# Decorrelated Random Forests for Image Classification

Agnieszka Putyra, Fang-Yu Liu, Fernando Bombardelli da Silva
*Technische Universität Berlin, Berlin, Germany*
{*aga.putyra, sponggyliu, bombardelli.f*}*@gmail.com*

Keywords: Random Forest, Decorrelated Trees, Ensemble Learning, Image Classification

Abstract: This paper introduces a novel method for training Random Forests for image classification, which we refer to as Decorrelated Random Forests. The goal of Decorrelated Random Forests is to reduce the correlation between the trees in the forest while keeping these individual classifiers as strong as possible. Strength and correlation play a vital role in our method because an estimate of the generalization error can be easily calculated based solely on these two measurements. With that, our algorithm estimates at each node creation the generalization error of the forest in the current state in order to obtain a Random Forest with increased accuracy for unseen data at the end of the training procedure. We provide a comparative analysis of the proposed method on a dataset of handwritten digits, and compare our approach with several state-of-the-art methods for image classification. Experiments show that our algorithm manages to decrease classification error rates when compared to a similar, standard Random Forest training method.

## 1 INTRODUCTION AND RELATED WORKS

Random Forests (RFs) are powerful methods of machine learning for classification, regression, and other tasks. RFs consist of a set of decisions trees, generated with random subsets of data. They are built with bagging (Breiman, 1996) and randomized training. Bagging implies that each decision tree is grown using independent subset of data, and randomized training refers to random selection of features at each node (parameter used for splitting). There are two important properties of RFs, strength and correlation. Strength refers to the accurateness of the individual trees in RFs, and correlation is a measure of dependencies between trees. RFs should be able to make as few mistakes as possible, and once the mistakes appear, then individual classifiers should make different mistakes. The stronger the individual decision trees and the lower the correlation between them, the better the performance.

Ho (1995) first proposed the general idea of random decision trees. In particular, she trained multiple trees on randomly chosen subspaces of the feature space. Through experiments her method outperformed standard decision tree on a handwritten digits classification dataset. However, the term Random Forest (RF) was coined by Breiman (2001), who developed a statistical framework for the analysis of tree-structured classifiers.

RFs have been successfully applied to the variety of computer vision tasks, including classification (Ho, 1995; Schulter et al., 2013; Ren et al., 2015), object detection (Kontschieder, 2012) as well as semantic segmentation (Schroff et al., 2008; Shotton et al., 2008). There are several characteristics, that make RFs particularly useful for computer vision tasks (Schulter et al., 2013): (i) fast in training and evaluation, (ii) robustness against label noise, (iii) suitability for parallel processing, (iv) good performance for high-dimensional input data and (v) inherent multi-class capability.

Lepetit and Fua (2006) proposed a keypoint recognition method using RFs by formulating the wide-baseline problem as a classification problem. They demonstrate the idea by developing a real-time system which detects planar, non-planar, and deformable objects. Another example of adapting RFs to specific computer vision problems is the Class-Specific Hough Forest (Gall and Lempitsky, 2013), which can be used in object detection in order to localize bounding boxes of the instances of a class in a test image. Kontschieder (2012) extended this method by proposing Context-Sensitive Decision Forests, which augments Hough Forests with contextual (intermediate prediction) information of the training set. Also semantic segmentation problems can be solved by RFs. Schroff et al. (2008) trained a RFs that can simulta-

neously segment an image into its semantic regions and label those regions from a predetermined set of classes. However, RFs are usually built beforehand with given training data, so they are not feasible for sequentially arriving training data. On this subject, on-line random forests were introduced to achieve on-line building decision trees (Saffari et al., 2009).

Ho (1998) first addressed the relation between the accuracy of forests and the independence among trees. She states that for a forest to achieve a better accuracy, it is critical that there should be some independence or dissimilarity between trees. And she proposed constructing a forest with trees building from randomly selected subset of a given feature space. Thus, each tree classifys in a different way. The generalization error of RFs can be described in terms of strength and correlation (Breiman, 2001). He showed that the generalization error of such ensemble method converges to a limit as the number of trees grows, and derived an upper bound for this error relating the strength of the trees and their correlation. According to it, RFs with trees that are strong — *i.e*. that missclassify as few data points as possible — and uncorrelated — *i.e*. whose decisions vary as much as possible — have lower generalization errors. Cutler and Zhao (2001) introduced a method for training RFs called Perfect Random Trees (PERT), which aims at reducing the error based on Breiman's upper bound. In this perspective, tree strength is increased by training each tree to fit its training set perfectly. Then, through randomization at the choice of split feature and threshold, the forest correlation is decreased.

