A GENERIC APPROACH FOR IMAGE CLASSIFICATION BASED ON DECISION TREE ENSEMBLES AND LOCAL SUB-WINDOWS

Raphaël Marée, Pierre Geurts, Justus Piater, and Louis Wehenkel

Department of Electrical Engineering and Computer Science,
University of Liège, Belgium
Raphael.Maree@ulg.ac.be

ABSTRACT

A novel and generic approach for image classification is presented. The method operates directly on pixel values and does not require feature extraction. It combines a simple local sub-window extraction technique with induction of ensembles of extremely randomized decision trees. We report results on four well known and publicly available datasets corresponding to representative applications of image classification problems: handwritten digits (MNIST), faces (ORL), 3D objects (COIL-100), and textures (OUTEX). A comparison with studies from the computer vision literature shows that our method is competitive with the state of the art, an interesting result considering its generality and conceptual simplicity. Further experiments are carried out on the COIL-100 dataset to evaluate the robustness of the learned models to rotation, scaling, or occlusion of test images. These preliminary results are very encouraging.

1. INTRODUCTION

Image classification is an important problem which is particularly difficult for traditional machine learning algorithms mainly because of the high number of input variables that may describe images (i.e. pixels). Indeed, with a high number of variables, these methods tend to produce very unstable models with low generalization performance (for decision trees, see [1]). Furthermore, computing times can also be detrimental in such extreme conditions. To handle this high dimensionality, image classification systems usually rely on a pre-processing step, specific to the particular problem and application domain, which aims at extracting a reduced set of interesting features from the initially huge number of pixels. This reduced set is then used as new input variables for traditional learning algorithms, possibly tuned for the specific application. The limitation of this approach is clear: When considering a new problem or application domain, it is necessary to manually adapt the pre-processing step by taking into account the specific characteristics of the new application. But, at the same time, recent advances in automatic learning have produced new methods that are able to handle more and more complex problems without requiring any a priori information about the application. These methods are increasingly competitive with methods specifically tailored for these domains.

In this paper, we evaluate a recent and generic machine learning algorithm based on decision tree ensembles that has been shown to perform remarkably well on a variety of tasks [1]. Motivated by the problem of image classification, we then introduce an extension of this algorithm that augments its generality even further. Both forms of the algorithm operate directly on the pixel values and do not extract any task-specific features. To demonstrate the performance of the algorithms, and of the extended version in particular, we have chosen four typical problems in image classification, and have ran exactly the same algorithm on each of them. In all four cases, the extended algorithm produces results competitive with the state of the art.

In section 2, we formally define the problem of image classification and we describe the two parts of our generic algorithm: extremely randomized trees (extra-trees) and local sub-window extraction. Section 3 presents the results of our experiments. The accuracy of our algorithm is compared to previous results on four datasets presented in the literature. Finally, the robustness of the method to rotation, scaling, and occlusion is analysed on the COIL-100 problem.

2. PROPOSED ALGORITHMS

The input of a generic learning algorithm for image classification is a training set of pre-classified images,

$$LS = \{(A^i, c^i), i = 1, \dots, N\}$$

where A^i is a $W_x \times W_y$ matrix describing the image and $c^i \in \{1, \dots, M\}$ is its classification (among M classes). The elements $a^i_{k,l}$ of A^i ($k = 1, \dots, W_x, l = 1, \dots, W_y$) describe image pixels at location (k, l) by means of an integer value in the case of grey level images or by means of 3 RGB values in the case of color images.

To handle information from pixels without any pre-processing, the learning algorithm should be able to deal efficiently with a large amount of data, first in terms of the number of images and classes of images in the learning set, but more importantly in terms of the number of values describing these images (the attributes). Assuming for example that $W_x = W_y = 128$, there are already $128 * 128 = 16384$ integer values describing images and this number is further multiplied by 3 if colors are taken into account.

