Thresholding a Random Forest Classifier*

Florian Baumann[1], Fangda Li[2], Arne Ehlers[1], Bodo Rosenhahn[1]

[1] Institut für Informationsverarbeitung (TNT), Leibniz Universität Hannover, Germany
[2] Electrical and Computer Engineering, Purdue University, United States

Abstract. The original Random Forest derives the final result with respect to the number of leaf nodes voted for the corresponding class. Each leaf node is treated equally and the class with the most number of votes wins. Certain leaf nodes in the topology have better classification accuracies and others often lead to a wrong decision. Also the performance of the forest for different classes differs due to uneven class proportions. In this work, a novel voting mechanism is introduced: each leaf node has an individual weight. The final decision is not determined by majority voting but rather by a linear combination of individual weights leading to a better and more robust decision. This method is inspired by the construction of a strong classifier using a linear combination of small rules of thumb (AdaBoost). Small fluctuations which are caused by the use of binary decision trees are better balanced. Experimental results on several datasets for object recognition and action recognition demonstrate that our method successfully improves the classification accuracy of the original Random Forest algorithm.

1 Introduction

Random Forest is a machine learning algorithm by Leo Breiman [1] for classification and regression consisting of an ensemble of independent decision trees. Each tree is learned with randomly selected samples and features. Compared to other ensemble learning algorithms, i.e. boosting [2], that build a flat tree structure of decision stumps, Random Forest is multi-class capable and has some preferable characteristics such as a faster training procedure and useful internal estimates [1].
Regarding a single tree in the forest a drawback is the susceptibility against small fluctuations and noise. A wrong decision of a node might lead to a completely other classification result. Many circumstances like noisy or blurred images, occlusions or a large intra-class variation can easily lead to a misclassification. Figure 1 illustrates a typical binary classification tree with three nodes to detect faces. Each node samples a feature in an image. If a node does not match because of the above mentioned circumstances the image is shifted down to another path (see node no. 2). Usually these small fluctuations are compensated by the majority voting of the Random Forest but our intention is to reduce the influence of these fluctiations.
Contribution: In this work, we focus on improving the accuracy of a Random Forest classifier and propose a thresholded variant inspired by the AdaBoost algorithm to eliminate the sensitivity against path fluctuations. Each leaf node is weighted according

*This work has been partially funded by the ERC within the starting grant Dynamic MinVIP.

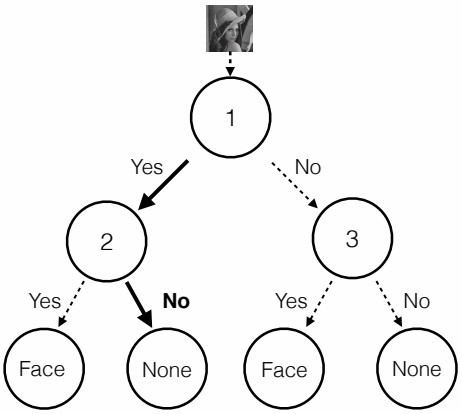
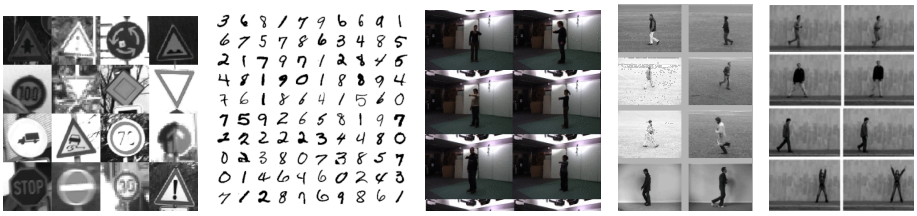


Fig. 1. A typical binary decision tree. Small changes/fluctuations like noisy/blurred images, occlusions or a large intra-class variation affect the correct way and lead to a misclassification.

to a computed uncertainty that acts like an importance indicator. Bad nodes are punished while well performing nodes are weighted higher. The class-specific weights of all trees are summed up and a threshold is introduced. The threshold is automatically learned and specifies the percentage of weights that have to be available for classifying the object. Furthermore, the threshold allows a parametrization of the algorithm and an additional possibility to adapt the forest to a specific dataset. Our method is inspired by the AdaBoost algorithm and the linear combination of weak classifiers for constructing a strong classifier.
This paper is structured as follows: Section 1.1 gives a brief overview about related work. Section 2 explains the AdaBoost algorithm while the Random Forest algorithm is described in Section 3. Our proposed method is explained in detail in Section 4. Experimental results are presented in Section 5 and conducted on well-known datasets for object recognition, GTSRB and MNIST, see Figures 2(a) and 2(b) and on action recognition datasets like IXMAS, KTH and Weizman, see Figures 2(c), 2(d) and 2(e). Our paper is concluded in Section 6.