Many methods have been developed in order to enhance strength of RFs. Kontschieder (2012) shared the same idea with Montillo et al. (2011), they proposed a so-called 'contextual-sensitive forests' including a set of contextual-sensitive decision trees which have the ability to do intermediate prediction with truncated trees before making decisions. Randomized feature selection also helps to produce stronger trees (Ren et al., 2015).

In standard RF, trees are grown independently (bagging), so there is no control over dependencies between individual trees. However, Rotation Forest (Rodriguez et al., 2006), a new ensemble generating method, is introduced which targets both individual accuracy and diversity within the ensemble. This method splits the feature set into subsets, runs principle componet analysis (PCA) on them, and then reassembles a new feature set. And the data is transformed accordingly, which is a rotation of coordinate axes. Different splits will lead to different rotations. Therefore, diverse classifiers are obtained. Also, they showed that for dataset from UCI Machine Learning repository, their method outperformed Bagging, AdaBoost, and Random Forest in WEKA.

More recently, Schulter et al. (2013) introduced Alternating Decision Forest (ADF), which incoporates global loss minimization into their RF growing scheme, and this scheme allows having control over the entire ensemble instead of growing isolated trees. Each training sample has its own weight, which is iteratively updated after growing a forest stage. ADFs alternate between global weight updates and parallel randomized tree growing. They evaluated the method in five different machine learning datasets and showed that ADF has a lower classification error when compared to common RFs and Boosted Trees. They also adopted the method into a Hough Forest for pedestrian and car detection, which improved accuracy in both scenarios. Similarly, Ren et al. (2015) proposed a method for dropping the size of a trained RF and reducing over-fitting on the data. This is achieved through a global refinement and global pruning procedure, which is realized jointly at the leaf nodes of all trees after the training is finished. Experiments showed a significant accuracy improvement when applied to a standard Random Forest, outperforming ADF in five standard datasets. Bader-El-Den (2014) incorporated the Genetic Algorithm within RFs, building the forest out of heterogeneous decision trees, heterogeneous here means trees with different number of features used at each split. They aimed to build a more diverse forest in order to enhance the accuracy of RFs. And their experimental results have shown that RFs performance could be boosted using the genetic algorithm approach.

This paper presents a new approach for training RFs, which we refer to as Decorrelated Random Forests. This method aims at decreasing the generalization error by reducing dependencies between the trees in the RF while keeping its individual classifiers strong. Section 2 presents the RF classifier and introduces the main contribution, describing the way strength and correlation are estimated. In section 3, we show our implementation design for the Decorrelated RF training algorithm. In sections 4 and 5, we describe the experiments and present the obtained results. Section 6 summarizes our work.

## 2 MATHEMATICAL MODEL

In this section, we formalize a Random Forest for the image classification problem. For that, let $X$ be the input space, $Y = \{1, \ldots, C\}$ the output space, where $C$ is the number of classes, and $f : X \rightarrow Y$ the function that maps an input to its ground truth class. To per-

form classification we build a set of $K$ binary decision trees, where each tree is indexed by $k \in \{1,\dots,K\}$. In our model, each tree is trained on a randomly sampled subset $T_k \subset X \times Y$, where for all $(x,y) \in T_k$ it holds that $y = f(x)$. After being trained, given an $x \in X$ a decision tree with index $k$ gives, for each class $\hat{y} = 1,\dots,C$, a probability $P_k(\hat{y}|x)$ of $\hat{y} = f(x)$. The results of the trees for each class $\hat{y} = 1,\dots,C$ are combined by average, thus the RF yields the probabilities

$$P(\hat{y}|x) = \frac{1}{K}\sum_{k=1}^{K} P_k(\hat{y}|x) \tag{1}$$

of $\hat{y} = f(x)$. With that, the prediction $y^*$ is given by

$$y^* = \arg\max_{y \in Y} P(y|x). \tag{2}$$

## 2.1 Predicting

Each non-leaf node in a binary decision tree partitions its dataset into two subsets (here referred to as *left* and *right*) according to a splitting function $s$ parametrized by $\theta = (\theta_1, \theta_2)$ as

$$s(x|\theta) = \begin{cases} 1 & \text{if } \varphi(x,\theta_1) < \theta_2, \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

where $\varphi$ is a function which calculates a feature for $x$ based on the parameter $\theta_1$. So, for a data point $x$, if $s(x|\theta) = 1$, then $x$ is in the left partition, if $s(x|\theta) = 0$, then it is in the right one.