In the next section, we present a recent method that is based on ensembles of decision trees. Several obvious shortcomings of this basic algorithm are simultaneously addressed by adding a wrapper technique that consists in classifying local sub-windows extracted from the original images. This latter variant is described in Subsection 2.2.

2.1. Ensemble of Extremely Randomized Trees

Ensemble methods are very popular in machine learning. These methods consist in improving an existing learning algorithm by combining the predictions of several models. They are very effective when used with decision trees that otherwise are often not competitive in terms of accuracy with other learning algorithms. We think that ensemble methods based on decision trees are a good starting point for designing a generic system for image classification. They do not make any a priori assumption about the application problem, they have been successfully applied to many complex problems in various application domains (see e.g. [2] for some recent applications) and, moreover, they compare very favorably with other state-of-the-art algorithms (e.g. Support Vector Machines [3]). In this context, we first propose to apply a particular ensemble method for decision trees to image classification problems. The method consists in building many extremely randomized trees (extra-trees). It was first proposed in [1].

Extremely randomized trees have the same structure as classical decision trees [4] but the induction algorithm is different. In the classical induction algorithm, the tree is grown in a top-down fashion by using the learning examples and searching at each node for the test that maximizes a score measure (e.g. a normalization of Shannon information [5] that evaluates the ability of a test to separate instances of the current learning subset). On the contrary, in the extremely randomized induction algorithm [1], a tree is grown by selecting at each node the test attribute fully at random and its threshold is chosen randomly around the mean of its current values. In the context of image classification, this yields the very simple recursive function shown in Table 1 where LS is initialized with all the learning examples [1]. Tests at the internal nodes are of the form $[a_{k,l} < a_{th}]$ that compare the value of the pixel at position (k, l) to a threshold a_{th}. Several extra-trees are then built from the same learning sample (in practice as many as possible) and, to make a prediction for an image, we propagate successively the entire image into all the trees and we assign to the image the majority class among the classes given by the trees.

Experiments in [6] have shown that this method compares favorably with other ensemble methods with trees on typical machine learning databases. Its good accuracy is explained by a bias/variance analysis in [1]. Briefly, since each tree is fully grown until it perfectly classifies the learning sample, the method has a low bias. At the same time, the randomization yields a set of trees that are as uncorrelated as possible, which yields a small variance when their predictions are aggregated. Among tree ensemble methods, the extremely randomized algorithm presented here is also especially interesting in the context of images because of its computational efficiency. Indeed, to a certain extent [2], its complexity is not related to the number of pixels in the image since at each tree node, an attribute and a threshold are selected at random. Assuming that the trees are approximately balanced, tree depth will be close to $\log N$. Since the development of one level requires order N operations, the complexity of the algorithm is thus $N \log N$ with respect to the number of images in the learning set, independently of the number of pixels. Furthermore, the prediction of one unseen image with one tree requires on average only $\log N$ tests (each of which involves comparing the value of a pixel to a threshold) and hence remains very fast even for very large learning set sizes.

[1] In this algorithm, the threshold on the score measure, s_{th}, ensures that a test is quite discriminating despite its random selection. Indeed, using this threshold, if a random test is not able to separate well instances, then another random test is considered. In all our experiments, this threshold will be fixed to 0.1 (which is 10% of the maximal possible value of the score).

