

Boosting bonsai trees for handwritten/printed text discrimination

Yann Ricquebourg, Christian Raymond, Baptiste Poirriez, Aurélie Lemaitre, Bertrand Coüasnon

▶ **To cite this version:**

Yann Ricquebourg, Christian Raymond, Baptiste Poirriez, Aurélie Lemaitre, Bertrand Coüasnon. Boosting bonsai trees for handwritten/printed text discrimination. Document Recognition and Retrieval (DRR), Feb 2014, San Francisco, United States. hal-00910718

HAL Id: hal-00910718
https://hal.science/hal-00910718

Submitted on 28 Nov 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Boosting bonsai trees for handwritten/printed text discrimination

Yann Ricquebourg[a], Christian Raymond[a],
Baptiste Poirriez[a], Aurélie Lemaitre[b] and Bertrand Coüasnon[a]

[a]Université Européenne de Bretagne, IRISA/INSA Rennes, France
[b]Université Européenne de Bretagne, IRISA/Université de Rennes-2, France

ABSTRACT

Boosting over decision-stumps proved its efficiency in Natural Language Processing essentially with symbolic features, and its good properties (fast, few and not critical parameters, not sensitive to over-fitting) could be of great interest in the numeric world of pixel images. In this article we investigated the use of boosting over small decision trees, in image classification processing, for the discrimination of handwritten/printed text. Then, we conducted experiments to compare it to usual SVM-based classification revealing convincing results with very close performance, but with faster predictions and behaving far less as a black-box. Those promising results tend to make use of this classifier in more complex recognition tasks like multiclass problems.

Keywords: Handwritten/Printed text discrimination, Boosting, Maurdor campaign

1. INTRODUCTION

During the last decades, the problem of classifying texts into printed or handwritten regions arose in the document image analysis. Moreover, the presence of both printed and handwritten texts in the same document is an important difficulty in the automation of the optical character recognition procedure. Indeed, both printed and handwritten texts are often present in many different kinds of documents: blank or completed forms, annotated prints, private or business correspondence, etc. In any of these cases, from a digitized version of the document it is crucial to detect and distinguish printed from handwritten texts so as to process them differently with the appropriate tools to extract the most possible informations during the recognition analysis.

The Maurdor campaign,[1] where this study places its application context, is exactly aiming at this goal. Maurdor aims at evaluating systems for automatic processing of written documents. To do so, the campaign intends to assess, in quantity and quality, the ability of such systems to retrieve relevant information from digital images obtained by digitizing paper documents. Prior to retrieving information, the problem then consists in characterizing semantics area of the documents (image, graphic, printed text, handwritten text, etc.). This campaign also favors efficient solutions, able to cope with a large variety of documents (kinds, languages, etc.).

In section 2, we present some evolutions in the solutions from the literature concerning discrimination between printed and handwritten texts. Section 3 motivates the use of boosting over small decision trees, called bonsai trees, unlike the classic use of simple decision-stumps that capture less information than a bonsai, and without falling in the over-fitting drawback of a full decision-tree. The different stages of our overall system are detailed in section 4. Section 5 describes the results of the experiments conducted to evaluate this system. Finally, section 6 summarizes the conclusions drawn from this study that shows using bonsai trees lead to an efficient classifier system, fast and not sensitive to parameters, and future work directions are proposed.

Further author information: (Send correspondence to Yann Ricquebourg) E-mails = FirstName.LastName@irisa.fr

2. STATE OF THE ART

Previous works on this subject of discrimination between printed and handwritten texts investigate various solutions.

In,[2] gradient and luminance histograms are extracted from the images, then transmitted to a neural network to segment the document into homogeneous semantic areas: printed characters, handwritten characters, photographs and paints. A neural network was also used in,[3] based on directional and symmetric features to distinguish printed from handwritten character images. In the same way, a method to discriminate handwritten from printed texts was proposed in,[4] using low level general features to classify using a feed-forward multilayer perceptron (MLP).

Other approaches proposed the use of statistical and structural features as nodes of tree classifiers for automatic separation of printed from handwritten text lines.[5,6] Then Hidden Markov Models (HMM) also took part to the competition as proposed in.[7] Outstanding results with Support Vector Machine (SVM) using Gabor filters and run-length histograms as features were obtained in,[8] and their system was also improved by a Markov Random Field (MRF) as post-processing. SVM were also used with Radon transform in,[9] whereas in[10] simple basic features like height, aspect ratio, density, maximum run-length are choosen to feed SVM.

In[11] was presented a method using simple generic features like the height of main body and different ratio of ascender and descender height to body computed from the upper-lower profile histograms. Those features fed a discriminant analysis approach for classification in printed or handwritten text.

Spectrum domain local fluctuation detection (SDLFD) was introduced[12] as a method to distinguish printed from handwritten texts. Local regions of the document are transformed into frequency domain feeding a MLP.

