Cascaded Random Forest for Fast Object Detection*

Florian Baumann, Arne Ehlers, Karsten Vogt, and Bodo Rosenhahn

Institut für Informationsverarbeitung,
Leibniz Universität Hannover,
Appelstraße 9a, 30167 Hannover, Germany

Abstract. A Random Forest consists of several independent decision trees arranged in a forest. A majority vote over all trees leads to the final decision. In this paper we propose a Random Forest framework which incorporates a cascade structure consisting of several stages together with a bootstrap approach. By introducing the cascade, 99% of the test images can be rejected by the first and second stage with minimal computational effort leading to a massively speeded-up detection framework. Three different cascade voting strategies are implemented and evaluated. Additionally, the training and classification speed-up is analyzed. Several experiments on public available datasets for pedestrian detection, lateral car detection and unconstrained face detection demonstrate the benefit of our contribution.

1 Introduction

Random Forest was introduced in 2001 by Leo Breiman [1] on the basis of bagging [2] and is part of the ensemble learning methods competing with boosting [3]. A common problem of data mining or ensemble learning algorithms is the inability to handle imbalanced training data. An inequality between positive and negative class usually leads to inferior detection accuracy. Several researchers already started focusing on enhancing Random Forest's performance on imbalanced training data [4–6]. For instance Chen et al. [4] proposed a cost sensitive learning and sampling technique, more recently Khoshgoftaar et al. [5] presented an empirical study of learning with imbalanced training data. Simulations of Strobel et al. [6] reveal that Random Forest also tends to prefer the majority class. When using Random Forest for detection a vast amount of negative examples is needed to achieve a robust classifier and a low false positive rate. That leads to a strong inequality between positive and negative class resulting in a Random Forest that focuses on the majority class.

Furthermore, we observed a second drawback: after learning several trees Random Forest optimally adapts to the training set reaching perfect splits. Hence, the classifier can neither improve the detection sensitivity nor reduce the false positive rate.

Another common problem of object detection algorithms is the lack of real-time processing. Especially on mobile devices with limited computational power a fast, efficient and robust algorithm is required. In 2001 Viola and Jones proposed a real-time object detection algorithm [10] based on AdaBoost with a cascade structure. The cascade structure is motivated by the assumption that it is easier to reject a non-object than

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

J.-K. Kämäräinen and M. Koskela (Eds.): SCIA 2013, LNCS 7944, pp. 131–142, 2013.

Fig. 1. Overview of training images. Left: Daimler's pedestrian dataset [7]. Middle: Faces [8]. Right: UIUC's car dataset [9].

to find an object. Viola and Jones combined boosted strong classifiers in several independent stages with the condition that any stage can reject a subwindow but in order to accept a subwindow all stages need to be passed. Due to predominant rejection in early stages, computation time is radically reduced. Additionally, in order to ensure better training success Viola and Jones proposed a bootstrap strategy to delete correct classified negative images after learning a stage. Afterwards, the reduced negative training set is refilled with misclassified images [10]. Earlier work was done by Sung and Poggio [11]. The authors proposed a bootstrap mechanism that incrementally selects negative training examples with high benefit.

In this paper, we propose to combine a Random Forest with a cascade structure that arranges several decision trees in a stage with an aggregated bootstrap strategy. Three different cascade voting schemes are implemented and evaluated to achieve a tradeoff between computational speed and detection accuracy. The proposed approach is applied to a dataset for pedestrian detection, car detection and unconstrained face detection. Figure 1 shows some training images.

The paper is structured as follows. Section 1.1 gives an overview of recent contributions. Section 2 briefly describes the Random Forest algorithm by Leo Breiman while Section 3 contains the contribution of this paper. Experimental results and an analysis about the speed-up are shown in Section 4. Finally Section 5 concludes the paper.

1.1 Related Work

Since Random Forest is computationally efficient and achieves excellent results it is widely used in many domains such as pose recognition from depth images [12], skin detection [13], pedestrian detection [14], traffic sign recognition [15] or biomedical tasks [16]. Bosch and Zisserman applied Random Forest to several multi-class datasets for image classification [17].