[2] In practice, several random selections are sometimes necessary to find an informative test according to the score threshold. But, usually, their number is small with respect to the total number of attributes.

```
**Build_extra_tree**($LS$):
  • If $LS$ contains images all of the same class, return a leaf
    with this class associated to it;
  • Otherwise:
      1. Set $[a_{k,l} < a_{th}]$=Choose_a_random_split($LS$);
      2. Split $LS$ into $LS_{left}$ and $LS_{right}$ according to
         the test $[a_{k,l} < a_{th}]$ and build the subtrees
         $\mathcal{T}_{left}$ = build_extra_tree($LS_{left}$) and $\mathcal{T}_{right}$ =
         build_extra_tree($LS_{right}$) from these subsets;
      3. Create a node with the test $[a_{k,l} < a_{th}]$, attach $\mathcal{T}_{left}$
         and $\mathcal{T}_{right}$ as successors of this node and return the
         resulting tree.

**Choose_a_random_split**($LS$):
  1. Select a pixel location $(k, l)$ at random;
  2. Select a threshold $a_{th}$ at random according to a distribution
     $N(\mu_{k,l}, \sigma_{k,l})$, where $\mu_{k,l}$ and $\sigma_{k,l}$ are respectively the
     mean and standard deviation of the pixel values $a_{k,l}$ in $LS$;
  3. If the score of this test is greater than a given threshold $s_{th}$,
     return the test $[a_{k,l} < a_{th}]$;
  4. Otherwise, return to step 1 and select a different location. If
     all locations have already been considered, return the best
     test so far.
```

Table 1. Extra-tree algorithm for image classification

2.2. Local Sub-window Extraction

Even though the previous method handles a large number of input variables efficiently, the number of tests on pixels (that are combined along a path from the root node to a terminal node) in a tree is limited by the size of the learning set. In practice, when the number of images available for learning is small compared to the total number of pixels, the algorithm cannot combine enough tests on pixels to provide acceptable models. Furthermore, from the point of view of computer vision, this algorithm is a global approach to image classification as it operates on the entire image without focusing on local appearances. Limitations of global approaches are well known in image classification, including sensitivity to occlusions, scale changes and different viewpoints or orientations. Another drawback of this global method is that it does neither consider possible repetition of small patterns nor their unspecific image locations.This is particularly true in texture images but this must also be considered to ensure translation invariance.

To address those shortcomings, we have adopted another generic approach that is popular in image classification (e.g. [7], [8], and [9]). It consists in building models from several local subwindows extracted from original images of the learning set. However, our approach is different from those cited above: In order to be efficient and generic, our extraction algorithm is random and does not require filtering of informative windows. Indeed, to a certain extent, determining which image pixels discriminate over the entire set of images will implicitly be done by the tree induction algorithm.

In this variant, the induction of extra-trees is then carried out in two simple steps, given a window size $w_1 \times w_2$ and a number N_w:

- Extract N_w sub-windows at random from training set images (by first selecting an image at random from LS and then selecting a sub-window at a random location in this image) and assign to each sub-window the classification of its parent image;
- Grow each extra-trees to classify these N_w sub-windows by using all the $w_1 * w_2$ pixel values that characterize them.

To make a prediction for an image with an ensemble of extra-trees grown from sub-windows, the following procedure is used:

- Propagate successively all possible sub-windows of size $w_1 \times w_2$ from this image into the ensemble of extra-trees;
- Assign to the image the majority class among the classes assigned to the sub-windows.

Because in our experiments N_w will be greater than the original learning set size N, this variant will increase the computing times needed to build a model. Also, since testing an image entails testing all of its sub-windows, prediction time will substantially increase.

3. EXPERIMENTS

In this section, we carry out two runs of experiments. The first run aims at showing the generality of our approach. To this end, we tested it on four well-known and publicly available datasets corresponding to typical classification problems: recognition of handwritten characters and particularly digits (MNIST), faces (ORL), objects (COIL-100), and textures (OUTEX). The main characteristics of those datasets are summarized in Table 2 and an overview of their images is proposed by Figure 1. Our results are presented in Section 3.1 as well as a comparison with specific approaches found in the literature. Significance with respect to the state of the art is high as we stricly followed public protocols. In Section 3.2, the second run of experiments (on COIL-100 only) aims at analysing in more detail the algorithm by studying its robustness to learning set composition, rotation, scaling, and occlusion of test images.