1.1 Related Work

Generally, we can distinguish between two main research areas: the improvement of AdaBoost by using methods of Random Forest and the improvement of Random Forest by using methods of AdaBoost. Most of the works try to improve AdaBoost [10–15]. For instance, researchers implement a decision tree as a weak learner and use this tree to construct a strong classifier. Other researchers replace the exhaustive search of the AdaBoost algorithm by the random feature selection mechanism of Random Forest or reduce the amount of data points to decrease the training time of AdaBoost [16].
There are some other works that combine key contributions of AdaBoost into the Random Forest. For instance, Schulter et al. minimizes losses via keeping an adaptive weight distribution over the training samples [17, 18] . Bernard et al. propose a Dynamic



(a) (b) (c) (d) (e)

Fig. 2. (a) Example images of the German Traffic Sign Recognition Benchmark [3]. (b) Example images of the MNIST dataset for handwritten digit recognition [4]. Example images of the human action recognition datasets: (c) IXMAS multi-view [5, 6] (d) KTH [7] (e) Weizman [8, 9].

Random Forest algorithm which is based on an adaptive tree induction procedure [19]. In addition to these works, we propose a framework inspired by the AdaBoost algorithm that combines the basic idea of a linear combination of small rules of thumb. Instead of using a majority voting, the final decision is gathered by summing up class-specific weights for each leaf node and determining the decision by introducing a threshold.

2 AdaBoost

In this chapter, we focus on the main principles of AdaBoost that are important to understand this work. Detailed information are available in the works of Freund and Shapire [2] and Viola and Jones [20]. AdaBoost is a machine learning algorithm proposed by Freund and Shapire [2], which was further enhanced with Haar-like Features and Integral Images and applied to the task of real-time face detection by Viola and Jones [20]. Typically, an AdaBoost classifier consists of a stage of several strong classifiers with increasing complexity. Each strong classifier is constructed by a linear combination of weak classifiers while a weak classifier consists of a feature with a threshold and a weight (see [20] for more information). The weight is computed with respect to the number of misclassified examples[3] while the influence of poorly performing weak classifiers is decreased.

Training: Given a training set with N samples, $(\boldsymbol{X},\boldsymbol{Y}) = \{(x_1,y_1),...,(x_N,y_N)\}$, where x_i is an image containing all potential features and y_i = 0, 1 for negative and positive label. Initialize weights for negative and positive separately. For training a weak classifier:

1. For every possible feature, learn a potential weak classifier and calculate error based on sample weights.
2. For all potential weak classifiers, choose the one with lowest error.
3. Update all weights in a manner that current misclassified samples are weighted higher for training the next weak classifier.
4. Normalize weights for all samples.

[3] The weight is $\alpha = \log \frac{1-\epsilon}{\epsilon}$ with ϵ, the sum of misclassified sample weights.

After learning several weak classifiers, the corresponding error is used to weight each weak classifier and determine the influence in the linear combination:

$$h(\boldsymbol{x}) = \begin{cases} 1, & \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Classification: $h(\boldsymbol{x})$ is the final decision, $h_t(\boldsymbol{x})$ is the decision of weak classifier t and α_t is the weight of weak classifier t. Each α_t is computed based on the error rate of that specific weak classifier. $\frac{1}{2}$ serves as a threshold for classification and the idea of thresholding is introduced into Random Forest.

3 Random Forest

In this Section the Random Forest algorithm developed by Leo Breiman [1] is briefly described. Random Forest is an ensemble learning method for classification and regression, which combines the idea of Bagging [21] with a random feature selection proposed by Ho [22, 23] and Amit [24]. Random Forest consists of CART-like decision trees that are independently constructed on a bootstrap sample. Compared to other ensemble learning algorithms, i.e. boosting [2] that build a flat tree structure of decision stumps, Random Forest uses an ensemble of unpruned decision trees, is multi-class capable and has some preferable characteristics [1]:

- Similar or better accuracy than AdaBoost.
- Robust to noise and outliers.
- Faster training than bagging or boosting.
- Useful internal estimates: error, strength, correlation and variable importance.