In,[13] simple basic features (like width, height, area, density and major axis) were also extracted and then processed with various data mining algorithms. And recently,[14] also used basic and generic shape features (area, perimeter, for factor, major-minor axes, roundness, compactness) to separate handwritten and printed texts using a k-nearest neighbor algorithm (KNN).

Out of those studies, we drew the conclusion that generic and basic features were often retained as interesting enough to achieve the goal, at word-level as in recent works[13,14] as concluded to be most promising by,[8] and that SVM classification turned out to be a solution now very often selected with strong results.

3. WHY BOOSTING BONSAI TREES?

To process all future OCR stages we would like an efficient classifier able to deal with multiclass/multilabel problems, able to manage arbitrary input features and able to deal with a big dataset. In this work we investigate the use of boosting decision trees. Those approaches turned out to be very efficient in Natural Language Processing, with discrete features. Their interesting aspects, detailed hereafter motivated the experiment to transfer this classifier to the image world, with continuous numeric features.

Boosting is a meta-learning algorithm, the final model is a linear combination of several classifiers built iteratively. The principle is to assign an identical weight at the beginning to all training samples and use a weak learner (classifier better than random guessing) to classify the data. Then all misclassified samples are "boosted", their weight are increased and an other weak classifier is run on the data with the new distribution. Misclassified samples by the previous classifier are boosted and the process iterate until we decide to stop it. Thus all individual classifier are linearly combined according to their respective performance to build the final model. More precisely we will use a multiclass/multilabel algorithm of boosting called Adaboost.MH.[15] This algorithm is derived using a natural reduction of the multiclass, multilabel data to binary data. Adaboost.MH maintain a set of weights over training examples and labels. As boosting progresses, training examples and their corresponding labels that are difficult to predict correctly get incrementally higher weights, while examples and labels that are easy to classify get lower weights.

Algorithm is presented in figure 1, given S a set of training samples $(x_i, Y_i), \ldots, (x_m, Y_m)$ where each instance $x_i \in X$, and each label $Y_i \in \gamma$ the set of all possible labels. D_t is the weight distribution, and $ht(x_i, l)$ is the real valued prediction of the weak learner of the presence of the label l into x_i.

Given: $(x_i, Y_i), \ldots, (x_m, Y_m)$ o $x_i \in X$, $Y_i \in \gamma$
Init $D_1(i, l) = \frac{1}{mk}$
For $t = 1, \ldots, T$:

- Provide distribution D_t to the weak learner
- Get weak hypothesis $h_t : X * \gamma \rightarrow \Re$
- Choose $\alpha_t \in \Re$
- Update:$D_{t+1}(i, l) = \frac{\exp(-\alpha_t \gamma[l] ht(x_i, l))}{Z_t}$
  where Z_t is a normalisation factor that make D_{t+1} a distribution

Output final hypothesis:

$$f(x, l) = \sum_{t=1}^{T} \alpha_t h_t(x, l)$$

Figure 1. AdaBoost.MH algorithm

In,[15] "decision stumps" are chosen as weak learners which are simply one level decision trees (two leaves). AdaBoost in combination with trees has been described as the "best off-the-shelf classifier in the world".[16] In bonzaiboost,[17] we propose to build deeper, but small, decision trees (i.e decision trees with low depth, we will refer to them as bonsai trees: they are deep enough to capture a piece of information while not being sensitive to over-fitting). This strategy turned out to be very effective in practice since bonzaiboost is ranked first on multiclass problems on `http://mlcomp.org` (a web site that evaluates classification algorithms on representative classification problems).

As said above, using a SVM classifier appears to be a good choice for hand/typed separation since we are facing a binary classification problem with numeric input features. This classifier has been widely used and is state-of-the-art for this kind of problems. Despite these performances in most of the classification problems, SVM exhibits several drawbacks that we may avoid with our proposed solution. We address next a point-to-point comparison between our proposed algorithm and SVM.

3.1 Training time

Concerning SVM, when features number is greatly lower than instances number a fast linear kernel would not be efficient, a RBF kernel is then a reasonable choice, but makes the training phase costly since the complexity is quadratic.

Concerning boosting, the complexity of Adaboost.MH is linear according to the number of training samples and number of labels. We have to add the complexity of the weak learner that is linear, for instance with a binary decision tree, of the number of nodes and the number of features. We can note that the boosting algorithm itself is not parallelizable but the weak learner is in two ways: induce tree nodes in parallel, evaluate features in parallel. Processing numeric features in a decision tree is not efficient, because the decision tree look at each tree node for the best threshold to apply in order to split the current node, for that it examine every threshold in order to find the best that can become costly if many different numeric values exists in the training data.