More recently Gall et al. introduced Hough Forest which base on a Random Forest enhanced with a generalized Hough transform. Gall and Lempitsky applied a class-specific Hough Forest to object detection [18] achieving state-of-the-art results for several datasets. Furthermore, Gall et al. employed a Hough Forest in tracking, action recognition [19] and action classification [20].

Another approach is formulated by Felzenszwalb et al.[21]. The authors describe a general method for building a cascade from part-based models. Rematas and Leibe proposed a cascaded voting scheme [22] that reduces the computation effort without

losing accuracy. The main idea is that a single tree is used as an indicator to find regions-of-interest while the following trees only process the indicated regions.

In contrast, we incorporate a cascade into Random Forest and integrate a bootstrap framework to distribute huge quantities of images to stages. Hence, the learning success is increased and the drawback of imbalanced training data avoided. Furthermore we propose three different voting schemes to show a tradeoff between speed-up and low false-positive rate.

2 Random Forest

In this section the Random Forest algorithm by Leo Breiman [1] is briefly described. Random Forest is an ensemble learning method based on Breiman's bagging idea [2]. A 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 [3] that build a flat tree structure of decision stumps, a Random Forest uses an ensemble of 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

A tree is grown using the following algorithm:

1. Choose n_{tree} samples with M variables from N training samples at random.
2. The remaining samples are used to calculate the out-of-bag error (OOB-error).
3. At each node specify $m_{try} << M$ variables at random based on best split.
4. Completely grow the tree without pruning.

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 a feature vector $\boldsymbol{x}$:

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

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

$$I_{h_t(\boldsymbol{x})=c} = \begin{cases} 1, & h_t(\boldsymbol{x}) = c, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Classification. 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 (1).

3 Cascaded Random Forest

Our contribution consists of three main parts: (1) a cascade training and testing framework with three different voting strategies, (2) a weighting scheme for stage classifiers to penalize poorly performing stages and (3) a bootstrap strategy to incrementally construct new training sets. Figure 2 shows the pseudocode of a cascaded Random Forest training with weighted voting strategy.

A cascade consists of several stages with increasing complexity whereby each stage contains at least one tree. Trees are added until a given true positive and true negative rate is achieved. There are several advantages of a cascade structure: huge quantities of images can be distributed among stages, reduction of false positives and increasing computation speed at classification.

The sample's label is determined by the class probabilities of different trees arranged in S stages, therefore we extend Equation (1) to:

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

Furthermore we have observed that the detection accuracy of some stages is slightly lower than others. To decrease the influence of these poorly performing stages we calculate a weighting factor α for each stage. By incorporating the F_1 $(fmeasure)$[1] score into the weighting factor we exploit the harmonic mean of $precision$ and $recall$ on a validation set. Equation 3 is therefore extended to:

$$y(\boldsymbol{x}) = \underset{c}{\operatorname{argmax}} \left(\frac{1}{T \cdot \sum_{s=1}^{S} \alpha_s} \sum_{s=1}^{S} \alpha_s \sum_{t=1}^{T} I_{h_t(\boldsymbol{x})=c} \right), \tag{4}$$

$$\text{where } \alpha_s = \exp\left(\text{fmeasure}\right)$$

Alternatively, the OOB-error can be used to calculate the weighting factor. Moreover, α_s is linearly normalized to a range between 0 and 1. The weight of poorly performing stages is reduced so that their influence in the majority voting is decreased. Equations (3) and (4) do not take the early rejection of detections into account.
Our contribution comprises three different voting strategies:

1. Weighted voting: Each stage is weighted with respect to its single performance.
2. Full stage reject: Any stage can reject an image. To classify a positive object all stages need to be passed, similar to the Viola-detector [10]. Real-time processing and a reduction of false positive rates is achieved.
3. Majority stage reject: Each stage incorporates the votes of already processed stages. An image can only be accepted if the majority of stages pass it. A tradeoff between real-time processing, reduction of false positive rates and a higher true positive rates is achieved.

