Decision Trees and Random Subwindows for Object Recognition

Raphaël Marée RAPHAEL.MAREE@ULG.AC.BE
Pierre Geurts PIERRE.GEURTS@ULG.AC.BE
Justus Piater JUSTUS.PIATER@ULG.AC.BE
Louis Wehenkel LOUIS.WEHENKEL@ULG.AC.BE
Department of EE & CS, Institut Montefiore, University of Liège, B-4000 Liège, Belgium

Abstract

In this paper, we compare five tree-based machine learning methods within our recent generic image-classification framework based on random extraction and classification of subwindows. We evaluate them on three publicly available object-recognition datasets (COIL-100, ETH-80, and ZuBuD). Our comparison shows that this general and conceptually simple framework yields good results when combined with ensembles of decision trees, especially when using Tree Boosting or Extra-Trees. The latter is particularly attractive in terms of computational efficiency.

1. Introduction

Object recognition is an important problem within image classification, which appears in many application domains. In the object recognition literature, local approaches generally perform better than global approaches. They are more robust to varying conditions because these variations can locally be modelled by simple transformations (Matas & Obdržálek, 2004). These methods are also more robust to partial occlusions and cluttered backgrounds. Indeed, the correct classification of all local features is not required to correctly classify one image. These methods are generally based on region detectors (Mikolajczyk et al., 2005) and local descriptors (Mikolajczyk & Schmid, 2005) combined with nearest-neighbor matching.

In this paper, we compare five tree-based machine learning methods within the generic image classification framework that we proposed in earlier work (Marée et al., 2005). It is based on random extraction of subwindows (square patches) and their classification by decision trees.

Appearing in *Proceedings of the workshop Machine Learning Techniques for Processing Multimedia Content*, Bonn, Germany, 2005.

2. Framework

In this section, we briefly describe the framework proposed by (Marée et al., 2005). During the training phase, subwindows are randomly extracted from training images (2.1), and a model is constructed by machine learning (2.2) based on transformed versions of these (Figure 1). Classification of a new test image (2.3) similarly entails extraction and description of subwindows, and the application of the learned model to these subwindows. Aggregation of subwindow predictions is then performed to classify the test image, as illustrated in Figure 2. In this paper, we evaluate various tree-based methods for learning a model.

2.1. Subwindows

The method extracts a large number of possibly overlapping, square subwindows of random sizes and at random positions from training images. Each subwindow size is randomly chosen between 1×1 pixels and the minimum horizontal or vertical size of the current training image. The position is then randomly chosen so that each subwindow is fully contained in the image. By randomly selecting a large number (N_w) of subwindows, one is able to cover large parts of images very rapidly. This random process is generic and can be applied to any kind of images. The same random process is applied to test images. Subwindows are resized to a fixed scale (16×16 pixels) and transformed to a HSV color space. Each subwindow is thus described by a feature vector of 768 numerical values. The same descriptors are used for subwindows obtained from training and test images.

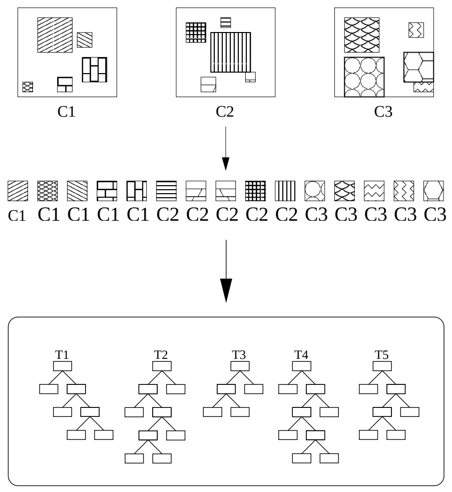

Figure 1. Learning: the framework first randomly extracts multi-scale subwindows from training-set images, then resizes them and builds an ensemble of decision trees.