Finally, here follows the main aspects of each of the two classifiers:

	SVM	Boosting
complexity	quadratic	linear
numeric features process	distance	threshold
parallelisation	yes	yes
overall	equality	

3.2 Tuning phase

SVM need to be finely tuned: in presence of a RBF kernel, at least two parameters need to be tuned to make the SVM efficient, the Gaussian parameter γ and C, the trade-off between training error and margin. To tune

these parameters, a cross-validation step on a variety of these pairs of parameters is necessary. The complete SVM training process may become very costly.

On the other hand, no parameter need to be finely tuned in our boosting, there is only two "human-readable" parameters:

1. the tree depth: going further than one level tree ("decision stumps") allows to capture structure in input data (XOR problem can not be solved with one level tree). Trees should be deep enough to capture important structure information while kept as simple as possible to avoid the instability effect of decision trees and keeping a good generalisation performance. Practically, going to depth 2, 3 or 4 is sufficient for many tasks and the precise depth choice is not crucial since the leaning curves for different choice of depth will meet each other at a given number of iterations.
2. the number of iterations: the number of iterations is not crucial, it need just to be big enough to get the tangent of errors that actually remains stable because boosting is not prone to overfitting in the general case[18] even if this phenomenon appears in some circumstances (i.e. presence of noise).[19]

We can conclude that this tuning phase is clearly a huge drawback of SVM when confronted to a big amount of data. Finding the good parameters that make the SVM efficient is a very costly process.

3.3 Data preparation

SVM needs data preparation:

1. although particular variants turn out to be capable of feature selection when combined with a particular norm,[20] with most of the implementations, like libSVM,[21] a pre-processing step of features selection is welcome to filter useless features that might otherwise impact negatively the SVM performance
2. SVM can deal only with numeric features, all no-numeric features have to be transformed
3. to avoid the prevalence of some feature to others, features should be scaled

Whereas for boosting:

1. no need of feature selection since the tree base classifier is doing it intrinsically
2. decision trees can deal with several feature type: numeric or discrete
3. no numeric scaling is necessary

Then, boosting over decision trees will obviously simplify the data preparation process.

3.4 Multiclass/multilabel classification

SVM needs heuristics to perform multiclass/multilabel classification: the usual way to do multiclass or/and multilabel classification with a SVM is to decompose the problem into several binary problems using the *one-vs-all* or the *one-vs-one* paradigm and managing a vote scheme. The whole process combined with the necessary step of tuning make the whole process extremely costly.

The boosting algorithm Adaboost.MH is intrinsically multilabel/multiclass and the cost to pass from a binary problem to a multiclass/multilabel one is light.

Thus, boosting should be clearly of better interest thanks to its potential for future more complex classification situations.

3.5 Post-processing interpretation possibilities

SVM classifier works as a black box: you have no feedback of the model on your data, like what features are relevant.

In boosting, each decision tree give an interesting feedback on the data thanks to the readable rules it produced, thus many interesting statistics may be computed from the complete boosting model: what features have been selected? how many times ? what features conjonction have been used ? etc.

4. OUR SYSTEM

This system has been designed and tested in the context of the Maurdor campaign.[1] Here we focused on the problem of distinguishing handwritten from printed text blocks. It is important to notice that in the process implied by the Maurdor campaign, this task is invoked with a block of homogeneous text (only handwritten or only printed).

4.1 Text-block segmentation into lines and words

The method used for text-block segmentation has been presented in.[22] This method is dedicated to the segmentation of homogeneous text blocks into lines and words. It is based on the notion of perceptive vision that is used by the human vision. The goal is to combine several points of view of the same image in order to detect the text lines.

We first consider an image at low resolution (the dimensions of the original image are divided by 16). At that resolution level, the text lines appear as line segments. Thus, we apply a line segment extractor, based on Kalman filtering. The line segment extraction gives a prediction on the presence of text lines. Then, we use the extraction of connected components at a high resolution level in order to verify the presence of a text line.

Once the text lines have been detected, our goal is to segment each text line into word. Thus, we use a neighbour distance based on Voronoi tesselation. We compute the distance between each neighbouring connected components. Then, using the k-mean algorithm, we compute a distance threshold between intra-word distances and inter-word distances. Thanks to this threshold, we group together the connected components that belong to the same word.

The figure 2 presents some examples of the segmentation of text blocks into words. More details on the method can be found in.[22]

4.2 Features extraction

This next step makes use of the segmentation information provided above to split document image into the corresponding words sub-images. Then, the feature extraction process transforms each of these sub-images into a real-value vector.

