FAST MULTI-CLASS IMAGE ANNOTATION WITH RANDOM SUBWINDOWS AND MULTIPLE OUTPUT RANDOMIZED TREES

Marie Dumont[1], Raphaël Marée[2], Louis Wehenkel[1,2], and Pierre Geurts[1,2]
[1]*Systems and Modeling - Department of Electrical Engineering and Computer Science*
[2]*Bioinformatics and Modeling - GIGA-R*
University of Liège, Sart-Tilman B28, B-4000 Liège, Belgium
{*m.dumont,raphael.maree,l.wehenkel,p.geurts*}*@ulg.ac.be*

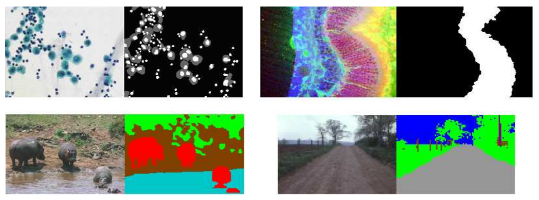
Keywords: image annotation, machine learning, decision trees, extremely randomized trees, structured outputs

Abstract: This paper addresses image annotation, i.e. labelling pixels of an image with a class among a finite set of predefined classes. We propose a new method which extracts a sample of subwindows from a set of annotated images in order to train a subwindow annotation model by using the extremely randomized trees ensemble method appropriately extended to handle high-dimensional output spaces. The annotation of a pixel of an unseen image is done by aggregating the annotations of its subwindows containing this pixel. The proposed method is compared to a more basic approach predicting the class of a pixel from a single window centered on that pixel and to other state-of-the-art image annotation methods. In terms of accuracy, the proposed method significantly outperforms the basic method and shows good performances with respect to the state-of-the-art, while being more generic, conceptually simpler, and of higher computational efficiency than these latter.

1 INTRODUCTION

In this paper, we propose a new supervised learning approach for fast multi-class image annotation. Given a training set of images with pixel-wise labelling (i.e. every pixel is labelled with one class among a finite set of predefined classes), the goal is to build a model to predict accurately the class of every pixel of any new, unseen image. We distinguish this problem from image classification (where the goal is to assign classes to entire images), image segmentation (where the goal is to partition images into coherent regions without semantic labelling), object detection (where the goal is to determine if an object of a certain type is present in an image), and image retrieval (where the goal is to retrieve images similar to a set of given images). In some sense, image annotation could be considered as the joint detection, recognition, and segmentation of (multiple) objects in one image. Image annotation is also named image labelling in (He et al., 2004), simultaneous object recognition and segmentation in (Shotton et al., 2006), background/foreground separation or figure/ground segregation (when only two classes are defined) in (Yu and Shi, 2003). Examples of images and their annotation are given in Figure 1. Since manual annotation is labor intensive and some applications require the annotation of thousands of images, there is a strong need for automatic annotation tools. Moreover, possible applications of such techniques are numerous, eg. in aerial photography analysis or in biomedical problems where high-throughput microscopy systems can generate thousands of images daily that represent very different phenotypes under various experimental conditions. In this context, generic methods that can automatically learn to adapt to problem specificities are strongly needed.

Figure 1: Examples of images (left) with their manual annotation (right). Each color on the right images represents a different class.



Several methods have been proposed in the literature to automatically annotate images. Some of them are ad-hoc methods developed for specific image families while others seek to be more generic. In (Cour and Shi, 2007) and (Borenstein and Malik, 2006), a bottom-up approach is used to segment horse images into coherent regions and a top-down method groups the segments into larger regions. The method of (Meurie et al., 2003) was applied to cell images and consists in simplifying the images by filtering, using a SVM pixel classifier, realizing a marker extraction and finally, applying color watershed segmentation. In (Bertelli et al., 2007), a method is proposed to separate foreground and background of retina images, based on the minimization of a cost function expressing the dissimilarity between the image to annotate and a reference image. Several authors (Shotton et al., 2006; Shotton et al., 2007; He et al., 2004; Verbeek and Triggs, 2007) have proposed to use conditional random field models for image annotation. These approaches differ in the features that are exploited in the model (which are based on texture, shape, color, layout, location...), the incorporation of local and/or global information, and in the way the model parameters are learned.

Our starting point in this paper is the image classification method of Marée et al. (Marée et al., 2003; Marée et al., 2005) based on the extraction of random subwindows, their description by raw pixel values, and their classification by extremely randomized trees (Geurts et al., 2006a). Randomization of patch extraction was first proposed in (Marée et al., 2003) for image classification and subsequently in (Marée et al., 2005; Nowak et al., 2006). To the best of our knowledge it was never considered for multi-class pixelwise labelling. Different variants of randomized trees were used in the recent years by an increasing number of computer vision researchers, respectively for content-based image retrieval (Marée et al., 2007; Philbin et al., 2007), keypoint tracking (Lepetit and Fua, 2006) (inspired by (Amit and Geman, 1997) for handwritten digit recognition), object categorization (Bosch et al., 2007; Moosmann et al., 2006), visual identification (Nowak and Jurie, 2007), bilayer video segmentation (Yin et al., 2007), robot navigation (Jodogne et al., 2006; Ernst et al., 2006), image reconstruction (Geurts et al., 2006b). Very recently (Shotton et al., 2008), ensembles of randomized trees were used to build bags of semantic textons for image annotation. Another decision tree model, the Probabilistic Boosting-Tree, was used in the context of the detection of object boundaries (Dollar et al., 2006) to classify as edge or non-edge the central pixel of image patches described by roughly 50000 features.