2.2. Learning

At the learning phase, a model is automatically built using subwindows extracted from training images. First, each subwindow is labelled with the class of its parent image. Then, any supervised machine learning algorithm can be applied to build a subwindow classification model. Here, the input of a machine learning algorithm is thus a training sample of N_w subwindows, each of which is described by 768 real-valued input variables and a discrete output class (Figure 1). The learning algorithm should consequently be able to deal efficiently with a large amount of data, first in terms of the number of subwindows and classes of images in the training set, but more importantly in terms of the number of values describing these subwindows.

In this context, we compare five tree-based methods: one single-tree method based on CART (Breiman et al., 1984), and four ensemble methods: Bagging (Breiman, 1996), Boosting (Freund & Robert Schapire, 1996), Random Forests (Breiman, 2001), and Extra-Trees (Geurts, 2002). Extra-Trees only were originally used by (Marée et al., 2005).

2.3. Recognition

In this approach, the learned model is used to classify subwindows of a test image. To make a prediction for a test image with an ensemble of trees grown from subwindows, each subwindow is simply propagated into each tree of the ensemble. Each tree outputs conditional class probability estimates for each subwindow. Each subwindow thus receives T class probability estimate vectors where T denotes the number of trees in the ensemble. All the predictions are then averaged and the class corresponding to the largest aggregated probability estimate is assigned to the image. Note that we will simply consider that one single tree method is a particular case where $T = 1$.

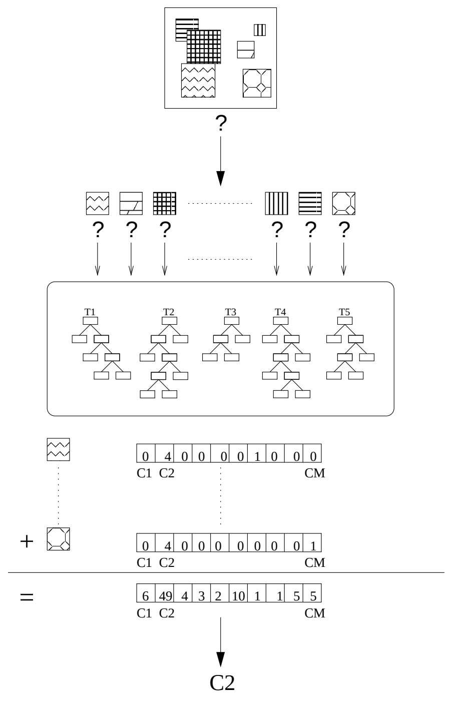

Figure 2. Recognition: randomly-extracted subwindows are propagated through the trees (here $T = 5$). Votes are aggregated and the majority class is assigned to the image.

3. Experiments

Our experiments aim at comparing decision tree methods within our random subwindow framework (Marée et al., 2005). To this end, we compare these methods on three well-known and publicly available object recognition datasets: household objects in a controlled environment (COIL-100), object categories in a controlled environment (ETH-80), and buildings in urban scenes (ZuBuD). The first dataset exhibits substantial viewpoint changes. The second dataset also exhibits higher intra-class variability. The third dataset contains images with illumination, viewpoint, scale and orientation changes as well as partial occlusions and cluttered backgrounds.

3.1. Parameters

For each problem and protocol, the parameters of the framework were fixed to $N_w = 120000$ learning subwindows, $T = 25$ trees, and $N_{w,test} = 100$ subwindows are randomly extracted from each test image. In (Marée et al., 2005), the parameters were fixed to $N_w = 120000$, $T = 10$, and $N_{w,test} = 100$. Ensemble methods are influenced by the number of trees T that are aggregated. Usually, the more trees are aggregated, the better the accuracy. We will further evaluate the influence of these parameters in Section 4.

For each machine learning method within the framework, the values of several parameters need to be fixed. In our experiments, single decision trees are fully developed, i.e. without using any pruning method. The score used to evaluate tests during the induction is the score proposed by (Wehenkel, 1997) which is a particular normalization of the information gain. Otherwise our algorithm is similar to the CART method (Breiman et al., 1984).