For this system, we preferred to use a standard and general features set (that we use as a library in different application domains, such as word or number recognition for handwriting[23]) instead of designing dedicated and specific features. So, each sub-image leads to the computation of a 244-dimensional real-value vector composed of the following set of features:

- Basic features like width, height, surface, pixel-value average, centre of inertia coordinates, moments of inertia (giving 11 components).
- 13^{th} order Zernike moments[24] (giving 105 components).

$$Z_{pq} = \frac{p+1}{\pi} \int_0^{2\pi} \int_0^{+\infty} \overline{V_{pq}(r,\theta)} f(r,\theta)\, r \,\mathrm{d}r \,\mathrm{d}\theta \tag{1}$$

where p is the radial magnitude and q is the radial direction, and $\overline{V}$ denotes the complex conjugate of a Zernike polynomial V, defined by $V_{pq}(r,\theta) = R_{pq}(r)e^{iq\theta}$ where $p-q$ is even with $0 \leqslant q \leqslant p$ and R is a real-valued polynomial:

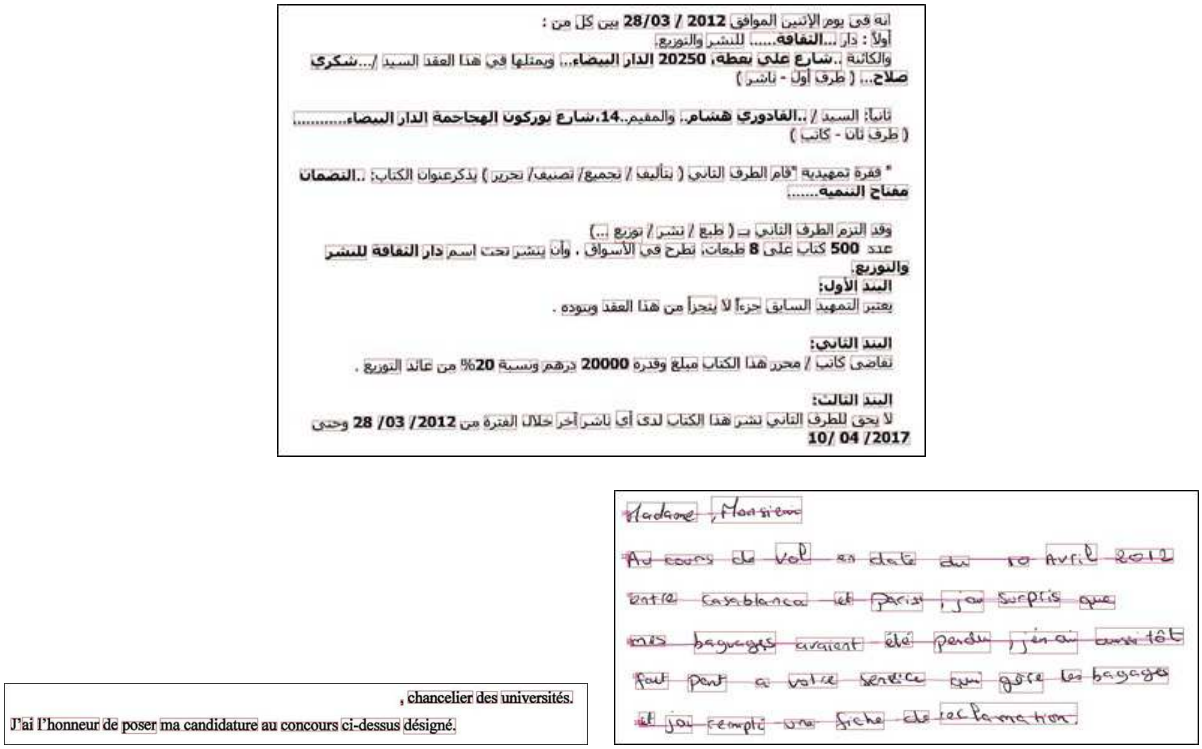


Figure 2. Segmentation of text blocks into words, using the proposed perceptive line of words approach

$$V_{pq}(r,\theta) = R_{pq}(r)e^{iq\theta} \quad \text{where } p-q \text{ is } even \text{ and } 0 \leqslant q \leqslant p$$

$$R_{pq}(r) = \sum_{m=0}^{\frac{p-q}{2}} (-1)^m \frac{(p-m)!}{m!(\frac{p-2m+q}{2})!(\frac{p-2m-q}{2})!} r^{p-2m}$$

- Histograms of 8-contour directions using Freeman chain code representation (with a zoning in 16 areas (2x8), implying 16 histograms, giving 128 components);

4.3 Learning with bonzaiboost

Each features vector is supplied to the bonzaiboost learning. We should not forget that processing numeric features is a penalizing case for the decision-tree based boosting, that has to deal with a huge amount of thresholds (as pointed out in section 3.1) associated to all the float-values. Currently bonzaiboost uses a greedy algorithm to select a candidate among all numeric thresholds.

To address this problem, the numeric precision was tested to be rounded naively to 10^{-2} to reduce the number of candidate threshold. It can drastically cut their number and speed up the training up to a factor 2 generally. It's important to notice that this reduction is performed at no significant cost concerning overall recognition results in experiments. Nevertheless, this is actually not optimal because the best reduction should be different for each feature. We are currently implementing a smart automatic method to extract thresholds candidate that would lead to an important training time reduction without loosing accuracy.