[1] $F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$. Best value at 1, worst at 0

Train Random Forest cascade with weighted voting

- Select maximum number of stages S and number of trees per stage T.
- Select true-positive tp and true-negative tn target rates per stage.
- *for* $(0 < s \leq S)$
  - *for* $(0 < t \leq T)$
    - Train single tree.
    - *if* $(tp_{s,t} > tp \;\&\; tn_{s,t} > tn)$ Stage completed.
  - Calculate $\alpha_s = \exp(\text{fmeasure})$.
  - Delete true-negatives
  - Refill with false-positives
- Normalize weights α to a range of 0 and 1

Fig. 2. The Random Forest cascade training with weighted voting. S stages are constructed each containing a maximum number of T trees.

Each voting scheme implies a bootstrap strategy to incrementally build a new beneficial training set, similar to [10, 11]. A commonly used bootstrap strategy proceeds as follows: after learning each stage all previously trained stages are reapplied to the negative training set in order to delete true negatives. In most cases all negative examples are deleted so that the training set only consists of positive images. In the next step all existing stages are reapplied to random images, at random position and scale, taken from a negative high-resolution dataset. If a generated subwindow is misclassified it will be added to a new negative training set to be learned by the following stage. By introducing a stage-wise training with a bootstrap strategy it is possible to uniformly distribute huge quantities of images to stages so that large differences between classes are avoided. This prevents the problem of imbalanced training sets [4, 5] and provides the maximal learning success for the Random Forest.

4 Experimental Results

We decided to evaluate three public available datasets for: (1) pedestrian detection, (2) (multi-class) lateral car detection and (3) (multi-class) face detection. Three Random Forest classifiers with weighted voting, full stage and majority rejection are trained. The results are compared to state-of-the-art methods. The influence of different parameter settings is also analyzed and an overview of training- and classification time consumption is given.

4.1 Pedestrian Detection

For our first experiment we used the Daimler mono pedestrian dataset by Munder and Gavrila [7]. The dataset provides three training sets and two test sets (T1 and T2),

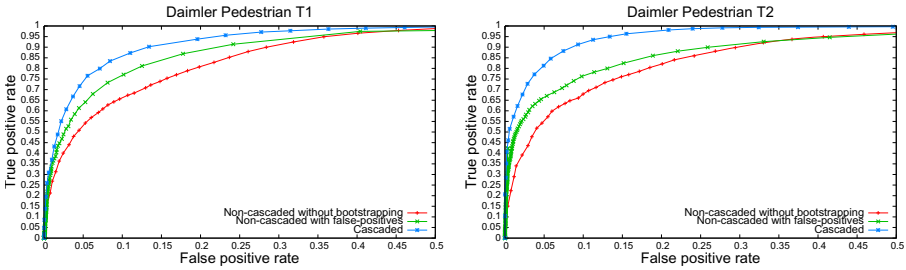

Fig. 3. ROC curves for the Daimler pedestrian datasets T1 and T2. We compare the cascaded classifier to a non-cascaded with equal training images and to a non-cascaded without additional images.

each consisting of 4800 pedestrians and 5000 non-pedestrians. Moreover, the dataset provides 3600 non-target images intended to extract additional negative patches. In order to be illumination and contrast invariant, all images have been normalized to zero mean and unit variance. A feature is then constructed from each normalized grayscale pixel value. Since the dataset does not need a sliding-window approach we decided to use the weighted voting strategy (see Section 3) without a rejection of subwindows. We trained a cascaded Random Forest classifier with 300 stages, each containing one tree. After training each stage true negatives were deleted and 4000 false positives were refilled leading to an overall negative training set of 1201000 distributed images. To obtain comparability and to show the drawback of imbalanced training data, the negative dataset was saved to train a non-cascaded classifier with 300 trees. Since this variant does not benefit from a stage-wise distribution of images the performance is presumably worse. Additionally, we trained another non-cascaded classifier without bootstrap sample.

Figures 3(a) and 3(b) show ROC curves for all three experiments on the T1 and T2 dataset in comparison. As expected, the cascaded classifier clearly outperforms both non-cascaded variants. Presumably the non-cascaded classifier with huge initial training set shows the impact of imbalanced training data and produces more false positives. The non-cascaded variant without additional images produces even more false positives, due to its smaller negative training set. In comparison to [23] we achieve better results than Viola-Jones OpenCV classifier and competitive results to control-point features.