3.1. Evaluation of Accuracy

The two variants (i.e. extra-trees and extra-trees on local sub-windows) have been tried on the four datasets. In both cases, the values of some parameters need to be fixed. Extra-trees are only influenced by one parameter: the number of trees T. However, as extremely randomized trees are grown independently, the more trees are aggregated, the better. So, in our study, we have used for each problem a number of trees that stabilizes error rates on the test samples. In the case of extra-trees, a large number of trees is usually necessary (from 50 on MNIST to 500 on ORL and COIL-100), while, with sub-windows, the error rate is stabilized much sooner (10 trees are sufficient in all problems). For extra-trees with sub-windows, additional parameters are the size of sub-windows $w1 \times w2$ and the number N_w of them which are extracted during the learning phase. Like for the number of trees in the ensemble, accuracy appears to be a monotonically increasing function of N_w On the last three problems, N_w was fixed to 120000. On MNIST, as the initial learning sample size is already quite large, we further

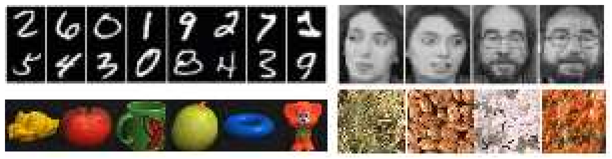

Fig. 1. Overview of the four databases: MNIST, ORL, COIL-100, OUTEX

increase this number to 360000. Note that, in all cases, a larger value of N_w could further improve the accuracy but could become detrimental in terms of computing times. Moreover, accuracy is very much influenced by the size of sub-windows. The optimal size is problem-dependent. In our experiments, this latter parameter has been tuned manually on each problem. Nevertheless, cross-validation could be used to determine its value automatically.

Table 3 summarizes the error rates obtained by our two variants and compares them with results found in the literature. The test protocols and related works are discussed below on each problem.

DBs	# images	# attributes	# classes
MNIST	70000	784 $(28 * 28 * 1)$	10
ORL	400	10304 $(92 * 112 * 1)$	40
COIL-100	7200	3072 $(32 * 32 * 3)$	100
OUTEX	864	49152 $(128 * 128 * 3)$	54

Table 2. Database summary

DBs	*Extra-trees*	*Sub-windows*	Related work
MNIST	3.26%	2.63%	12% to 0.7% [10]
ORL	4.56% ± 1.43	2.13% ± 1.18	7.5% to 0% [11]
COIL-100	2.04%	0.39%	12.5% to 0.1% [12]
OUTEX	64.35 %	2.78%	9.5% to 0.2% [13]

Table 3. Evaluation of accuracy: error rates on all problems.

3.1.1. *MNIST, Database of Handwritten Digits*

The MNIST database[3] [10] consists of 70000 handwritten digits that have been size-normalized and centered in images of 28×28 pixels with 256 grey levels per pixel. Different writing styles are characterized by thin or thick strokes, slanted characters, etc.

In the literature, the first 60000 images are generally used for learning and the remaining 10000 examples are used for validation. The results in Table 3 are obtained by strictly following this protocol. Error rates in [10] vary from 12% to 0.7%. In comparison, with 50 extra-trees, we get 3.26%. Using sub-windows, the error rate drops to 2.63% (with $T = 10$, $N_w = 360000$ and $w_1 = w_2 = 24$).

3.1.2. *ORL, Face Database*

The ORL database[4] from AT&T Laboratories Cambridge contains faces of 40 distinct persons with 10 images per person that differ

[3]http://yann.lecun.com/exdb/mnist/
[4]http://www.uk.research.att.com/facedatabase.html

in lighting, facial expressions (open/closed eyes, smiling/not smiling), facial details (glasses/no glasses) and contain minor variations in pose. The size of each image is 92×112 pixels, with 256 grey levels per pixel.