Random Forests depends on an additional parameter k which is the number of attributes randomly selected at each test node. In our experiments, its value was fixed to the default value suggested by the author of the algorithm which is the square root of the total number of attributes. According to (Breiman, 2001) this value usually gives error rates very close to the optimum.

With the latest variant of Extra-Trees (Geurts et al., 2005), the parameter k is the number of attributes randomly selected at each test node. We fixed it to the default value which is the square root of the total number of attributes. The main differences with Random Forests are that the algorithm randomizes also cut-point choice while splitting a tree node and grows the tree from the whole learning set while Random Forests uses bootstrap sampling.

Boosting does not depend on another parameter but it nevertheless requires that the learning algorithm does not give perfect models on the learning sample (so as to provide some misclassified instances). Hence, with this method, we used with decision trees the stop-splitting criterion described by (Wehenkel, 1997). It uses a hypothesis test based on the G^2 statistic to determine the significance of a test. In our experiments, we fixed the nondetection risk α to 0.005.

3.2. COIL-100

COIL-100[1] (Murase & Nayar, 1995) is a dataset of 128×128 color images of 100 different 3D objects (boxes, bottles, cups, miniature cars, etc.). Each object was placed on a motorized turntable and images were captured by a fixed camera at pose intervals of 5°, corresponding to 72 images per object. Given a new image, the goal is to identify the target object in it.

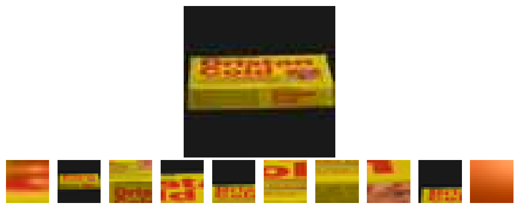

Figure 3. COIL-100: some subwindows randomly extracted from a test image and resized to 16×16 pixels.

On this dataset, reducing the number of training views increases perspective distortions between learned views and images presented during testing. In this paper, we evaluate the robustness to viewpoint changes using only one view (the pose at 0°) in the training sample while the remaining 71 views are used for testing. Using this protocol, methods in the literature yield error rates from 50.1% to 24% (Matas & Obdržálek, 2004). Our results using this protocol (100 learning images, 7100 test images) are reported in Table 1. Tree Boosting is the best method for this problem, followed by Extra-Trees, Tree Bagging, and Random Forests. One decision tree has a higher error rate. Examples of subwindows randomly extracted and resized to 16×16 pixels are given in Figure 3.

3.3. ETH-80

The Cogvis ETH-80 dataset[2] contains 3280 color images (128×128 pixels) of 8 distinct object categories (apples, pears, tomatoes, cows, dogs, horses, cups, cars). For each category, 10 different objects are provided. Each object is represented by 41 images from different viewpoints.

In our experiments, we used for each category 9 objects in the learning set ($8*9*41 = 2952$ images), and the remaining objects in the test set ($8*1*41 = 328$ images). We evaluate the methods on 10 different partitions, and the mean error rate is reported in Table 1. Here, Extra-Trees are slightly inferior while Tree Boosting and Tree Bagging are slightly better than other methods.[3]

[1] `http://www.cs.columbia.edu/CAVE/`

[2] `http://www.vision.ethz.ch/projects/categorization/eth80-db.html`

[3] We observed that the adjustment of the Extra-Tree parameter k to the half of the total number of attributes, instead of the square root, yields a 20.85% mean error rate. Such improvements might also be obtained for Random Forests.

Table 1. Classification error rates (in %) of all methods on COIL-100, ETH-80, ZuBuD ($T = 25$, $N_w = 120000$, $N_{w,test} = 100$)).