4.2 Lateral Car Detection

As the second experiment we used the lateral car dataset collected by the Cognitive Computation Group of the University of Illinois [9]. It contains 1050 grayscale training images (550 cars and 500 non-car images). The images were mirrored and normalized to an overall set of 1600 images. The dataset also provides test images, ground truth data and an evaluation tool for an automatic precision and recall calculation. The multi-scale test set consists of 108 test images with 139 cars at different scales. In this experiment,

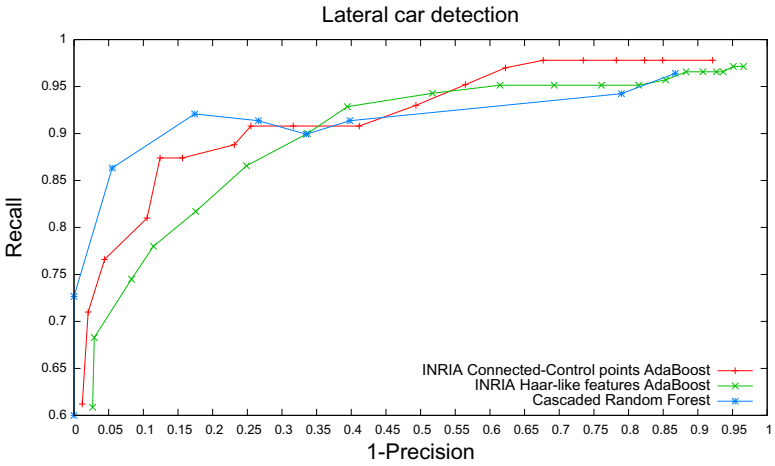

Fig. 4. ROC curve for the lateral car detection dataset. We compare the cascaded Random Forest to an AdaBoost classifier with connected-control-point features [23] and Haar-like features. For the important range of high precision the cascaded Random Forest shows superior performance. ROC line points are taken from [23].

each pixel value was used to construct a feature. The dataset needs a multi-scale sliding-window approach which requires a more complex classifier. We continue to use the weighted voting strategy but with 400 stages, each containing one tree. The bootstrap strategy remains the same. After each stage true negatives were deleted and 2000 false positives were refilled. The amount of false positive was reduced to coincide with the number of positive training images.

Figure 4 shows the ROC curve of the cascaded Random Forest classifier in comparison to control-point and Haar-like features trained by the AdaBoost algorithm of [23]. For a better comparison to other frameworks we use the 1-precision/recall metrics. Without any false detections the cascaded Random Forest already achieves a true-positive rate of 0.72 and with 25 false detections the classifier achieves a true-positive rate of 0.93. In comparison to [23] the cascaded Random Forest outperforms the control-point and Haar-like features for the important range of high precision (left part of ROC).

4.3 Unconstrained Face Detection

For training the face detector we used the ”MPLab GENKI-4K” dataset [8] from the Machine Perception Laboratory in California. It consists of 4000 faces under a wide range of illumination conditions. To achieve better results the training images were aligned, mirrored and normalized. A feature was constructed from each pixel value. We evaluated our face detector on the Face Detection Data Set and Benchmark [24] which consists of 2845 images with 5171 faces. The dataset contains several lateral or occluded faces and is partially difficult to classify, see Figure 6. The FDDB also includes a benchmark tool to compare the detector’s quality to different methods.