Published results in the literature range from 7.5% to 0% error rate ([14], [15], [16], [11]). Unfortunately, the protocol for testing is different from one paper to another (average of 3, 4, or 5 executions on different subsets of images) and furthermore the information provided does not always allow to reproduce it exactly. So, given the fact that this database is quite small and as there is no well-defined test protocol, our experiment uses resampling estimates to provide a fair assessment of our two methods. More precisely, we averaged the results of 100 runs of the two algorithms where, in each run, 5 distinct images were randomly drawn in each class to form the learning sample and the other half of the dataset was used to form a separate test sample. Following this procedure, we get with extra-trees ($T = 500$) a mean error rate of 4.56% and with sub-windows 2.13% (with $T = 10$ and $w_1 = w_2 = 32$). Of course, these error rates cannot directly be compared to other published results as the test protocols are different.

3.1.3. COIL-100, 3D Object Database

The Columbia University Object Image library[5] [17] gathers 3D objects databases. COIL-100 [18] is a dataset with colored images of 100 different 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 degrees. This corresponds to 72 images per object. In COIL-100, each image has been normalized to 128×128 pixels and are in true color. For our experiments, we have resized the original images down to 32×32 pixels.

As in many publications (e.g. [12]), we took for the learning sample 18 views for each of the 100 objects, starting with the pose at 0° and then going on with intervals of 20°. The remaining views were devoted to the test sample. Using this protocol, methods in the literature provide error rates from 12.5% to 0.1%. Extra-trees yields an error rate of 2.04% (with $T = 500$) that drops down to 0.39% with sub-windows (with $T = 10$ and $w1 = w2 = 16$).

3.1.4. OUTEX, Texture Database

Outex[6] [19] provides a framework for the empirical evaluation of texture analysis algorithms. The Contrib_TC_0006 dataset we took from Outex has been derived from the VisTeX dataset. It contains 54 colored textures and 16 images of 128×128 pixels in true color for each VisTex texture.

The OUTEX framework precisely defines the images to use in the learning and test sample (8 images for each texture in both ensembles). The paper [13] evaluates several feature extraction techniques and image transformation methods in combination with a traditional nearest neighbors algorithm. The resulting error rates on this dataset vary from 9.5% to 0.2%. Using the same protocol, the extra-trees method is especially bad with an error rate of 64.35% (even with $T = 1000$). On the other hand, sub-windows reduce error rates down to 2.78% (with $T = 10$ and $w1 = w2 = 4$). These results were foreseeable due to the global approach of extra-trees and localization aptitude of sub-windows, as we explained in section 2.2.

[5]http://www.cs.columbia.edu/CAVE/

[6]http://www.outex.oulu.fi /

3.1.5. Discussion

For each problem, our results are comparable to other approaches, but are slightly inferior to the best published results achieved by specialized algorithms tuned to the given problem. On the MNIST data, the sub-window algorithm was also outperformed by another generic method based on support vector machines [20]. On the ORL dataset, the comparison is only indicative. Given the difficulty of the OUTEX problem, it is not surprising that the best of specific methods performs better than our generic methods.

A great advantage of our algorithm, especially in its basic form, is its computational efficiency. Due to the random choice of the tests, the construction of extra-trees is very fast. For example, the construction of 1000 trees on OUTEX problem requires only about 5s[7]. On MNIST, 50 trees are grown in about 8 minutes. Also, the test of a new image is negligible (less than a millisecond whatever the problem). On the other hand, computing times with sub-windows are greater since the learning sample size increases. Furthermore, prediction times are also longer since they require the propagation of all sub-windows in the trees of the ensemble. For example, it takes about 8 minutes to build 10 extra-trees with sub-windows on OUTEX and about 0.6s to classify one image. Although our implementation is not optimal, we believe that these times are nevertheless reasonable.

3.2. Evaluation of Robustness

In contrast to most modern computer vision algorithms for recognition, our algorithms in their current form are not invariant to many important transformations of the image. Intensity variations are not expected to be a problem; any of the popular methods for intensity normalization should apply. Invariance to other transformations can be added as outlined at the end of this section. In the following paragraphs, we study the robustness of our algorithms under some typical conditions found in realistic image classification scenarios, using the COIL-100 dataset as an example.