Methods	COIL-100	ETH-80	ZuBuD
RW+Single Tree	19.20	22.04	10.43
RW+Extra-Trees	11.53	22.74	4.35
RW+R. Forests	13.06	21.31	4.35
RW+Bagging	12.77	20.34	3.48
RW+Boosting	10.75	20.27	3.48

3.4. ZuBuD

The ZuBuD dataset[4] (Shao et al., 2003) is a database of color images of 201 buildings in Zürich. Each building in the training set is represented by five images acquired at five random arbitrary viewpoints. The training set thus includes 1005 images, while the test set contains 115 images of a subset of the 201 buildings. Images were taken by two different cameras in different seasons and under different weather conditions, and thus contain a substantial variety of illumination conditions. Partial occlusions and cluttered background are naturally present (trees, skies, cars, trams, people, . . .) as well as scale and orientation changes due to the position of the photographer. Moreover, training images were captured at 640×480 while testing images are at 320×240 pixels.

About five papers have so far reported results on this dataset that vary from a 59% error rate to 0% (Matas & Obdržálek, 2004). Our results are reported in Table 1. Due to the small size of the test set, the difference between the methods is not dramatic and only one image makes the difference between the two best ensemble methods (Tree Boosting, Tree Bagging) and the two others (Extra-Trees and Random Forests). One single decision tree is again inferior. Figure 4 shows the 5 images misclassified by Extra-Trees and Random Forests, while the last one is correctly classified by Tree Boosting and Tree Bagging. For this last image, the correct class is ranked second by Extra-Trees and Random Forests.

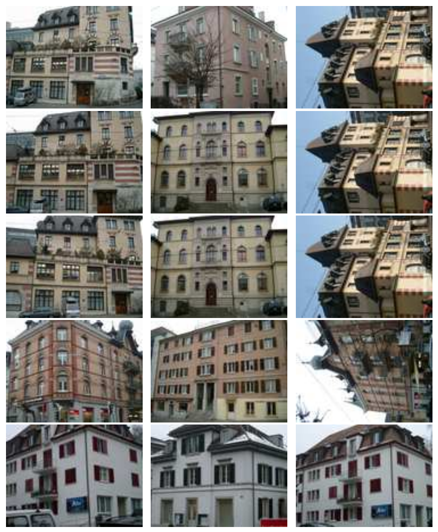

Figure 4. ZuBuD: misclassified test images (left), training images of predicted class buildings (middle), training images of correct buildings (right).

4. Discussion

The good performance of this framework was explained by (Marée et al., 2005) by the combination of simple but well-motivated techniques: random multi-scale subwindow extraction, HSV pixel representation and recent advances in machine learning that have produced new methods that are able to handle problems of high dimensionality.

For real-world applications, it may be useful to tune the framework parameters if a specific tradeoff between accuracy, memory usage and computing times is desired. Then, in this section, we discuss the influence of the framework parameters (4.1, 4.2, 4.3) on the ZuBuD problem which exhibits real-world images, and we present some complexity results (4.4).

4.1. Variation of N_w

Figure 5 shows that the error rate decreases monotonically with number of learning subwindows (for a given number of trees ($T = 25$) and a given number of test subwindows ($N_{w,test} = 100$)). For all methods, we observe that using $N_w = 60000$ subwindows already gives good results, and that $N_w = 180000$ does not improve accuracy, except for one single decision tree.

4.2. Variation of T

Figure 6 shows that the error rate decreases monotonically with the number of trees, for a given number of training subwindows ($N_w = 120000$) and test subwindows ($N_{w,test} = 100$). We observe that using $T = 10$ trees is already sufficient for this problem for all ensemble methods.

[4]`http://www.vision.ee.ethz.ch/showroom/zubud/index.en.html`

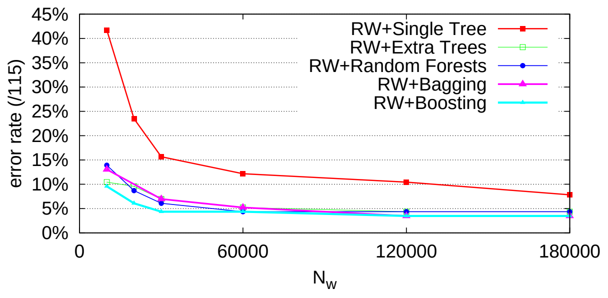

Figure 5. ZuBuD: error rate with increasing number of training subwindows.