Training: Given a dataset containing N examples for training:

$$(\boldsymbol{X}, \boldsymbol{Y}) = \{(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_N, y_N)\}, \tag{2}$$

where $\boldsymbol{x}_i$ is the feature vector of M dimensions and y_i is the class label which value is between 1 and K. To grow a tree, the following steps are necessary:

1. Choose n_{tree} samples from the whole training set ($\boldsymbol{X}$,$\boldsymbol{Y}$) at random.
2. The remaining samples are used to calculate the out-of-bag error (OOB-error).
3. At each node randomly specify $m_{try} << M$ variables and find the best split.
4. Completely grow the tree to the largest possible extension without pruning.

Classification: A completed Random Forest consists of several classification trees ($1 \leq t \leq T$) in which the class probabilities, estimated by majority voting, are used to calculate the sample's label $y(\boldsymbol{x})$ with respect to the feature vector $\boldsymbol{x}$:

$$y(\boldsymbol{x}) = \underset{c}{\operatorname{argmax}} \left(\frac{1}{T} \sum_{t=1}^{T} I_{h_t(\boldsymbol{x})=c} \right) \tag{3}$$

The decision function $h_t(\boldsymbol{x})$ provides the classification of a tree to a class c with the indicator function I:

$$I_{h_t(\boldsymbol{x})=c} = \begin{cases} 1, & h_t(\boldsymbol{x}) = c, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

A sample is classified by passing it down each tree until a leaf node is reached. A classification result is assigned to each leaf node and the final decision is determined by taking the class having the most votes, see Equation (3).

4 Proposed Method

In AdaBoost [20], the strong classifier is constructed by several small rules of thumb, also see Equation (1) and for classification, only if a weak classifier matches in an image the corresponding weight of this weak classifier is summed up. Additionally, the influence of this decision depends on an a-priori computed weight to decrease the influence of poorly performing weak classifiers. Finally, several decisions are combined by using a linear combination.
In this work, we integrate the concept of a linear combination of decisions into the Random Forest algorithm and interpret a leaf node as a weak classifier leading to several advantages in comparison to a standard binary decision tree:

1. Decisions are weighted with respect to an uncertainty.
2. Not all decisions are necessary to classify an object.
3. Decisions are more robust.
4. Using an additional threshold leads to a parameterizable Random Forest.
5. A Random Forest can be better adapted to a specific task.

In the following, we present α values as a weight parameter for leaf nodes. Further, we introduce thresholds based on these α values to improve the final classification accuracy.

4.1 Leaf Node α-Weight

We introduce a weight to determine the importance of each leaf node. This importance is computed with respect to the depth of the corresponding leaf node:

$$\alpha_{l,c} = \begin{cases} \frac{1}{log(D_l)}, & c = k_l, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

where $\alpha_{l,c}$ is the alpha value of a leaf node l for class c, k_l is the trained label of leaf node l, D_l is the depth of leaf node l.
Experiments on how to compute the weight of a leaf node have clearly shown that using the depth instead of a error rate on the relative frequency of classes leads to better accuracies. Figure 3 reveals that wrong decisions are mainly caused by paths with a large depth (the experiment was conducted on a action recognition dataset). Therefore, $\alpha_{l,c}$ is being punished if the depth gets larger.

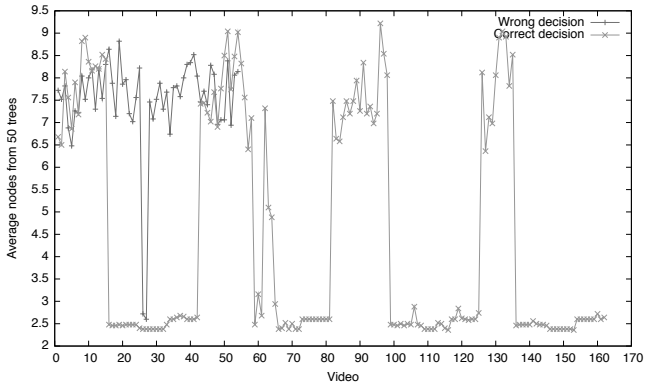


Fig. 3. A large depth mainly leads to a misclassification. Therefore, the depth is used as an indicator for the decision.