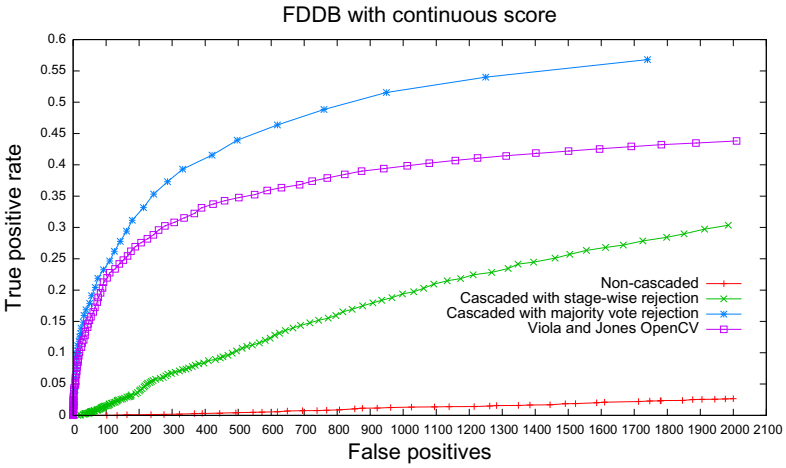

Fig. 5. ROC curve for Face Detection Data Set and Benchmark [24] with continuous score. The cascaded Random Forest with majority vote rejection clearly outperforms the other methods.

To achieve better detection accuracy for more complex datasets and real-time performance we evaluated both rejection voting schemes (see Section 3). As already mentioned, both strategies imply unweighted stages but differ in their rejection scheme. In the full stage rejection strategy, any stage can reject a subwindow (similar to Viola-detector) and in the majority rejection strategy, the votes of already processed stages are incorporated and subwindows are only classified negative if the majority of stages reject them. This leads to a tradeoff between performance and detection accuracy.

We trained a cascaded Random Forest with majority stage rejection and 1500 trees arranged in 1500 stages. After each stage true negatives were deleted and 15000 false positives were refilled. The Random Forest with full stage rejection was trained with 30 stages, each containing 50 trees and with similar bootstrap strategy. More than 30 stages led to an exceedingly strict classifier without any positive detections. The optimal parameters (amount of stages, trees and false positives) were empirically chosen. In both cases we noticed that nearly all negative examples were deleted after training one stage.

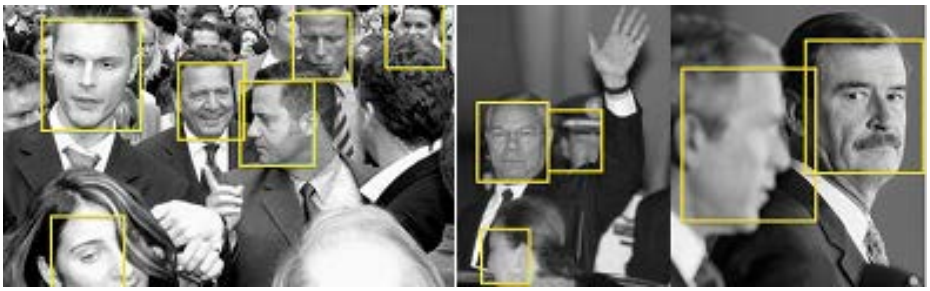

Fig. 6. Example detections on the FDDB [24]. Lateral or partially occluded faces are detected.

Figure 5 shows a ROC curve comparing two types of cascaded classifiers and a non-cascaded classifier to an OpenCV Viola-detector. The cascaded variant with majority rejection clearly outperforms the other methods. The full stage rejection strategy achieves a true positive rate of 0.30 at 2000 false positives, due to its smaller negative training set derived from only 30 stages. The non-cascaded classifier produces more false positives and only achieves similar true positive rates at the range of about 30000 false positives, its negative training set is too small. In comparison to state-of-the-art results [25] we achieve competing detection accuracy and compared to similar frameworks like the Viola and Jones OpenCV detector, the cascaded Random Forest clearly outperforms these frameworks.

Figure 6 shows some convincing face detections on FDDB. Typical boosting detectors can only detect faces if they exactly conform to the training images. The cascaded Random Forest classifier is more flexible, detecting also partially occluded or lateral faces.

4.4 Multi-class Capability

One of the advantages of Random Forest is the ability for multi-class detection in contrast to common boosting detectors. Thus, as a small experiment we trained a cascaded Random Forest with a third object class. In case of the face detector we manually labeled the training images as male or female and in case of the car detector as left-faced or right-faced car. The training procedure remains the same. Figure 7 shows some promising results. Neither the FDDB nor the car dataset provide groundtruth for such a case, hence finding adequate datasets to evaluate the capability of our proposed method for multi-class detection is planned for future work.