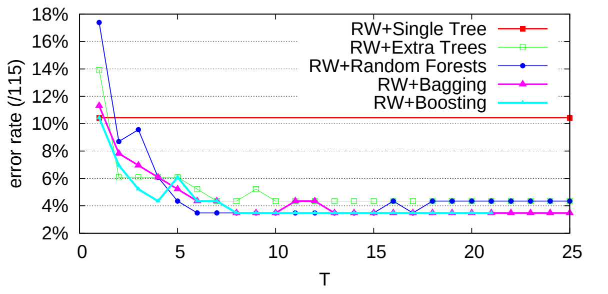

Figure 6. ZuBuD: error rate with increasing number of trees.

4.3. Variation of $N_{w,test}$

Figure 7 shows that the number of test subwindows also influences the error rate in a monotonic way, for a given number of training subwindows ($N_w = 120000$) and a given number of trees ($T = 25$). We observe that using $N_{w,test} = 25$ is already sufficient for this problem with ensemble methods, but the aggregation of more subwindows is needed for a single decision tree.

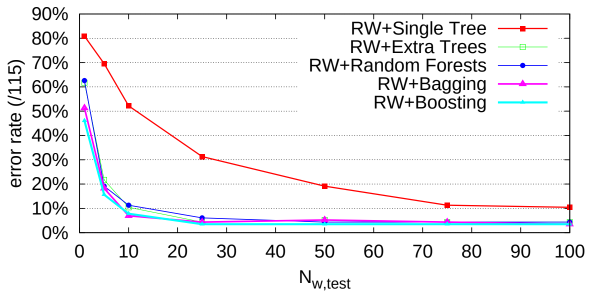

Figure 7. ZuBuD: error rate with increasing number of test subwindows.

4.4. Some notes on complexity

Our current implementation cannot be considered optimal but some indications can be given about memory and running-time requirements. With this framework, original training images and their subwindows are not necessary to classify new images after the construction of the model, contrary to classification methods based on nearest neighbors. Here, only the ensemble of trees is used for recognition.

Table 2. ZuBuD: average tree complexity and learning time.

Methods	Cmplx	Learning Time
RW+Single Tree	92687	3h36m30s
RW+Extra-Trees	148080	14m05s
RW+Random Forests	77451	2h14m54s
RW+Bagging	63285	53h35m46s
RW+Boosting	28040	54h21m31s

Learning times for one single decision tree and ensembles of $T = 25$ trees are reported in Table 2, considering that subwindows are in main memory. The complexity of tree-based method induction algorithm is of order $O(N_w log N_w)$. Extra-Trees are particularly fast due to their extreme randomization of both attributes and cut-points while splitting a tree node. Single tree complexity (number of nodes) is also given in Table 2 as a basic indication of memory usage.

To classify a new image, we observed that the prediction of one test subwindow with one tree requires on average less than 20 tests (each of which involves comparing the value of a pixel to a threshold), as reported in Table 3[5]. The minimum and maximum depths are also given. To classify one unseen image, the number of operations is thus multiplied by T, the number of trees, and by $N_{w,test}$, the number of subwindows extracted. The time to add all votes and search the maximum is negligible. Furthermore, extraction of one subwindow is very fast because of its random nature.

On this problem, we have also observed that pruning Extra-Trees could substantially reduce their complexity (downto a tree complexity average of 25191 with the same stop-splitting criterion as Tree Boosting, thus giving an average test depth of 15.4) while keeping the same accuracy. In practical applications where prediction times are essential, the use of pruning is thus certainly worth exploring.