Performing prediction of an input $x \in X$ in a tree with index $k$ consists in a recursive evaluation of a function $F_k$ given by

$$F_k(x,n) = \begin{cases} P_k(y|x) & \text{if } n \text{ is leaf node,} \\ F_k(x, \text{LeftChild}(n)) & \text{if } s(x|\theta_{k,n}) = 1, \\ F_k(x, \text{RightChild}(n)) & \text{if } s(x|\theta_{k,n}) = 0. \end{cases} \tag{4}$$

Then, probability $P_k(y|x)$ can be computed by $F_k(x, Root(k))$, and the final prediction by Eq. (2).

## 2.2 Training

Essentially, training a RF consists in growing individual decision trees. At each step a node is created and the algorithm has to decide whether to stop partitioning the dataset in the case that a stop criterion is fulfilled — hence creating a leaf node — or to continue the growing process. In the latter case, the optimal splitting function of the node's corresponding training data must be determined. More precisely, let $Z$ be the set of labeled data points in the training set of the current node, and let the measure of entropy $H$ of $Z$ be defined as

$$H(Z) = -\sum_{i=1}^{C} P(c_i|Z) \cdot \log_C P(c_i|Z), \tag{5}$$

where $C$ is the number of classes and $P(c_i|Z)$ is the proportion of data points of the class $c_i$ in the set $Z$. Then, the algorithm must find the parameter $\theta = (\theta_1, \theta_2)$ characterizing the splitting function $s$ that maximizes the information gain

$$I(\theta) := H(Z) - \frac{|L_\theta|\, H(L_\theta)}{|Z|} - \frac{|R_\theta|\, H(R_\theta)}{|Z|}, \tag{6}$$

where $L_\theta := \{(x,y) \in Z \,|\, s(x|\theta) = 1\}$ is the partition of the data points forwarded to the left and $R_\theta := \{(x,y) \in Z \,|\, s(x|\theta) = 0\}$ to the right.

In our method, trees are grown in a breadth-first manner. At each node, first, we randomly sample $M$ candidates for the parameter $\theta_1$ — which determines the feature $\varphi(x|\theta_1)$ to be used. For each one of the candidates, we choose a respective split point $\theta_2$ with Algorithm 1. With this algorithm, we seek to find a compromise between optimality and complexity: while it might find non-optimal split points, it has the advantage of being fast to compute. At the next step, we select $m < M$, out of these candidates, which yield the highest information gains. Finally, from these $m$ best parameters, we choose the one which minimizes a generalization error estimate $\xi > 0$ based on strength and correlation (*c.f.* Section 2.3).

**Algorithm 1** Split Point Selection

**Input:** Data points $X = (x_1,\dots,x_K)$, labels $Y$, parameter $\theta_1$, number of classes $C$
**Output:** Split point $\theta_2$ for a given $\theta_1$

```
// Calculate the features determined by θ1
Φ := [φ(x_1, θ_1), ..., φ(x_K, θ_1)]
// Calculate the mean for every class in array Φ
for c := 1 to C do
    Means[c] := mean(Φ[Y == c])
end for
// diff(A) calculates differences between adjacent elements of the array A
gap := argmax(diff(sort(Means)))
// θ2 is chosen as a value between the most distant classes in the feature array Φ
θ2 := (Means[gap] + Means[gap+1]) / 2
return θ2
```

## 2.3 Estimating Strength and Correlation

As mentioned above, there are two variables influencing generalization error of RFs: strength and corre-

lation. Breiman (2001) introduced the c/s2 ratio relating these two variables in an upper bound for the generalization error of a random forests. The c/s2 ratio is defined for $s > 0$ as

$$\text{c/s2} = \frac{\overline{\rho}}{s^2}, \tag{7}$$

where $\overline{\rho}$ is the correlation between the trees and $s$ is the expectation of the strength. In order to estimate these values, he presents the following procedure.