4.2 Decision Function

The original decision function, see also Equation (3), is enhanced to the following Equation:

$$y(\boldsymbol{x}) = \underset{c}{\operatorname{argmax}} \left(\frac{\sum_{t=1}^{T} I_{h_t(\boldsymbol{x})=c}}{\sum_{t=1}^{T} \sum_{l=1}^{L} \alpha_{l,c}} - \gamma_c \right), \tag{6}$$

where the sample's label $y(\boldsymbol{x})$ is chosen with respect to the indicator function $I_{h_t(\boldsymbol{x})=c}$ of a tree and normalized by the sum of the class-specific weights $\alpha_{l,c}$ of all trees and leaf nodes. Further, the final decision is regulated by the γ_c function leading to smoothed decisions. The modified indicator function $I_{h_t(\boldsymbol{x})=c}$ is defined as:

$$I_{h_t(\boldsymbol{x})=c} = \begin{cases} \alpha_{l,c}, & h_t(\boldsymbol{x}) = c, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

By introducing the modified indicator function it is not necessary that all leaf nodes vote for a class (similar to AdaBoost). To find γ_c, all training samples are reapplied to the forest and the sum of misclassified leaf nodes for all m samples is computed:

$$\gamma_c = max \left\{ \underbrace{\sum_{t=1}^{T} \alpha_{l,c}}_{\text{sample 1}}, ..., \underbrace{\sum_{t=1}^{T} \alpha_{l,c}}_{\text{sample m}} \right\}, \tag{8}$$

where γ_c represents the relative easiness for a sample to be misclassified as class c. A relatively higher γ_c indicates a sample was relatively easier to be misclassified as class c. As a result, later in Equation (6), a sample needs relatively more votes to be classified as class c. In other words, the final decision is no longer the most voted class. Earlier in the paper, we mention that Random Forest has a lack of considering different capabilities over different classes. Suppose a forest is trained with mostly class 1 samples

and only a few of class 2. When classifying a class 2 test sample, likely the correct votes for class 2 will be outnumbered by the misclassified votes for class 1, since the majority of classifying power is designated for class 1, see Figure 4. By introducing the regularization term, this drawback is eliminated.

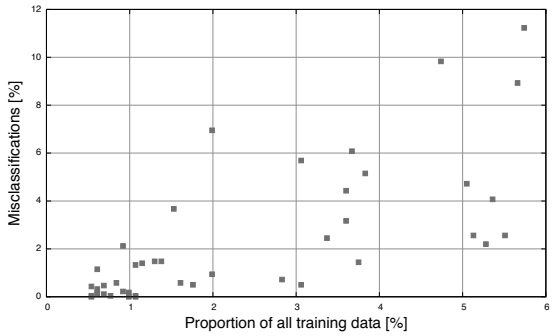


Fig. 4. Each datapoint represents a class with respect to the proportion to all training samples and to the number of misclassified samples. Classes with a larger number of samples obtain more misclassifications.

5 Experimental Results

In this Section, we introduce the datasets used to evaluate our method and report results in comparison to a standard Random Forest implementation and to state-of-the-art methods.

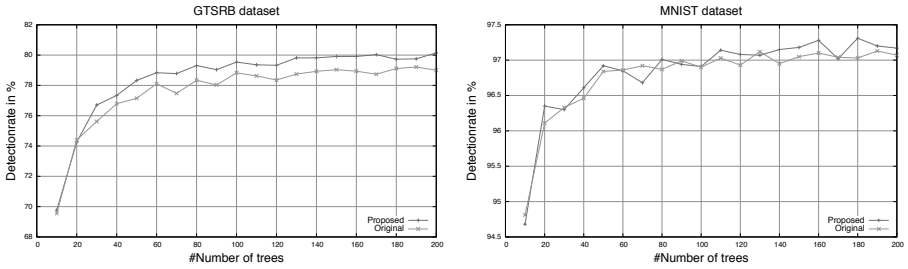
5.1 Datasets

The **GTSRB** dataset (German Traffic Sign Recognition Benchmark) is a large and life-like dataset, containing traffic signs with many different background changes, illumi-

Method	Error rate (%)
Bernard et al. [25]	6.73
Bernard et al. [26]	5.92
Fan et al. [27]	4.95
RF 4.6-7 in R[4]	3.0
Proposed method	**2.69**

Table 1. Our proposed framework in comparison to state-of-the-art methods on the MNIST dataset.