3.2.1. Generalisation

Let us first study the generalisation ability of our approach by considering different learning sample sizes. On the COIL-100 dataset, reducing the number of training views increases perspective distortions between learned views and images presented during testing. Table 4 shows error rates under various choices of training images where nv denotes the number of views per object selected in the training sample. The remaining $72 - nv$ views are reserved for testing. The azimuthal angles corresponding to the selected training views are also given in the table and are thus repeatable and comparable. As expected, error rates increase when the size of the learning set decreases. Extra-trees with sub-windows are always better than extra-trees whatever the number of learning views, and their errors remain very small up to 8 images per object. These results are directly comparable – and in fact very similar – to those published by Obrzalek and Matas [12], which are also shown in the table.

3.2.2. Rotation

The second experiment aims at analysing the robustness of induced models to image-plane rotation of the test images. To this end, we

[7]Our code is written in C and the program runs on a Pentium IV 2.4GHz with 1GB memory.

Generalisation	*Extra-trees*	*Sub-windows*	LAFs [12]
$nv = 36; k * 10°$	0.33 %	0.06 %	-
$nv = 18; k * 20°$	2.04 %	0.39 %	0.1 %
$nv = 8; k * 45°$	7.55 %	1.53 %	0.6 %
$nv = 4; 45° + k * 90°$	12.46 %	4.94 %	5.3 %
$nv = 2; 0°, 90°$	24.91 %	12 %	12.2 %
$nv = 1; 0°$	36.1 %	24.83 %	24 %

Table 4. Error rates with different learning and test sample sizes for COIL-100 images.

have tested models grown from the original images (with 18 views per object in the learning sample) on rotated versions of the test images. Results for rotations between $0°$ and $45°$ are reported in Table 5. Extra-trees with sub-windows are shown to be more robust to rotations than extra-trees. Error rates are acceptable with this variant up to about $20°$ of rotation. Although they are not taken into account during the learning phase, the algorithm is thus robust to small rotational deviations.

Rotation	*Extra-trees*	*Sub-windows*
$ra = 0°$	2.04 %	0.39 %
$ra = 10°$	5.31 %	0.85 %
$ra = 20°$	22.80 %	3.20 %
$ra = 30°$	43.11 %	8.85 %
$ra = 45°$	76.61 %	31.69 %

Table 5. Error rates with 2D rotated test images for COIL-100 (18 views for learning): test images are rotated by ra degrees.

3.2.3. *Scaling*

Another important image transformation is scaling. As extra-trees require that all images contain the same number of pixels, they can not classify images of different sizes. Extra-trees on sub-windows however may classify images of any size, provided only that this size is larger than the sub-window size. We thus study invariance to scaling with this latter variant. To this end, we tested the model built from 32×32 images and sub-windows of 16×16 pixels (with $T = 10$) on scaled version of the test images ranging from 16×16 to 48×48 pixels. Results are summarized in Table 6. They remain quite good for moderate variations in scale. The poor results obtained with test images of the size of the sub-windows is explained by the fact that in this case only one sub-window can be extracted from test image and thus the classification corresponds to the average of only 10 classifications (one per extra-tree), which is not enough for the variance-reduction effect of extra-trees to take place.

Scaling	*Sub-windows*
16x16	71.72 %
24x24	2.67 %
32x32	0.39 %
40x40	1.43 %
48x48	7.15 %

Table 6. Error rates with scaled test images for COIL-100 (18 views for learning).

Occlusion	*Extra-trees*	*Sub-windows*
$of = 0\%$	2.04 %	0.37 %
$of = 25\%$	3.72 %	0.37 %
$of = 37.5\%$	12.10 %	4.72 %
$of = 40.625\%$	15.96 %	7.07 %
$of = 43.75\%$	21.85 %	14.04 %
$of = 46.875\%$	33.93 %	26.76 %
$of = 50\%$	49.26 %	47.69 %

Table 7. Error rates with occluded test images for COIL-100 (18 views for learning): of % of right part of each test image is set to black.