Let the trees in a RF be identified by a sequence $\{\Theta_k\}_{k=1}^K$ of random vectors, so that each tree casts a vote for a class $y = h(x,\Theta_k) \in Y$ given an input $x \in X$, that is,

$$h(x,\Theta_k) := \arg\max_{y \in Y} P_k(y|x). \tag{8}$$

Also, let $P_\Theta(h(x,\Theta) = j)$ denote the proportion of the trees which cast their votes to class $j$, and $mr(x,y)$ denote the margin function of the RF for a data point $(x,y)$, which is defined as

$$mr(x,y) = P_\Theta(h(x,\Theta) = y) - P_\Theta(h(x,\Theta) = \hat{j}(x,y)), \tag{9}$$

where

$$\hat{j}(x,y) := \arg\max_{j \neq y} P_\Theta(h(x,\Theta) = j) \tag{10}$$

is the most wrongly voted class for the input $x$.

Then, the strength $s$ of the RF is given by the expectation of the margin over a dataset,

$$s = \mathbb{E}_{x,y}[mr(x,y)], \tag{11}$$

and the correlation $\overline{\rho}$ by the following equation,

$$\overline{\rho} = \frac{\text{Var}_{x,y}(mr(x,y))}{\mathbb{E}_\Theta[sd_{x,y}(\Theta)]^2}, \tag{12}$$

where $\text{Var}_{x,y}(mr(x,y))$ is the variance of the margin over a dataset, and $sd_{x,y}(\Theta_k)$ is the standard deviation of the margin of the tree with parameter $\Theta_k \in \{\Theta_k\}_{k=1}^K$ over a dataset. The standard deviation can be calculated by (Breiman, 2001)

$$sd_{x,y}(\Theta_k) = \sqrt{p_1(\Theta_k) + p_2(\Theta_k) + (p_1(\Theta_k) - p_2(\Theta_k))^2}, \tag{13}$$

where

$$p_1(\Theta_k) := \mathbb{E}_{x,y}[h(x,\Theta_k) = y], \tag{14}$$

$$p_2(\Theta_k) := \mathbb{E}_{x,y}[h(x,\Theta_k) = \hat{j}(x,y)]. \tag{15}$$

Breiman (2001) introduces the so-called out-of-bag estimate for strength and correlation. For that, given the training set $T_k$ of each tree $k = 1,\ldots,K$, the probability $P_\Theta(h(x,\Theta) = j)$ is substituted by a function $Q$, given by

$$Q(x,j) = \frac{\sum_k I(h(x,\Theta_k) = j; (x,y) \notin T_k)}{\sum_k I((x,y) \notin T_k)}, \tag{16}$$

which is calculated at the end of the training procedure for the whole dataset. In this manner, in order to exclude the training set of the estimates, the class of each data point $x$ is predicted solely at those trees which have not used $x$ for learning. (Hence out-of-bag estimate.) Expectations are computed by averaging over the respective variables.

In our approach, we want to have a relatively fast-to-calculate estimate of the generalization error to use during training and which solely depends on the tree's bootstrap dataset. For that, while creating a node, instead of using an out-of-bag estimate, as explained above, we compute $P_\Theta(h(x,\Theta) = j)$ for the training set $T$ of the current tree as

$$P_\Theta(h(x,\Theta) = j) = \frac{\sum_{k=1}^K I(h(x,\Theta_k) = j)}{K}, \tag{17}$$

where $K$ is the number of tree in the ensemble. With this, we define the measure $\xi$ that we use to select the best feature when training the RF and also covers the case of non-positive strength:

$$\xi = \begin{cases} \overline{\rho}/s^2 & \text{if } s > 0, \\ U - s & \text{otherwise}, \end{cases} \tag{18}$$

where $U > 0$ is an upper bound of the c/s2 ratio. The use of $U$ is convenient to give a RF a cost even if its expected strength $s$ is non-positive. Refer to the appendix for the calculation of $U$.

# 3 IMPLEMENTATION DESIGN

Here we present pseudo code for implementation of Decorrelated Random Forests:

As mentioned in Section 2.2, at each node, $M$ candidates for parameter $\theta_1$ (characterizing the feature extraction $\varphi$) are randomly sampled. In our implementation, the choice of this parameter reflects how many and which patches are extracted from the input image $x$, and how they are combined together. More precisely, first, a number $n \in \{1,2,4\}$ of image patches is randomized. Subsequently, the coordinates of the $n$ regions are randomly chosen along with an operator which gives each patch a real value $v_1,\ldots,v_n$. Possible operators are *minimum value*; *maximum value*; *mean*; and *median*. Patches have fixed size of 5×5 pixels. Then, the feature for a given data point is given by

$$\varphi(x|\theta_1) = \begin{cases} v_1 & \text{if } n = 1, \\ v_1 - v_2 & \text{if } n = 2, \\ (v_1 - v_2) - (v_3 - v_4) & \text{if } n = 4. \end{cases} \tag{19}$$

The respective parameters $\theta_2$ are determined via Algorithm 1. Finally, the best candidate is chosen according to the pseudo code shown above.