[4] http://www.wise.io/blog/benchmarking-random-forest-part-1



(a) (b)

Fig. 5. A comparison of original Random Forest to our proposed method. (a) On the GTSRB dataset. (b) On the MNIST dataset. Our proposed method clearly outperforms the original Random Forest.

nation changes, occlusions and distortions [3]. This dataset consists of more than 40 classes and more than 50000 images in total.
Further, experiments were conducted on the **MNIST** dataset for handwritten digit recognition [4]. This dataset consists of 60000 training images and 10000 test images. For both experiments, the pixel values of each image are used as features for classification.
More experiments are conducted using typical human action recognition datasets:
KTH: The well-known and publicly available KTH dataset [7] consists of six classes of actions. Each action is performed by 25 persons in four different scenarios. The KTH dataset consists of 599 videos. Similar to [28], a fixed position bounding box with a temporal window of 24 frames is selected, based on annotations by Lui [29]. Presumably, a smaller number of frames is sufficient [30].
Weizman: We evaluate our proposed framework on the well-established Weizman action dataset [8, 9]. In our opinion, the Weizman dataset is already solved since many researchers report accuracies of 100%. However, in recent publications [31–33] this dataset is still used to evaluate the corresponding methods. In order to allow a comparison to these works and to show the benefit of our framework, we use this dataset too. The Weizman dataset consists of nine actions while each action is performed by nine different persons. We manually labeled the dataset and used the bounding boxes for the classification. The bounding boxes are available for download at our homepage[5].
IXMAS: Additionally, our framework is evaluated on the IXMAS dataset for multi-view action recognition [5, 6]. The IXMAS dataset contains 12 classes of actions. Each action is performed three times by 12 persons while the body position and orientation is freely chosen by the actor. The IXMAS dataset consists of 1800 videos.

5.2 Comparison to an original Random Forest and to state-of-the-art methods

In this Section we compare the proposed method to a standard Random Forest implementation and to state-of-the art methods. In the following, we report results on the GTRSB, MNIST object recognition dataset. For these experiments the pixel values are

[5] http://www.tnt.uni-hannover.de/staff/baumann/

Name	Accuracy (%)
Jhuang et al. [34]	98.80
Lin et al. [35]	100.00
Blank et al. [8]	100.00
Gorelick et al. [9]	100.00
Schindler and Van Gool [30]	100.00
Proposed method	**100.00**

Table 2. Accuracy for our proposed framework on the Weizman dataset. The accuracy is 100%.

Name	Accuracy (%)
Yeffet and Wolf [36]	90.1
Laptev et al. [37]	91.8
Schindler and Van Gool [30]	92.7
Kihl et al. [38]	93.4
Baumann et al. [39]	94.4
Proposed method	**96.88**

Table 3. Accuracy for our proposed framework on the KTH dataset. The accuracy is 96.88%.

directly used as features for classification. Better results might be achieved by using more suitable features. Further, we report results on the KTH, Weizman and IXMAS action recognition dataset. For these datasets, Motion Binary Patterns are used [39].

GTSRB: Figure 5(a) illustrates our method in comparison to a standard Random Forest implementation with respect to the number of trees. Our method clearly outperforms the original Random Forest. At this point, we mention that the pixel values are directly used as features for classification. Better results might be achieved by using more suitable features like HOG features. Regarding the competition results table[6] other researchers report results at 99.98% by using more complex features for classification while a standard Random Forest implementation achieves 64.15%.

MNIST: In Figure 5(b) we compare our proposed method to the original Random Forest on the MNIST dataset with respect to the number of trees. Despite of some fluctuations our proposed method outperforms the standard implementation. Especially at high detection rates our method achieves a better detection accuracy. Table 1 compares our method to state-of-the art algorithms where the lowest error rate of 2.69% is achieved.

Weizman: Table 2 illustrates the results for the Weizman dataset in comparison to state-of-the-art methods. We report an accuracy of 100%. For this dataset, Motion Binary Patterns are used [39].

KTH: Table 3 presents a comparison to recent approaches on the KTH dataset. Motion

Name	Accuracy (%)
Wang et al. [40]	76.50
Wu et al. [41]	78.02
Li and Zicker [42]	81.22
Proposed method	**84.20**

Table 4. A comparison to single- and multi-feature methods on the IXMAS multi-view action recognition dataset.