4.4 Post-processing interpretation

As presented before, after the training stage, the model built by bonzaiboost may give some interesting and interpretable informations, that could lead to an optimization of the features set.

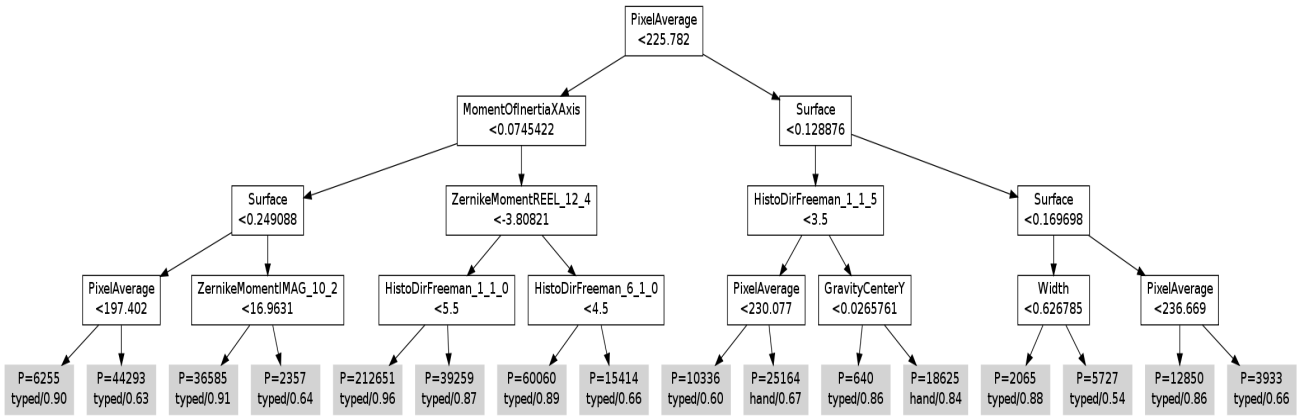


Figure 3. The weak learner computed at the iteration 1 of the boosting. Each node is presented with the binary test selected. Each leaf shows the number P of samples that belong to it, and the majority label with its probability

For instance, the features are automatically selected, and you can output which ones are the most used by the system (see table 1). Moreover, each bonsai tree represents human-readable rules that can exhibit how the system makes its decision (see figure 3).

5. EVALUATION

We evaluated our system using the images from the documents of the Maurdor campaign.[1] This corpus proposes documents in English, French and Arabic, of different kinds: blank or completed forms, printed but also manually annotated business documents, private and handwritten correspondence sometimes with printed letterheads, printed but also manually annotated business correspondence, other documents such as newspaper articles or blueprints, etc.

We conducted two experiments in parallel to compare our system based on boosting using bonzaiboost,[17] and a reference one where the classification stage has been rewritten, based on SVM using the open-source library libSVM.[21]

5.1 Training

To train our two systems, we used the *train0* and *train1* datasets of Maurdor campaign, resulting in nearly 500,000 ground-truth word images corresponding to 3000 pages (see table 2). The SVM classifier converged in about 10 hours to train (with a modified version of libSVM to use parallelism, 16 threads and 20 GB of parameter cache, otherwise lasting 133 hours), whereas the bonzaiboost classifier was stopped after 4000 iterations of training that lasted about 56 hours.

From one hand, it is worth mentioning that boosting can be continued as far as the user wants. The limit of 4000 iterations is set by default only arbitrary, since we observed out of experiments that this value was far beyond the number of needed iterations to reach good performance. bonzaiboost used indeed 51 seconds by iteration and could be stopped (and resumed if needed) far earlier as visible on the convergence curves of figure 4, with a linear saving of time. Moreover, as visible on this figure, continuing the training does not lead to over-fitting since the validation set still shows improving results while the train set has already reached its maximum.

Besides, we can use thresholds reduction by rounding numeric values described earlier. Here this cut their number from more than 49 millions, to about 1 million different thresholds, thus implying a faster round iteration, reduced from 51 seconds to 29 seconds, that would also save time.

On the other hand, as described in section 3.2, SVM needs a fine tuning of all the parameters to reach good performance. Without this tuning, and using the default parameters, only poor results of table 3 were obtained. We then performed a pre-processing before the full training, to tune the two main parameters of the RBF kernel in SVM using a cross-validation script supplied with libSVM, but increasing the cost by a factor 450 by default.