[5]The average tree depth was calculated empirically over the 287500 propagations (100 subwindows for each of the 115 test images, propagated through $T = 25$ trees), except for one single decision tree and for Tree Boosting (because the algorithm stopped after $T = 21$ trees).

Table 3. ZuBuD: average subwindow test depth.

Methods	Depth	min	max
RW+Single Tree	16.59	9	29
RW+Extra-Trees	18.26	8	34
RW+Random Forests	16.44	7	33
RW+Bagging	15.98	8	34
RW+Boosting	15.04	6	28

5. Conclusions

In this paper, we compared 5 tree-based machine learning methods within a recent and generic framework for image classification (Marée et al., 2005). Its main steps are the random extraction of subwindows, their transformation to normalize their representation, and the supervised automatic learning of a classifier based on (ensembles of) decision tree(s) operating directly on the pixel values. We evaluated the tree-based methods on 3 publicly-available object recognition datasets. Our study shows that this general and conceptually simple framework yields good results for object recognition tasks when combined with ensembles of decision trees. Extra-Trees are particularly attractive in terms of computational efficiency during learning, and are competitive with other ensemble methods in terms of accuracy. This method with its default parameter allows to evaluate very quickly the framework on any new dataset.[6] However, if the main objective of a particular task is to obtain the best error rate whatever the learning time, Tree Boosting appears to be a better choice. Tuning the parameters (such as the value of k in Extra-Trees, or the stop-splitting criterion) might further improve the results.

For future work, it would be interesting to perform a comparative study with SVMs. The framework should also be evaluated on bigger databases in terms of the number of images and/or classes and with images that exhibit higher intra-class variability and heavily cluttered backgrounds (such as the Caltech-101[7], Birds, or Butterflies[8] datasets).

6. Acknowledgment

Raphaël Marée is supported by GIGA-Interdisciplinary Cluster for Applied Genoproteomics, hosted by the University of Liège. Pierre Geurts is a Postdoctoral Researcher at the National Fund for Scientific Research (FNRS, Belgium).

[6]Java implementation is available for evaluation at `http://www.montefiore.ulg.ac.be/~maree/`

[7]`http://www.vision.caltech.edu/feifeili/Datasets.htm`

[8]`http://www-cvr.ai.uiuc.edu/ponce_grp/data/`

References

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*, 123–140.

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

Breiman, L., Friedman, J., Olsen, R., & Stone, C. (1984). *Classification and regression trees.* Wadsworth International (California).

Freund, Y., & Robert Schapire, E. (1996). Experiments with a new boosting algorithm. *Proc. Thirteenth International Conference on Machine Learning* (pp. 148–156).

Geurts, P. (2002). *Contributions to decision tree induction: bias/variance tradeoff and time series classification.* Doctoral dissertation, Department of Electrical Engineering and Computer Science, University of Liège.

Geurts, P., Ernst, D., & Wehenkel, L. (2005). Extremely randomized trees. *Submitted.*

Marée, R., Geurts, P., Piater, J., & Wehenkel, L. (2005). Random subwindows for robust image classification. *Proc. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR).*

Matas, J., & Obdržálek, S. (2004). Object recognition methods based on transformation covariant features. *Proc. 12th European Signal Processing Conference (EUSIPCO 2004).* Vienna, Austria.

Mikolajczyk, K., & Schmid, C. (2005). A performance evaluation of local descriptors. *PAMI, to appear.*

Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., & Gool, L. V. (2005). A comparison of affine region detectors. *International Journal of Computer Vision, to appear.*

Murase, H., & Nayar, S. K. (1995). Visual learning and recognition of 3d objects from appearance. *International Journal of Computer Vision*, *14*, 5–24.

Shao, H., Svoboda, T., & Van Gool, L. (2003). *Zubud - Zurich building database for image based recognition* (Technical Report TR-260). Computer Vision Lab, Swiss Federal Institute of Technology, Switzerland.

Wehenkel, L. A. (1997). *Automatic learning techniques in power systems.* Kluwer Academic Publishers, Boston.