**Algorithm 2** Training Algorithm

**Input:** Num. of trees, Max. height, Stop crit., $M$, $m$
  Initialize trees with random subsets of data
  **while** maximum height not reached **do**
    **for all** nodes in the current height **do**
      **if** stop criterion is fulfilled **then**
        Create **leaf node**
      **else**
        Randomly sampled $M$ features candidates
        **for all** $M$ candidates **do**
          Determine split point by Algorithm 1
          Calculate information gain
        **end for**
        Sort the $M$ candidates by information gain
        **for all** $m$ top candidates **do**
          Determine RF's strength and correlation
          Calculate corresponding c/s2 ratio
        **end for**
        Create **split node** with smallest c/s2 ratio
      **end if**
    **end for**
  **end while**

Table 1: Properties of the chosen datasets.

| Dataset | #Train | #Test | Pic. Size | #Classes |
|---|---|---|---|---|
| USPS | 2007 | – | 16×16 | 10 |
| MNIST | 55000 | 10000 | 28×28 | 10 |
| CIFAR-10 | 50000 | 10000 | 32×32 | 10 |

The tree growth process stops if (i) the number of data points in the node's dataset is less than a predefined threshold, or (ii) the probability of one of the classes in this dataset is greater than a predefined threshold, or (iii) the maximum depth is reached.

# 4 EXPERIMENTS

We carried out three types of experiments both to evaluate the influence of model parameters in classification accuracy and training time, and also to obtain a comparison between our Random Forest training algorithm with and without the decorrelation procedure. These experiments investigate (i) how the trained forest's strength, correlation, c/s2 ratio and classification error are influenced by the model parameters $M$ and $m$, *i.e.* number of sampled feature candidates and number of c/s2 ratio computations in the node creation, respectively; (ii) how these parameters affect training time; and (iii) what effect maximum height and number of trees have on the decorrelation approach compared to the standard one. For that, we used a relatively small but expressive digit classification dataset of the United States Postal Service (USPS) (Hull, 1994).

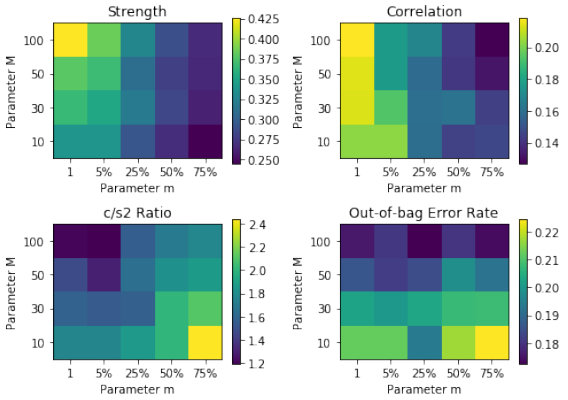


Figure 1: Influence of the number of sampled node split candidates (Parameter $M$) and the relative number of candidates tested for decorrelation (Parameter $m$) in strength, correlation, c/s2 ratio and error estimate. RFs are better the larger (yellower) the strength is and the lower (bluer) correlation, c/s2 ratio and error rate are.

Moreover, in order to compare our approach with the state of the art, we evaluated it on two other standard datasets: MNIST[1] and CIFAR-10 (Krizhevsky and Hinton, 2009). MNIST database contains grayscale images of handwritten digits of size 28×28 pixels. It consists of 55000 training images and 10000 test images. CIFAR-10 consists of 60000 color images of size 32×32 in 10 classes, including airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Images are split into 50000 training images and 10000 test images. In our experiment, we convert the color images to grayscale to reduce the complexity of the problem. Properties of the chosen datasets are summarized in Table 1.

## 4.1 Parameter setting

For the first two experiments with the USPS dataset, we fix maximum height equal to 5 and number of trees to 25, and let the parameters $M$ and $m$ vary. Each tree is initialized with a bootstrap training set of a quarter of the dataset's size. For the rest of the experiments on this dataset, we set $M = 100$ and $m = 5$, and vary maximum height and number of trees.