Fig. 7. Training of a cascaded Random Forest with a third object class. (Yellow): male face / left-faced car, (purple): female face / right-faced car.

Table 1. Speed-up at training time applied to the car detection dataset. The cascaded Random Forest requires significantly less time training due to the distribution of negative images among the stages.

Type	Images	Time [min]
Cascaded with 300 stages and 300 trees	1201000	**1.53**
Non-cascaded with 300 trees	1201000	821.16
Non-cascaded with 300 trees	4000	0.20

Table 2. Frames per second on the face detection dataset. The cascaded classifier with majority vote rejection achieves the highest frame rate.

Type	Trees , Stages	FPS
Cascaded with majority vote	1 , 1500	**15.63**
Cascaded with full stage rejection	50 , 30	6.24
Cascaded with majority vote	50 , 30	3.94
Non-cascaded	1500 , 1	0.08

4.5 Speed-Up

The classifiers were trained on an Intel Core i7-2700K. The framework is written in C/C++, partially parallelized (bootstrapping, testing) and not optimized. Table 1 shows training time comparing a cascaded and two different types of non-cascaded Random Forest applied to the lateral car dataset. The non-cascaded Random Forest differ in the amount of training images. In case of the cascaded variant the training time includes the deletion of true negatives and reapplying the classifier for collecting false positives. The huge difference between cascaded and the first non-cascaded Random Forest is caused by the large initial training dataset.

Table 2 shows the classification time on the face detection dataset. We compared a cascaded Random Forest with majority rejection to a classifier with full stage rejection and to a non-cascaded one. The cascaded classifier with 1500 stages and majority rejection achieves the highest frame rate. The full stage rejection variant should achieve the

Table 3. Stage rejections of face detection dataset (5 of 1500 stages). 99% of all generated subwindows are rejected by the first and second stage with minimal computational effort.

Stage Index	Rejected subwindows
1	**87.762557%**
2	**11.336876%**
3	0.0347972%
4	0.0072001%
5	0.0092063%

highest frame rate but due to 50 trees per stage and the resulting higher complexity the strategy achieves a lower frame rate. The non-cascaded classifier achieves the lowest frame rate since every generated subwindow needs to be classified by 1500 trees.

Table 3 shows the amount of rejected subwindows with respect to the processed stage. Nearly 90% of all generated subwindows are rejected by the first stage with minimal effort. About 99% are rejected by the first and second stage leading to an increasing computation time.

4.6 Discussion

The results demonstrate that an increasing number of stages comprising the bootstrap strategy lead to a more robust classifier achieving a better detection accuracy. Furthermore, we suggest to train each stage with a minor number of trees, in most cases only one tree is necessary to correctly classify all training examples. If it is required to reduce the number of false positives it is more applicable to increase the number of stages than the number of trees. Additionally, the speed-up analysis shows that the cascaded variants achieve highest frame rates. Finally, the cascaded Random Forest with majority vote rejection leads to best detection accuracy and highest frame rate.

5 Conclusion

In this paper we proposed a Random Forest framework which incorporates the cascade structure of the well-known Viola-detector. A bootstrapping mechanism allows to distribute huge negative training sets among the cascade stages. By this, the learning success can be increased and the problem of imbalanced training classes is reduced.

We composed three different voting strategies: weighted voting, full stage and majority stage rejection and show a tradeoff between real-time processing and low false positive rate. Experiments were conducted on public available datasets for pedestrian detection (Daimler), lateral car detection (UIUC) and unconstrained face detection (FDDB). The results demonstrate that we achieve real-time processing, state-of-the-art detection accuracy, multi-class capability as well as an accelerated training procedure.

