ± 0.35	32.17 ± 0.37	31.23 ± 0.19	30.60 ± 0.25	30.49 ± 0.08	30.40 ± 0.15
15	39.76 ± 1.18	24.64 ± 0.43	21.66 ± 0.38	20.38 ± 0.17	19.79 ± 0.20	19.61 ± 0.13	19.41 ± 0.14
20	36.07 ± 0.54	21.52 ± 0.32	18.83 ± 0.15	17.72 ± 0.12	17.09 ± 0.12	16.93 ± 0.11	16.72 ± 0.13
25	35.52 ± 0.41	21.17 ± 0.25	18.52 ± 0.21	17.16 ± 0.28	16.66 ± 0.22	16.38 ± 0.22	$\mathbf{16.18 \pm 0.13}$

Table 4: Different parameter choices of the number of trees T and the maximum tree depth $D_{\max}$ for Alternating Decision Forests on the *Char74k* data set.

HF is an object detection approach that describes an object by a set of 16×16 patches P_i, each storing several appearance feature channels $\Phi_c(P_i)$, one offset vector $\mathbf{d}_i$ that points to the object center, and a label y_i indicating that the patch belongs to an object. Negative patches are extracted from background regions and do not store an offset vector. Given those patches P_i, a Random Forest is trained with two different splitting criteria: (i) a standard classification criterion (see Sec. 2) and (ii) a regression criterion [16]. The former criterion tries to split patches according to the class label, the latter one according to the offset vectors (only for positive patches). The splitting function $s(P_i;\Theta)$ in HF is defined as

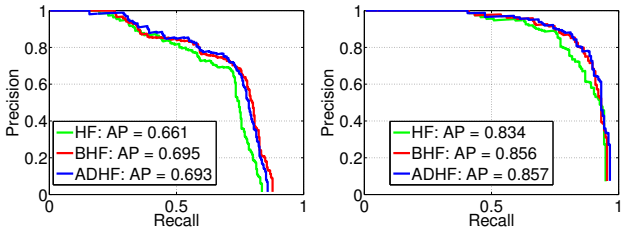
$$s(P_i;\Theta) = \begin{cases} 0 & \text{if } P_i^{\Theta_1}(\Theta_2) - P_i^{\Theta_1}(\Theta_3) < \Theta_4 \\ 1 & \text{otherwise} \end{cases}, \quad (10)$$

where Θ defines the splitting parameters. The feature channel is given by Θ_1, two pixel positions within the patch are defined by Θ_2 and Θ_3, and a random threshold is given by Θ_4. The resulting RF stores a foreground probability and a set of offset vectors in its leaf nodes. The two criteria ensure that the leafs are clean with respect to both, labels and offset vectors.

However, as in standard RFs, also Hough Forests only optimize the entropy on the node level, *i.e.*, locally, without regarding a global loss function. This results in the same disadvantages as for standard RFs. We thus also incorporate the minimization of a global loss function via iterative weight updates in the classification criterion in this framework. Each patch P_i is thus assigned a weight w_i, which is always updated after training a single stage of the classifier according to a given global loss function $l(\cdot)$, as described in Sec. 2.

The inference process of both, HFs and ADHFs, is equal and follows the generalized Hough voting scheme. Each patch in a test image votes for tentative object centers in a Hough image, where local maxima indicate detected objects [16].

Evaluation: We evaluate ADHF on two different data sets, *TUD-pedestrian* [2] and *ETHZ-cars* [21]. We compare with standard Hough Forests [16] (HFs), but also with Boosted Hough Forests (BHFs), *i.e.*, Boosting with a Hough Tree as weak learner, similar to Boosted Trees [17]. For both data sets, we extract a total number of 12000 patches and train a forest with 10 trees, a maximum depth of 15 and 5000 random tests per node. We also evaluate different loss functions for ADHF and BHF. For a fair comparison, we randomly extract the set of patches P_i once and let all methods, *i.e.*, ADHFs, BHFs, and HFs, use the same set for training. We thus prevent benefits for one method, just due to a better training set. Furthermore, we average our results over 5 independent runs because of the random nature of the methods.



(a) (b)

Figure 4: Precision-Recall curves of ADHF, BHF, and HF on (a) the *TUD-pedestrian* and (b) the *ETHZ-cars* data sets.

Interestingly, our evaluations revealed that different loss functions perform better on different data sets. While for *TUD-pedestrian* both non-convex and noise-robust loss functions, *i.e.*, Savage and Tangent, give the best results, for *ETHZ-cars* the Logit and Exponential losses outperform the others. This observation holds for both, ADHF and BHF. An intuitive explanation for this result is that the *ETHZ-cars* data set contains less noise than the *TUD-pedestrian* data set, as cars typically fit well in the bounding boxes, which is not true for pedestrians.

We present our results as precision-recall curves and average precision values [10] in Fig. 4. For ADHF and BHF, we show the curves of the best performing loss functions as presented above. As can be seen, ADHF and BHF can both outperform standard HF, while BHF take much more time to train (around 7 times), *cf.*, Tab. 3.