For classification of handwritten digits from the MNIST dataset, we used RF with 100 trees, where each tree was trained with 1000 randomly sampled images. As suggested by Breiman (2001), we set the number of features tested at each node to 28, which is the square root of feature dimension. Moreover, we

[1] See `http://yann.lecun.com/exdb/mnist/`

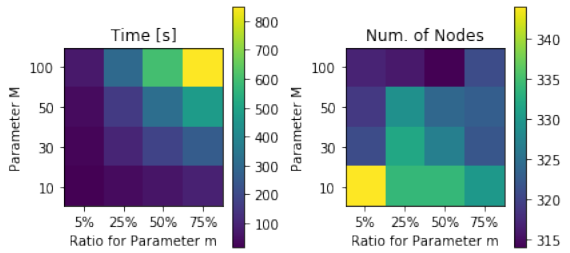


Figure 2: Influence of the number of sampled node split candidates (Parameter $M$) and the relative number of candidates tested for decorrelation (Ratio for Parameter $m$) in the training time (in seconds) and in the number of nodes of the trained RF.

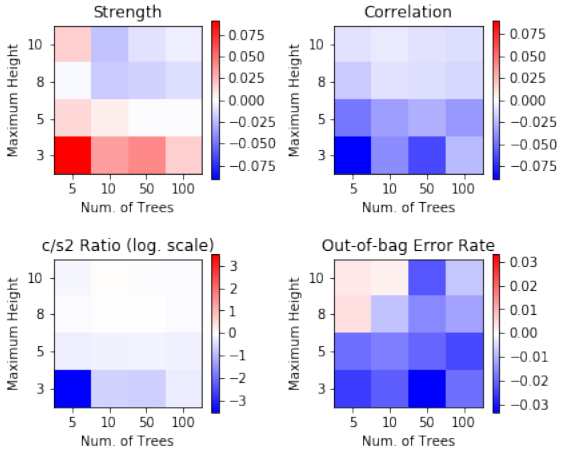


Figure 3: Comparison of strength, correlation, c/s2 ratio and error rate between RFs trained with and without our decorrelation procedure. The figures depict the numerical difference of the respective aspects as the value for RF without decorrelation subtracted from the one with decorrelation procedure. Decorrelated RFs are better the more positive (redder) the strength comparison (top-left plot) is and the more negative (bluer) correlation, c/s2 ratio and error rate plots are.

set maximum depth of the trees to 15 and the minimum number of samples required for further splitting to 10.

We used most of the same parameters on CIFAR-10 as on MNIST, including number of trees, number of samples used to train a tree, maximum depth, and the minimum number of samples required for further splitting. Only the number of features tested at each node is set to 32.

## 5 RESULTS

Figure 1 illustrates how the strength, correlation, c/s2 ratio and the error estimate vary with changes of the parameters $M$ and $m$. One can observe on the charts that the more candidate splits are sampled — *i.e.* the higher $M$ is — and the fewer candidates are tested in the decorrelation procedure — *i.e.* the smaller $m$ is —, the stronger the trees become. Similarly, the generalization error estimate *c/s2 ratio* decreases in RFs that have higher $M$ and relatively low $m$. The effect of the decorrelation algorithm may be noticed in the resulting RF correlation, which tends to decline as $m$ grows. Moreover, a general decrease of error rate can be seen, when $M$ is higher. From that, we can conclude that a good parameter choice would definitively be high $M$ and relatively low $m$ (about 5%).

The results of our experiments concerning training time and number of nodes, *i.e.* size of the trees, as a functions of those same parameters is depicted in Figure 2. These experiments were made on an Intel Core i3 at 2.3 GHz with 4GB RAM. The code was written in Python and the classifier trained on Linux. From these pictures, it can be seen that the parameter $m$ drives the training time up, which may have an increase of about eight times at $M = 100$ as $m$ gets larger. This can be explained by the fact that the decorrelation procedure is the processing bottleneck, thus the more time the c/s2 ratio is evaluated the longer the training process will take. On the other hand, the constructed trees tend to be smaller if $m$ is larger. This might suggest that less overfitting is occurring in these cases and that the classifier generalizes better. Furthermore, for this same reason, one could argue that the time spent training more compact trees will pay off by resulting in a reduction of prediction time.