Table 1. Frequency of each attribute selected at nodes of the 4000 trees, during the 4000 boosting iterations of training

Frequency	Attribute
1715	PixelAverage
1177	Surface
971	ZernikeMomentREAL_0_0
672	ZernikeMomentREAL_2_0
590	ZernikeMomentREAL_6_0
583	ZernikeMomentREAL_6_2
568	ZernikeMomentIMAG_4_2
562	ZernikeMomentIMAG_3_1
556	ZernikeMomentREAL_4_0
551	ZernikeMomentREAL_3_1
549	GravityCenterY
542	ZernikeMomentREAL_8_2
532	WidthHeightRatio
525	ZernikeMomentIMAG_6_2
511	ZernikeMomentIMAG_5_1
506	Orientation
501	ZernikeMomentREAL_12_4
499	ZernikeMomentREAL_8_4
494	ZernikeMomentREAL_8_0
494	ZernikeMomentREAL_10_2
493	ZernikeMomentIMAG_7_1
492	ZernikeMomentREAL_12_2
485	ZernikeMomentREAL_10_0
473	ZernikeMomentIMAG_1_1
469	ZernikeMomentIMAG_10_2
467	ZernikeMomentIMAG_7_3
465	ZernikeMomentREAL_12_8
465	ZernikeMomentREAL_12_0
464	ZernikeMomentIMAG_13_1
462	ZernikeMomentREAL_9_1
460	ZernikeMomentREAL_5_1
457	ZernikeMomentIMAG_5_3
455	ZernikeMomentREAL_12_6
452	GravityCenterX
451	ZernikeMomentREAL_7_1
450	ZernikeMomentIMAG_8_2
448	ZernikeMomentREAL_10_4
448	ZernikeMomentIMAG_11_1
446	ZernikeMomentIMAG_9_1
445	ZernikeMomentIMAG_2_2
444	ZernikeMomentIMAG_13_5
437	ZernikeMomentIMAG_12_2
431	ZernikeMomentIMAG_10_4
430	ZernikeMomentIMAG_9_3
429	ZernikeMomentIMAG_12_4
426	ZernikeMomentREAL_9_3
426	ZernikeMomentREAL_5_3
426	ZernikeMomentREAL_1_1
423	ZernikeMomentREAL_13_3
420	ZernikeMomentREAL_11_1
419	ZernikeMomentREAL_4_2
419	ZernikeMomentREAL_13_5
418	ZernikeMomentREAL_11_5
415	ZernikeMomentREAL_6_4
⋮	⋮
58	FreemanHistoDir_1_0_5
57	FreemanHistoDir_6_1_7
49	FreemanHistoDir_2_1_7

Table 2. Datasets of the Maurdor campaign

Name	Documents	Word images
train0	1000	157,761
train1	2000	338,453
dev1	1000	165,309
test1	1000	181,239

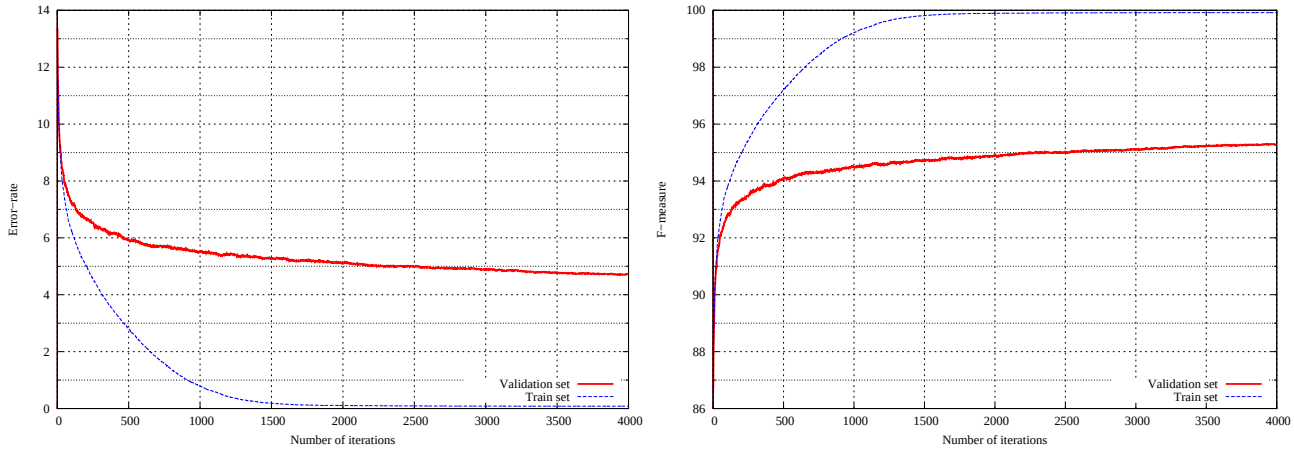


Figure 4. Error-rate and F-measure curves for bonzaiboost. It illustrates the unsensivity to over-fitting: whilst the train reached a maximum result after 2000 iterations, further training iterations still improve the validation results