6. Conclusion

We proposed a novel classifier, termed *Alternating Decision Forest* (ADF), which illustrates how to formulate the Random Forest training as a global loss minimization problem. In contrast to the local optimization in standard Random Forests, we grow the trees stage-wise and include sample weight updates to minimize the global loss, as in Boosting algorithms. Our results on machine learning data confirm that the novel classifier inherits the benefits of Random Forests (fast training and testing, multi-class capability, *etc.*), while yielding *better results* and *more compact models* in terms of tree depth. We also demonstrate how this new formulation can be integrated in the popular Hough Forest framework for object detection, which yields better results on standard benchmark data sets compared to the original Hough Forests. We finally note that ADFs are relatively easy to implement and can replace standard RFs in any application without great efforts.

Acknowledgement: This work was supported by the Austrian Science Foundation (FWF) project Advanced Learning for Tracking and Detection in Medical Workow Analysis (I535-N23) and by the Austrian Research Promotion Agency (FFG) project FACTS (832045).

References

[1] Y. Amit and D. Geman. Shape Quantization and Recognition with Randomized Trees. *Neural Computation*, 9(7):1545–1588, 1997. 1, 2, 6

[2] M. Andriluka, S. Roth, and B. Schiele. People-Tracking-by-Detection and People-Detection-by-Tracking. In *CVPR*, 2008. 7

[3] G. Biau, L. Devroye, and G. Lugosi. Consistency of Random Forests and Other Averaging Classifiers. *JMLR*, 9(9):2015–2033, 2008. 1

[4] C. M. Bishop. *Pattern Recogntion and Machine Learning*. Springer, 2007. 2

[5] L. Breiman. Random Forests. *ML*, 45(1):5–32, 2001. 1, 2, 4, 5, 6

[6] T. E. d. Campo, B. R. Babu, and M. Varma. Character Recognition in Natural Images. In *VISAPP*, 2009. 5

[7] R. Caruana, N. Karampatziakis, and A. Yessenalina. An empirical Evaluation of Supervised Learning in High Dimensions. In *ICML*, 2008. 1

[8] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, 2006. 5

[9] A. Criminsi and J. Shotton. *Decision Forests for Computer Vision and Medical Image Analysis*. Springer, 2013. 1

[10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *IJCV*, 88(2):303–338, 2010. 7

[11] A. Frank and A. Asuncion. UCI Machine Learning Repository, 2010. University of California, Irvine, School of Information and Computer Sciences. Available online at http://archive.ics.uci.edu/ml. 5

[12] Y. Freund and L. Mason. The Alternating Decision Tree Learning Algorithm. In *ICML*, 1999. 4

[13] Y. Freund and R. E. Shapire. Experiments with a New Boosting Algorithm. In *ICML*, 1996. 3

[14] Y. Freund and R. E. Shapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997. 3

[15] J. H. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, 29(5):1189–1232, 2001. 2, 3

[16] J. Gall and V. Lempitsky. Class-Specific Hough Forests for Object Detection. In *CVPR*, 2009. 1, 6, 7

[17] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, 2009. 3, 4, 5, 6, 7

[18] J. J. Hull. A Database for Handwritten Text Recognition Research. *PAMI*, 16(5):550–554, 1994. 5

[19] J. Jancsary, S. Nowozin, and C. Rother. Loss-Specific Training of Non-Parametric Image Restoration Models: A New State of the Art. In *ECCV*, 2012. 5

[20] P. Kontschieder, S. Rota Bulò, A. Criminisi, P. Kohli, M. Pelillo, and H. Bischof. Context-Sensitive Decision Forests for Object Detection. In *NIPS*, 2012. 5

[21] B. Leibe, N. Cornelis, K. Cornelis, and L. Van Gool. Dynamic 3D Scene Analysis from a Moving Vehicle. In *CVPR*, 2007. 7

[22] C. Leistner, A. Saffari, and H. Bischof. Semi-Supervised Random Forests. In *ICCV*, 2009. 2

[23] Y. Lin. A Note on Margin-based Loss Functions in Classification. *Statistics & Probability Letters*, 68(1):73–82, 2004. 2

[24] H. Masnadi-Shirazi, V. Mahadevan, and N. Vasconcelos. On the design of robust classifiers for computer vision. In *CVPR*, 2010. 3, 5, 6

[25] H. Masnadi-Shirazi and N. Vasconcelos. On the Design of Loss Functions for Classication: theory, robustness to outliers, and SavageBoost. In *NIPS*, 2008. 3, 5, 6

[26] T. Mitchell. *Machine Learning*. Mcgraw-Hill Higher Education, 1997. 2

[27] A. Montillo, J. Shotton, J. Winn, J. E. Iglesias, D. Metaxas, and A. Criminisi. Entangled Decision Forests and their Application for Semantic Segmentation of CT Images. In *IPMI*, 2011. 5

[28] F. Moosmann, B. Triggs, and F. Jurie. Fast discriminative visual codebooks using randomized clustering forests. In *NIPS*, 2006. 1

[29] R. E. Shapire. The Strength of Weak Learnability. *ML*, 5(2):197–227, 1990. 3

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

[31] H. Zou, J. Zhu, and T. Hastie. New multicategory boosting algorithms based on multicategory Fisher-consistent losses. *Annals of Applied Statistics*, 2(4):1290–1306, 2008. 2