Figure 3 shows the results of the comparison experiments between RFs trained with the proposed decorrelation procedure and without it. The illustrations compare strength, correlation, c/s2 ratio and error rate for different values of maximum tree height and number of trees in the forest on a diverging, color-coded scale centered at 0. The values shown represent the difference between both methods. More precisely, for each measured characteristic (strength, correlation, *etc.*), we depict the value for the RF without decorrelation subtracted from the one with decorrelation. In order to better visualize the chart for c/s2 ratio, we display the logarithm of the corresponding differences. According to these graphs, the training algorithm with decorrelation creates stronger and less correlated trees for small RFs, such as those with five trees or with maximum height equal to three or five. Similarly, the c/s2 ratio for those RFs are also significantly reduced by applying decorrelation. Never-

Table 2: Decorrelated Random Forest compared with other RF-based methods on the MNIST dataset.

| Method | Test set error |
|---|---|
| Decorrelated Random Forest | 7.09% |
| Alternating Decision Forest (Schulter et al., 2013) | 2.71% |
| Boosted Trees (Hastie et al., 2009) | 3.15% |
| Random Forest (Ho, 1995) | 6.68% |
| Refined Random Forest (Ren et al., 2015) | 2.05% |

theless, larger RFs tend to be stronger when trained without our approach for decorrelation, and the differences in tree correlation and c/s2 ratio tend to zero. One reason for this is that the correlation between a greater number of trees naturally decreases by their increase in size. Furthermore, a lower error rate can be noticed with the use of decorrelation, however the deviations are too small to draw a more general conclusion on the subject. All in all, this figure suggests that applying our decorrelation algorithm may become especially advantageous for training smaller, more compact RFs.

Furthermore, we compared our method with the state of the art. Table 2 presents the classification errors of different RF-based methods, including Decorrelated RF, ADF (Schulter et al., 2013), Boosted Trees (Hastie et al., 2009), RF (Ho, 1995) and Refined RF (Ren et al., 2015) measured on the MNIST validation dataset. With Decorrelated RF algorithm we achieved an error of 7.09%, which is bigger than the classification errors of the selected RF-based methods. The reason of such poor performance in comparison to other algorithms may be insufficient computational capacity. We expect that increasing the maximum height of the RF to 25 would improve the performance. Due to high computational burden of performing such an experiment we were unable to test it.

On CIFAR-10, the error rate of testing set is 63%, which should be improved with further parameter tuning. Due to our limited computation resource, it took a long time for training. Therefore, there is not enough time to run more tests.

# 6 CONCLUSION AND FUTURE WORK

In this work, we presented a Decorrelated Random Forests algorithm, which reduces classification error by decreasing the correlation between the trees in the RF while keeping them as strong as possible. The proposed approach generates the forest in a breadth-first manner in order to estimate its generalization error during the creation of each split node using strength and correlation. We also presented a new heuristic-based algorithm for split point selection in order to cope with the computational complexity of the training procedure.

Our experiments on machine learning data confirm that the Decorrelared Random Forests algorithm results in lower correlation, c/s2 ratio as well as generalization error compared to classic RF algorithm, especially for smaller, more compact RFs. The drawback of presented method is that the decorrelation procedure is complex and requires a longer computation time. Possible area of future research is optimization of the implementation of Decorrelated RF that reduces computational complexity and thus shorten training time.

# REFERENCES


Bader-El-Den, M. (2014). Self-adaptive heterogeneous random forest. In *Computer Systems and Applications (AICCSA), 2014 IEEE/ACS 11th International Conference on*, pages 640–646. IEEE.

Breiman, L. (1996). Bagging predictors. *Machine learning*, 26:123–140.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Cutler, A. and Zhao, G. (2001). PERT-perfect random tree ensembles. *Computing Science and Statistics*, 33:490–497.

Gall, J. and Lempitsky, V. (2013). Class-specific hough forests for object detection. In *Decision forests for computer vision and medical image analysis*, pages 143–157. Springer.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning*.

Ho, T. K. (1995). Random decision forests. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 1, pages 278–282. IEEE.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, 20(8):832–844.

Hull, J. J. (1994). A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence*, 16(5):550–554.

Kontschieder, P., B. S. R. C. A. K. P. P. M. . B. H. (2012). Context-sensitive decision forests for object detection. In *Advances in neural information processing systems*, pages 431–439.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images.

Lepetit, V. and Fua, P. (2006). Keypoint recognition using randomized trees. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 28.9, pages 1465–1479.

Montillo, A., Shotton, J., Winn, J., Iglesias, J., Metaxas, D., and Criminisi, A. (2011). Entangled decision forests and their application for semantic segmentation of ct images. *In Information Processing in Medical Imaging*, pages 184–196.