Table 3. Classification results on the validation set

SVM (without tuning)								
Label	Predicted	Truth	Correct	Error	Precision %	Recall %	F-measure %	Error-rate %
hand	21,141	30,415	16,822	13,593	79.57	55.31	65.26	44.69
printed	144,168	134,894	130,575	13,593	90.57	96.80	93.58	10.08
All	165,309	165,309	147,397	17,912	89.16	89.16	89.16	10.84

SVM (with tuning)								
Label	Predicted	Truth	Correct	Error	Precision (%)	Recall (%)	F-measure (%)	Error-rate (%)
hand	28,726	30,415	26,185	4,230	91.15	86.09	88.55	13.91
printed	136,583	134,894	132,353	4,230	96.90	98.12	97.51	3.14
All	165,309	165,309	158,538	6,771	95.90	95.90	95.90	4.10

bonzaiboost								
Label	Predicted	Truth	Correct	Error	Precision (%)	Recall (%)	F-measure (%)	Error-rate (%)
hand	29,128	30,415	25,868	4,547	88.81	85.05	86.89	14.95
printed	136,181	134,894	131,634	4,547	96.66	97.58	97.12	3.37
All	165,309	165,309	157,502	7,807	95.28	95.28	95.28	4.72

Tuning using all the data is actually too costly and the tuning phase has been conducted with only 10% of the full training dataset. Despite this reduction, this tuning stage lasted more than 12 days with the parallel version of libSVM using 16 threads.

5.2 Prediction

The validation set used in our experiments was *dev1* dataset, resulting in 181,239 test images. The SVM based system reached 4.10% of error-rate, that can be compared to 4.72% of error-rate for bonzaiboost (see table 3). If their respective performance results are quite close, it is also important to add that this decoding time is very fast for bonzaiboost (which needs only 1 min 30 sec to predict all the dataset, taken as a whole) whereas the SVM system needed 64 minutes for prediction (also with the input of all the dataset as a whole). We can note that for future multiclass or/and multilabels problems the bonzaiboost prediction time will remain stable when the SVM prediction time will increase as much as the number of class, since we will need to query as many SVM as classes.

Table 4. Results for the test set corresponding to round 1 of Maurdor campaign

System	Precision (%)	Silence (%)	Error-rate (%)
SVM	95.41	3.55	7.98
bonzaiboost	94.93	4.03	8.90
bonzaiboost-2	94.42	0.37	5.93

5.3 Final decision

In the context of the application for the Maurdor campaign, our system was tested on *test1* dataset, where we had to output a prediction at zone-level (in other words, for a group of images, each one supposedly corresponding to a single word of an homogeneous text). To this end, we added a majority vote stage taking the individual outputs of the above predictions, and voting for the group. Naturally, this lead to a significant improvement of the recognition, because the effect a mis-recognized entity can be masked by the majority vote inside the associated zone (see table 4 where both SVM and bonzaiboost systems reach around 95% of precision among the non silent outputs, thus implying an error-rate smaller than 10% considering all the inputs).

Concerning this final decision stage, it is also worth mentioning that:

- Those results are significantly better than the official result of the first competition round of the Maurdor campaign (that will be published) where our system finished first with a precision of 90.4% (but with a silent rate of 6.6%) compared to the second competitor system with a precision of 89.9% (with no silence at all). This is because we had the time to correct some bugs and improve the learning stage with more of the available data.
- We investigated improved voting strategies, in particular using the surface of the word and the confidence score of our bonzaiboost classifier. But since no score value nor distance is supplied as outputs of the available libSVM library, we were not able to use those improvements with the SVM for comparison. Nevertheless, they are given for information in table 4 as a third system, bonzaiboost-2, to show that the silence rate, where no majority could be found, can be overcome (and nearly reduced to zero) without significant lost of performance, in spite of very problematic images sometimes (see figure 5).

6. CONCLUSION

In this article we presented the idea to use the Adaboost.MH in a efficient way on small decision trees, and not usual simple decision-stumps, on our image classification task. In this application context, this classifier turned out to be very promising: its performance reached SVM which is generally used. Moreover it proposes very attractive advantages: automatic selection of features implying no prior choice, no scaling between features, only two general parameters (number of iteration and depth of the decision trees) not critical and which don't need to be finely tuned far from the costly optimization of critical SVM parameters. And last but not least, the prediction time is very fast (40 time faster than SVM in our experiments).

Concerning our application system of printed/handwritten text discrimination, short-term work will focus on improving the vote for a text region: instead of a majority vote, eventually with weights, we wish to add a training stage with bonzaiboost to learn the best way to vote according to different situations. Concerning longer-term work on document processing, the good properties of this classifier should enable to easily build multiclasses systems, able to predict multi labels properties, to recognize printed text, handwritten text and also graphic, logo, signature... As presented in introduction, such classifiers able to identify the semantic nature of document regions would improve the performance of OCR systems during the segmentation and structure analysis process.



Figure 5. Difficult samples from the Maurdor campaign datasets (mixed, twisted, hand-like printed, background...)