3.2.4. *Occlusion*

As a last experiment, we have studied the sensitivity of our models to occlusion by erasing increasing parts of the test images. Results are summarized in Table 7 where "$of = x\%$" means that $x\%$ of the pixels at the right part of the test images are replaced by black pixel values. Here again, extra-trees with sub-windows perform better than extra-trees and their accuracy remains good up to 40% of partial occlusions. These results are nevertheless inferior to those of Obrzalek and Matas [12] who obtained an error rate of 7.4% for 50% of occlusion which is comparable with our result for 40% of occlusion. An easy way to improve our method's robustness to larger occlusions could be to remove from the vote those sub-windows whose predictions seem to be "uncertain". For example, if a sub-window receives fewer than 5 identical predictions by the $T = 10$ trees, then this sub-window might not be taken into account in the final vote because its confidence is low. Experiments have shown that such a strategy would reduce the error rate from 47.69% down to 18.44% in the case of 50% occlusion. Additional investigations should be carried out to assess the validity of this approach.

3.2.5. *Discussion*

Since our algorithm considers raw pixel values at specific positions within the image or within a sub-window, many of the issues of template-matching techniques apply. In the case of the full-image classifier, any image transformations affect the single feature vector, which has a rather detrimental effect on classification accuracy. On the other hand, an image contains substantial redundancy – which is implied by the presence of informative structure – in the form of homogeneous regions and smooth intensity gradients. Thus, spatial image transformations do not alter all elements of the feature vector to the same extent. Moreover, some transformations result in image content being shifted to a different location while approximately preserving their local structure. The sub-window technique capitalizes on both of these effects: Many feature vectors will be left more or less intact by a given image transformation, resulting in remarkably robust performance. Their robustness to partial occlusions is competitive with the state of the art. This algorithm is also robust enough to small rotations and moderate scaling to admit, for example, brute-force multi-scale and/or multi-orientation processing at discrete scales and orientations during training or during testing.

Another way to achieve invariance to rotation and scaling is local normalization of sub-window sizes and orientations. For example, techniques similar to those proposed by Obrzalek and Matas

[12] could be applied. This is a subject of further study.

It should be noted that the parameters of the method used for this study are those that minimize the error in normal conditions. But these parameters have an influence on recognition rate with perturbed images. For example, we have observed that reducing the sub-window size increases the robustness to rotation. On the other hand, enlarging them gives better performance in presence of occlusions. This tradeoff between accuracy in normal condition and robustness to some transformations certainly deserves further investigation.

4. CONCLUSION AND FUTURE WORK

In this paper, we evaluate two generic algorithms for image classification based on ensembles of decision trees. First, extremely randomized trees [1] are applied directly on pixel values. This method yields good results and it is particularly attractive in terms of computational efficiency. Second, we propose a variant that involves extraction and classification of local sub-windows from images. This technique improves systematically the accuracy but increases computing times required to build a model and to predict the class of an image. On the four datasets we used, the accuracy of our algorithms is comparable to state-of-the-art techniques but slightly inferior to the best known results.

Specialized methods will probably always perform better than generic methods. Our results demonstrate however that the latter can come remarkably close. In many practical application contexts, a slight performance drop in exchange for reduced task-specific pre-processing and manual intervention may constitute a desirable trade-off.

The robustness of our approach was analysed in the presence of rotation, scaling and occlusions on the COIL-100 test set. We observed good robustness to small transformations introduced in test images. However, the method could certainly be improved according to this latter criterion by augmenting the learning sample with transformed versions of the original images. While general, this technique is especially interesting in the context of our algorithm because of its small computing times. Moreover, the sub-window variant can easily be augmented by techniques for normalizing rotation and scale to achieve invariance.

5. ACKNOWLEDGMENTS

Raphaël Marée is supported by the “Région Wallonne” (Belgium). Pierre Geurts is a research associate of the F.N.R.S., Belgium.