Ren, S., Cao, X., Wei, Y., and Sun, J. (2015). Global refinement of random forest. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 723–730.

Rodriguez, J. J., Kuncheva, L. I., and Alonso, C. J. (2006). Rotation forest: A new classifier ensemble method. *IEEE transactions on pattern analysis and machine intelligence*, 28(10):1619–1630.

Saffari, A., Leistner, C., Santner, J., Godec, M., and Bischof, H. (2009). On-line random forests. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 1393–1400. IEEE.

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

Schulter, S., Wohlhart, P., Leistner, C., Saffari, A., Roth, P. M., and Bischof, H. (2013). Alternating decision forests. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 508–515.

Shotton, J., Johnson, M., and Cipolla, R. (2008). Semantic texton forests for image categorization and segmentation. *In CVPR*.

# APPENDIX: Calculating an upper bound for the c/s2 ratio

Let $C > 0$ be the number of classes of the classification problem, $K > 0$ the number of trees in the forest, and $N > 0$ be the number of elements of the dataset on which the c/s2 is to be calculated. Then, we want to find an upper bound $U \in \mathbb{R}$ such that $U \geq \overline{\rho}/s^2$ for any RF with $s > 0$. That is, we want to find an upper bound for $\overline{\rho}$ and a lower bound for $s$ given that $s > 0$. We start with the former. For that, refer to Eq. (12) and note that $\mathrm{Var}_{x,y}(mr(x,y)) \leq 1$ since $mr(x,y) \leq 1$ for all $(x,y)$. Therefore, to find an upper bound for $\overline{\rho}$ we only need to find a positive (non-zero) lower bound for $\mathbb{E}_\Theta[sd_{x,y}(\Theta)]^2$. Since the standard deviation is non-negative, *i.e.* $sd_{x,y}(\Theta_k) \geq 0$ for any $k = 1, \ldots, K$, there exists a tree $\Theta_k$ which bounds the expectation as

$$\begin{aligned}\mathbb{E}_\Theta[sd_{x,y}(\Theta)]^2 &\geq \frac{sd_{x,y}(\Theta_k)^2}{K^2} \\ &= \frac{p_1(\Theta_k) + p_2(\Theta_k)}{K^2} \\ &\quad + \frac{(p_1(\Theta_k) - p_2(\Theta_k))^2}{K^2} \\ &\geq \frac{p_1(\Theta_k) + p_2(\Theta_k)}{K^2}, \end{aligned} \tag{20}$$

refer to Eq. (14) and (15) for the definitions of $p_1$ and $p_2$. Now, note that for any $p_1(\Theta_k)$, there exists a minimum value for $p_2(\Theta_k)$, namely, in the case that votes are distributed equally between the wrong classes, that is,

$$\begin{aligned} P_\Theta(h(x,\Theta) = \hat{j}(x,y)) &= P_\Theta(h(x,\Theta) = j) \qquad \forall j \neq y \\ &= \frac{1 - P_\Theta(h(x,\Theta) = y)}{C-1} \end{aligned} \tag{21}$$

Then, it holds that

$$p_2(\Theta_k) \geq \frac{1 - p_1(\Theta_k)}{C-1} \tag{22}$$

and thus,

$$\begin{aligned}\mathbb{E}_\Theta[sd_{x,y}(\Theta)]^2 &\geq \frac{p_1(\Theta_k)(C-2)+1}{K^2(C-1)} \\ &\geq \frac{1}{K^2(C-1)}, \end{aligned} \tag{23}$$

by the non-negativity of $p_1(\Theta_k)$ for any $k$.

Our upper bound for $\overline{\rho}$ follows from Eq. (12) and (23), and is given by

$$\overline{\rho} \leq K^2 (C-1). \tag{24}$$

In order to find a positive lower bound for $s$, note that $-1 \leq mr(x,y) \leq 1$ for all $(x,y)$. Thus, a lower bound can be determined as the smallest distance that the expected value $\mathbb{E}_{x,y}[mr(x,y)]$ may be from zero. Since the smallest margin is the one in which both terms in Eq. (9) differ by only the vote of one tree (*i.e.* $1/K$). Assuming there are $N$ data points, it must hold that

$$s \geq \frac{1}{KN}. \tag{25}$$

And, with that

$$\frac{\overline{\rho}}{s^2} \leq K^2 (C-1) (KN)^2 =: U. \tag{26}$$