[6] http://benchmark.ini.rub.de

Binary Patterns are used for classification [39]. The best achieved accuracy is 96.88% while most confusions appear between similar actions like running and jogging.
IXMAS: Table 4 illustrates the results in comparison to state-of-the-art methods using the IXMAS multi-view action recognition dataset. For each view a Random Forest classifier was learned. The final decision is determined by adding the class probabilities and choosing the class with the highest sum. The best accuracy is 84.20%. Most confusions appear for getting up, waving, punching and kicking. For this dataset, Motion Binary Patterns are used [39].

6 Conclusion

In this work, we propose a new weight representation of leaf nodes for the Random Forest algorithm. In contrast of evaluating every leaf's decision, a weight with respect to the depth of a path is introduced which compensates the statistical effect of long paths leading to better accuracies. By summing up the weights of all trees, a more robust decision is gathered and the effect of path fluctuations is eliminated. The final classification result is computed by the most weights above a specific class threshold. Further, the threshold of each class serves as a regularization term to correct discriminated misclassified samples caused by unbalanced training proportions and different classifiable classes. Experiments on several datasets for object recognition (GTSRB, MNIST) and action recognition (KTH, Weizman, IXMAS) confirm that our method achieves a better classification rate than the original Random Forest and than state-of-the-art methods.

References

1. Breiman, L.: Random forests. In: Machine Learning. Volume 45. (2001) 5–32
2. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: Machine Learning, Proceedings of the Thirteenth International Conference on, IEEE (1996) 148–156
3. Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C.: Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. Neural Networks (2012) –
4. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86** (1998) 2278–2324
5. Weinland, D., Özuysal, M., Fua, P.: Making action recognition robust to occlusions and viewpoint changes,. In: European Conference on Computer Vision (ECCV). (2010)
6. Weinland, D., Ronfard, R., Boyer, E.: Free viewpoint action recognition using motion history volumes. In: Computer Vision and Image Understanding (CVIU). (2006)
7. Schuldt, C., Laptev, I., Caputo, B.: Recognizing human actions: a local svm approach. In: Pattern Recognition. (ICPR). Proceedings of the 17th International Conference on. (2004)
8. Blank, M., Gorelick, L., Shechtman, E., Irani, M., Basri, R.: Actions as space-time shapes. In: Computer Vision (ICCV), 10th International Conference on. (2005) 1395–1402
9. Gorelick, L., Blank, M., Shechtman, E., Irani, M., Basri, R.: Actions as space-time shapes. Pattern Analysis and Machine Intelligence (PAMI), IEEE Transactions on **29** (2007) 2247–2253
10. Esuli, A., Fagni, T., Sebastiani, F.: Treeboost. mh: A boosting algorithm for multi-label hierarchical text categorization. In: String Processing and Information Retrieval, Springer (2006) 13–24