ACKNOWLEDGMENTS

This work has been conducted in collaboration with Cassidian (a division of EADS, producing security and defense systems) to study, develop and implement a prototype for automatic recognition of documents intended to be integrated into a management chain of military intelligence, for the French Ministry of Defence (DGA).

REFERENCES

[1] DGA, Cassidian, and LNE, "Maurdor campaign dataset," (2013). `http://www.maurdor-campaign.org`.

[2] Imade, S., Tatsuta, S., and Wada, T., "Segmentation and classification for mixed text/image documents using neural network," in [*Document Analysis and Recognition, 1993., Proceedings of the Second International Conference on*], 930–934 (1993).

[3] Kuhnke, K., Simoncini, L., and Kovacs-V, Z. M., "A system for machine-written and hand-written character distinction," in [*Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 2) - Volume 2*], *ICDAR '95*, 811–, IEEE Computer Society, Washington, DC, USA (1995).

[4] Violante, S., Smith, R., and Reiss, M., "A computationally efficient technique for discriminating between hand-written and printed text," in [*Document Image Processing and Multimedia Environments, IEEE Colloquium on*], 17/1–17/7 (1995).

[5] Pal, U. and Chaudhuri, B., "Machine-printed and hand-written text lines identification," *Pattern Recognition Letters* **22**(3-4), 431 – 441 (2001).

[6] Mazzei, A., Kaplan, F., and Dillenbourg, P., "Extraction and classification of handwritten annotations," in [*UbiComp '10*], (2010).

[7] Guo, J. and Ma, M., "Separating handwritten material from machine printed text using hidden markov models," in [*Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*], 439–443 (2001).

[8] Zheng, Y., Li, H., and Doermann, D., "Machine printed text and handwriting identification in noisy document images," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **26**(3), 337–353 (2004).

[9] Zemouri, E. and Chibani, Y., "Machine printed handwritten text discrimination using radon transform and svm classifier," in [*Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on*], 1306–1310 (2011).

[10] Shirdhonkar, M. and Kokare, M. B., "Discrimination between printed and handwritten text in documents," in [*IJCA Special Issue on ?Recent Trends in Image Processing and Pattern Recognition?. RTIPPR '10*], (2010).

[11] Kavallieratou, E. and Stamatatos, S., "Discrimination of machine-printed from handwritten text using simple structural characteristics," in [*Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*], **1**, 437–440 Vol.1 (2004).

[12] Koyama, J., Kato, M., and Hirose, A., "Distinction between handwritten and machine-printed characters with no need to locate character or text line position," in [*Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*], 4044–4051 (2008).

[13] da Silva, L., Conci, A., and Sanchez, A., "Word-level segmentation in printed and handwritten documents," in [*Systems, Signals and Image Processing (IWSSIP), 2011 18th International Conference on*], 1–4 (2011).

[14] Patil, U. P. and Begum, M., "Word level handwritten and printed text separation based on shape features," *International Journal of Emerging Technology and Advanced Engineering* **2**(4), 590–594 (2012).

[15] Schapire, R. E. and Singer, Y., "BoosTexter: A boosting-based system for text categorization," *Machine Learning* **39**, 135–168 (2000). `http://www.cs.princeton.edu/~schapire/boostexter.html`.

[16] Breiman, L., "Arcing classifiers," *ANNALS OF STATISTICS* **26**, 801–823 (1998).

[17] Raymond, C., "Bonzaiboost," (2013). `http://bonzaiboost.gforge.inria.fr`.

[18] Freund, Y. and Schapire, R. E., "A short introduction to boosting," in [*In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*], 1401–1406, Morgan Kaufmann (1999).

[19] Vezhnevets, A. and Barinova, O., "Avoiding boosting overfitting by removing confusing samples," in [*Proceedings of the 18th European conference on Machine Learning*], *ECML '07*, 430–441, Springer-Verlag, Berlin, Heidelberg (2007).

[20] Bradley, P. and Mangasarian, O. L., "Feature selection via concave minimization and support vector machines," in [*Machine Learning Proceedings of the Fifteenth International Conference(ICML 98*], 82–90, Morgan Kaufmann (1998).

[21] Chang, C.-C. and Lin, C.-J., "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology* **2**, 27:1–27:27 (2011). Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[22] Lemaitre, A., Camillerapp, J., and Coüasnon, B., "A perceptive method for handwritten text segmentation," in [*Document Recognition and Retrieval XVIII*], (2011).

[23] Ricquebourg, Y., Coüasnon, B., and Guichard, L., "Evaluation of lexicon size variations on a verification and rejection system based on svm, for accurate and robust recognition of handwritten words," *Proc. SPIE, Document Recognition and Retrieval XX* **8658**, 86580A–86580A–11 (2013).

[24] Teague, M. R., "Image analysis via the general theory of moments," *Journal of the Optical Society of America (1917-1983)* **70**, 920–930 (August 1980).