References

1. Breiman, L.: Random forests. In: Machine Learning, vol. 45, pp. 5–32 (2001)
2. Breiman, L.: Bagging predictors. In: Machine Learning, vol. 24, pp. 123–140 (1996)
3. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: Proceedings of the Thirteenth International Conference on Machine Learning, pp. 148–156. IEEE (1996)
4. Chen, C., Liaw, A., Breiman, L.: Using random forest to learn imbalanced data. Technical report, Department of Statistics, University of Berkeley (2004)
5. Khoshgoftaar, T.M., Golawala, M., Hulse, J.V.: An empirical study of learning from imbalanced data using random forest. In: 19th International Conference on Tools with Artificial Intelligence (ICTAI), vol. 2, pp. 310–317. IEEE (2007)
6. Strobl, C., Boulesteix, A.L., Zeileis, A., Hothorn, T.: Bias in random forest variable importance measures: Illustrations, sources and a solution. BMC Bioinformatics (2007)

7. Munder, S., Gavrila, D.M.: An experimental study on pedestrian classification. Transactions on Pattern Analysis and Machine Intelligence (PAMI) 28, 1863–1868 (2006)
8. The MPLab GENKI Database, GENKI-4K Subset, http://mplab.ucsd.edu
9. Agarwal, S., Awan, A., Roth, D.: Learning to detect objects in images via a sparse, part-based representation. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 26(11), 1475–1490 (2004)
10. Viola, P., Jones, M.J.: Robust real-time face detection. International Journal of Computer Vision 57(2), 137–154 (2004)
11. Sung, K., Poggio, T.: Example based learning for view-based human face detection. Technical Report CBCL-112, Artificial Intelligence Laboratory, Massachusettes Inst. of Technology (1995)
12. Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A.: Real-time human pose recognition in parts from a single depth image. In: Computer Vision and Pattern Recognition (CVPR) (2011)
13. Khan, R., Hanbury, A., Stoettinger, J.: Skin detection: A random forest approach. In: 17th International Conference on Image Processing (ICIP), pp. 4613–4616. IEEE (2010)
14. Li, W., Xu, Z., Wang, S., Ma, G.: Pedestrian detection based on improved random forest in natural images. In: 3rd International Conference on Computer Research and Development (ICCRD), vol. 4, pp. 468–472. IEEE (2011)
15. Zaklouta, F., Stanciulescu, B., Hamdoun, O.: Traffic sign classification using k-d trees and random forests. In: The 2011 International Joint Conference on Neural Networks (IJCNN), pp. 2151–2155. IEEE (2011)
16. Díaz-Uriarte, R., De Andres, S.: Gene selection and classification of microarray data using random forest. BMC Bioinformatics 7, 3 (2006)
17. Bosch, A., Zisserman, A., Muoz, X.: Image classification using random forests and ferns. In: 11th International Conference on Computer Vision (ICCV), pp. 1–8. IEEE (2007)
18. Gall, J., Lempitsky, V.: Class-specific hough forests for object detection. In: Computer Vision and Pattern Recognition (CVPR). IEEE (2009)
19. Gall, J., Yao, A., Razavi, N., Van Gool, L., Lempitsky, V.: Hough forests for object detection, tracking, and action recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 33(11), 2188–2202 (2011)
20. Yao, A., Gall, J., Gool, L.V.: A hough transform-based voting framework for action recognition. In: Computer Vision and Pattern Recognition (CVPR), pp. 2061–2068. IEEE (2010)
21. Felzenszwalb, P., Girshick, R., McAllester, D.: Cascade object detection with deformable part models. In: Computer Vision and Pattern Recognition (CVPR), pp. 2241–2248. IEEE (2010)
22. Rematas, K., Leibe, B.: Efficient object detection and segmentation with a cascaded hough forest ISM. In: International Conference on Computer Vision Workshops (ICCVW), pp. 966–973. IEEE (November 2011)
23. Moutarde, F., Stanciulescu, B., Breheret, A.: Real-time visual detection of vehicles and pedestrians with new efficient adaboost features. In: IEEE IROS (2008)
24. Jain, V., Learned-Miller, E.: Fddb: A benchmark for face detection in unconstrained settings. Technical Report UM-CS-2010-009, University of Massachusetts, Amherst (2010)
25. Face Detection Data Set and Benchmark - Results, http://vis-www.cs.umass.edu/fddb/results.html