6. REFERENCES

[1] P. Geurts, *Contributions to decision tree induction: bias/variance tradeoff and time series classification*, Phd. thesis, Department of Electrical Engineering and Computer Science, University of Liège, May 2002.

[2] F. Roli and J. Kittler, Eds., *Proc. of the Third International Workshop on Multiple Classifier Systems*, Springer, 2002.

[3] C. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines*, MIT Press, Cambridge, MA, 2000.

[4] L. Breiman, J.H. Friedman, R.A. Olsen, and C.J. Stone, *Classification and Regression Trees*, Wadsworth International (California), 1984.

[5] L. Wehenkel, *Automatic learning techniques in power systems*, Kluwer Academic, Boston, 1998.

[6] P. Geurts, “Extremely randomized trees,” Tech. Rep., Department of Electrical Engineering and Computer Science, University of Liège, Belgium, June 2003.

[7] J. Dahmen, D. Keysers, and H. Ney, “Combined classification of handwritten digits using the ’virtual test sample method’,” in *Proc. Second International Workshop, MCS 2001 Cambridge, UK*, July 2001, pp. 99–108.

[8] M.S. Hoque and M. C. Fairhurst, “A moving window classifier for off-line character recognition,” in *Proc. of the Seventh International Workshop on Frontiers in Handwriting Recognition, Amsterdam*, September 2000, pp. 595–600.

[9] N. Winters and J. Santos-Victor, “Information sampling for appearance based 3d object recognition and pose estimation,” in *Proc. of the Irish Machine Vision Image Processing Conference, Maynooth, Ireland*, September 2001.

[10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[11] R. Paredes and A. Perez-Cortes, “Local representations and a direct voting scheme for face recognition,” in *Pattern Recognition in Information Systems, Proc. 1st International Workshop on Pattern Recognition in Information Systems*, July 2001, pp. 71–79.

[12] S. Obrzalek and J. Matas, “Object recognition using local affine frames on distinguished regions,” in *Electronic Proceedings of the 13th British Machine Vision Conference, University of Cardiff*, 2002.

[13] T. Mäenpää, M. Pietikäinen, and J. Viertola, “Separating color and pattern information for color texture discrimination,” in *Proc. 16th International Conference on Pattern Recognition*, 2002.

[14] G.-D. Guo, S. Li, and K. Chan, “Face recognition by support vector machines,” in *Proc. International Conference on Automatic Face and Gesture Recognition, 196-201.*, 2000.

[15] S. Lawrence, C. Lee Giles, A. C. Tsoi, and A. D. Back, “Face recognition: A convolutional neural network approach,” *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98–113, 1997.

[16] A. Nefian and M. Hayes, “Face recognition using an embedded HMM,” in *Proc. IEEE Conference on Audio and Video-based Biometric Person Authentication*, March 1999, pp. 19–24.

[17] H. Murase and S. K. Nayar, “Visual learning and recognition of 3d objects from appearance,” *International Journal of Computer Vision*, vol. 14, no. 1, pp. 5–24, 1995.

[18] S. Nene, S. Nayar, and H. Murase, “Columbia object image library: Coil-100,” Tech. Rep. CUCS-006-96, Department of Computer Science, Columbia University, 1996.

[19] T. Ojala, T. Mäenpää, M. Pietikäinen, J. Viertola, J. Kyllönen, and S. Huovinen, “Outex - new framework for empirical evaluation of texture analysis algorithms,” *Proc. 16th International Conference on Pattern Recognition, Quebec, Canada, 1:701-706*, 2002.

[20] C. J. C. Burges and B. Schölkopf, “Improving the accuracy and speed of support vector machines,” in *Advances in Neural Information Processing Systems*, M. C. Mozer, M. I. Jordan, and Thomas Petsche, Eds. 1997, vol. 9, p. 375, The MIT Press.