11. Grossmann, E.: Adatree 2: boosting to build decision trees or improving adatree with soft splitting rules. unpublished work done at the Center for Visualisation and Virtual Environments, University of Kentucky (2004)
12. Grossmann, E.: Adatree: Boosting a weak classifier into a decision tree. In: Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04) Volume 6 - Volume 06. CVPRW '04, Washington, DC, USA, IEEE Computer Society (2004) 105–
13. Roe, B.P., Yang, H.J., Zhu, J., Liu, Y., Stancu, I., McGregor, G.: Boosted decision trees as an alternative to artificial neural networks for particle identification. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **543** (2005) 577–584
14. Zhang, Z., Xie, X.: Research on adaboost. m1 with random forest. In: Computer Engineering and Technology (ICCET), 2010 2nd International Conference on. Volume 1., IEEE (2010) V1–647
15. Zhou, S.K.: A binary decision tree implementation of a boosted strong classifier. In: Analysis and Modelling of Faces and Gestures. Springer (2005) 198–212
16. Seyedhosseini, M., Paiva, A.R., Tasdizen, T.: Fast adaboost training using weighted novelty selection. In: Neural Networks (IJCNN), The 2011 International Joint Conference on, IEEE (2011) 1245–1250
17. Schulter, S., Roth, P.M., Bischof, H.: Ordinal random forests for object detection. In: Proc. German Conference on Pattern Recognition (GCPR/DAGM). (2013)
18. Schulter, S., Wohlhart, P., Leistner, C., Saffari, A., Roth, P.M., Bischof, H.: Alternating decision forests. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2013)
19. Bernard, S., Adam, S., Heutte, L.: Dynamic random forests. Pattern Recognition Letters **33** (2012) 1580–1586
20. Viola, P., Jones, M.J.: Robust real-time face detection. Computer Vision, International Journal of **57** (2004) 137–154
21. Breiman, L.: Bagging predictors. In: Machine Learning. Volume 24. (1996) 123–140
22. Ho, T.K.: Random decision forests. In: Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on. Volume 1., IEEE (1995) 278–282
23. Ho, T.K.: The random subspace method for constructing decision forests. Pattern Analysis and Machine Intelligence, IEEE Transactions on **20** (1998) 832–844
24. Amit, Y., Geman, D.: Shape quantization and recognition with randomized trees. Neural computation **9** (1997) 1545–1588
25. Bernard, S., Adam, S., Heutte, L.: Using random forests for handwritten digit recognition. In: Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on. Volume 2., IEEE (2007) 1043–1047
26. Bernard, S., Heutte, L., Adam, S.: On the selection of decision trees in random forests. In: Neural Networks, 2009. IJCNN 2009. International Joint Conference on, IEEE (2009) 302–307
27. Fan, G., Wang, Z., Wang, J.: Cw-ssim kernel based random forest for image classification. In: Visual Communications and Image Processing 2010, International Society for Optics and Photonics (2010) 774425–774425
28. O'Hara, S., Draper, B.: Scalable action recognition with a subspace forest. In: Computer Vision and Pattern Recognition, (CVPR). IEEE Conference on. (2012)
29. Lui, Y.M., Beveridge, J., Kirby, M.: Action classification on product manifolds. In: Computer Vision and Pattern Recognition, (CVPR). IEEE Conference on. (2010)
30. Schindler, K., Van Gool, L.: Action snippets: How many frames does human action recognition require? In: Computer Vision and Pattern Recognition, (CVPR). IEEE Conference on. (2008)

31. Li, W., Yu, Q., Sawhney, H., Vasconcelos, N.: Recognizing activities via bag of words for attribute dynamics. In: Computer Vision and Pattern Recognition, (CVPR). IEEE Conference on. (2013) 2587–2594
32. Tian, Y., Sukthankar, R., Shah, M.: Spatiotemporal deformable part models for action detection. In: Computer Vision and Pattern Recognition, (CVPR). IEEE Conference on. (2013)
33. Yu, T.H., Kim, T.K., Cipolla, R.: Unconstrained monocular 3d human pose estimation by action detection and cross-modality regression forest. In: Computer Vision and Pattern Recognition, (CVPR). IEEE Conference on. (2013) 3642–3649
34. Jhuang, H., Serre, T., Wolf, L., Poggio, T.: A biologically inspired system for action recognition. In: Computer Vision (ICCV), 11th International Conference on, IEEE (2007) 1–8
35. Lin, Z., Jiang, Z., Davis, L.S.: Recognizing actions by shape-motion prototype trees. In: Computer Vision (ICCV), 12th International Conference on, IEEE (2009) 444–451
36. Yeffet, L., Wolf, L.: Local trinary patterns for human action recognition. In: Computer Vision and Pattern Recognition, (CVPR). IEEE Conference on. (2009)
37. Laptev, I., Marszalek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. In: Computer Vision and Pattern Recognition, (CVPR). IEEE Conference on. (2008)
38. Kihl, O., Picard, D., Gosselin, P.H., et al.: Local polynomial space-time descriptors for actions classification. In: International Conference on Machine Vision Applications. (2013)
39. Baumann, F., Liao, J., Ehlers, A., Rosenhahn, B.: Motion binary patterns for action recognition. In: 3rd International Conference on Pattern Recognition Applications and Methods (ICPRAM). (2014)
40. Wang, Z., Wang, J., Xiao, J., Lin, K.H., Huang, T.: Substructure and boundary modeling for continuous action recognition. In: Computer Vision and Pattern Recognition, (CVPR). IEEE Conference on. (2012)
41. Wu, X., Xu, D., Duan, L., Luo, J.: Action recognition using context and appearance distribution features. In: Computer Vision and Pattern Recognition, (CVPR). IEEE Conference on. (2011)
42. Li, R., Zickler, T.: Discriminative virtual views for cross-view action recognition. In: Computer Vision and Pattern Recognition, (CVPR). IEEE Conference on. (2012)