The paper is structured as follows. We first describe a basic pixel annotation method and then our proposed method in Section 2. The first one is an adaptation of Marée et al's image classification method (Marée et al., 2005) to classify each pixel from a subwindow centered on it. The proposed approach exploits a multiple output extension of decision trees to simultaneously annotate all pixel labels of a subwindow. In Section 3, these methods are confronted on a diverse set of problems taken from the literature to assess their generality and compare their performances. We also analyse these two methods in terms of computing times and parameter sensitivity. Finally, we conclude and provide future work directions in Section 4.

2 METHODS

In Section 2.1, we briefly remind the basic principles behind the image classification method of (Marée et al., 2005). Section 2.2 describes our two extensions for dealing with the image annotation problem.

2.1 Image Classification

The goal of image classification is to build, from a learning sample of images, each labelled with a class among a set of predefined classes, a classifier able to predict as well as possible the class of unseen images. To this end, the method described in (Marée et al., 2003; Marée et al., 2005) first randomly extracts a large set of image subwindows and describes them by feature vectors composed of raw pixel values. Each subwindow is labelled with the class of its parent image. Then, a supervised learning method is used to build a subwindow classification model from these subwindows. At the prediction stage, subwindows are extracted from the unseen image and classified by the subwindow classifier. These predictions are then aggregated to get a single class prediction for the whole image.

In principle, any supervised classification method can be used to grow the subwindow classifiers. Marée et al. exploited a particular decision tree ensemble method, called extremely randomized trees or Extra-Trees (Geurts et al., 2006a). This method grows an ensemble of M (typically $M \in [10;100]$) unpruned trees, each one being created in a top-down fashion. With respect to other tree-based ensemble methods such as Tree Bagging or Random Forests, Extra-Trees select cut-points at random and use the whole learning sample rather than a bootstrap replica. Their node splitting algorithm depends on two parameters, namely the size K of the random subset of attributes considered at each split, and the minimal (sub)sample size to split a node, $n_{\min}$. In our experiments in Section 3, we will use the default value recommended in (Geurts et al., 2006a), i.e. K equal to the square root of the total number of attributes and $n_{\min} = 2$ (corresponding to fully developed trees). In Section 3.5, we will discuss the influence of K on accuracy and computing times.

2.2 Image Annotation

We address the problem of image annotation as follows: given a sample LS of images with every pixel labelled with one class among a finite set of predefined classes, build a classifier able to predict as accurately as possible the class of every pixel of an unseen image.

2.2.1 Subwindow Classification Model (or SCM)

A first obvious, but novel, extension of the image classification approach (Marée et al., 2005) for dealing with pixelwise labeling is to associate to each subwindow the class of its central pixel and then to label the pixels of an image with this classifier. The different steps of this approach are detailed below.

Learning sample generation. The learning sample of subwindows that will be used to create the subwindow central pixel classifier is created by extracting subwindows of fixed size $w \times h$ at random positions in training images. Each subwindow is then labeled by the class of the central pixel in its parent image. Each subwindow is described by the HSB values of its raw pixels, yielding a vector of $w \times h \times 3$ numerical attributes.

Learning. From the learning sample created at the previous step, a model is built using ensembles of extremely randomized trees (following (Marée et al., 2005)). Each test associated to an internal node of a tree simply compares a pixel HSB component to a numerical threshold ($T = [a < a_c]$). The search of the best internal test among K random tests requires to define a score measure. Different scores measures have been proposed in the decision tree literature. In this paper, we have chosen the Gini splitting rule of CART (Breiman et al., 1984) (also used in Random Forests) [1]. Formally, this score measure is defined as follows:

$$score(S,T) = G_C(S) - G_{C|T}(S), \qquad (1)$$

[1]We did some comparison with a log entropy based score that does not show any significant difference. We choose Gini entropy because it is faster to compute.

with

$$G_C(S) = \sum_{i=1}^{m} \frac{n_{i.}}{n_{..}}.(1 - \frac{n_{i.}}{n_{..}}), \qquad (2)$$

$$G_{C|T} = \sum_{j=1}^{2}\sum_{i=1}^{m} \frac{n_{ij}}{n_{..}}.(1 - \frac{n_{ij}}{n_{.j}}), \qquad (3)$$

where S is the subsample of size $n_{..}$ associated to the node to split, m is the number of pixel classes, $n_{i.} (i = 1,\ldots,m)$ is the number of subwindows in S whose central pixel is of class i, n_{i1} (resp. n_{i2}) is the total number of subwindows of S whose central pixel is of class i and which satisfy (resp. do not satisfy) the test T, while $n_{.1}$ (resp. $n_{.2}$) is the number of subwindows of S which satisfy (resp. do not satisfy) T.

The vector of probabilities which is attributed to each terminal node (or leaf) is computed by using the subset S of LS reaching this node: the i-th element ($i = 1,\ldots,m$) is the proportion of subwindows of S whose central pixel is in the ith class.

Prediction step. For each pixel a subwindow centered on it is extracted from the test image and its class is predicted using the Extra-Trees, by averaging the class-probability vectors returned by the trees of the ensemble and selecting the most probable class. Note that when we extract the subwindow centered on a pixel which is close to the borders of the image, we simply consider that the part of the subwindow that is out of the image is composed of black pixels.

Computational complexity. The number of operations to build M trees from a learning sample of N subwindows is $O(M\sqrt{whN}\log_2 N)$. The classification of a pixel with an ensemble of M balanced trees grown from a sample of N subwindows requires on the average $O(M \log_2 N)$ tests. The number of subwindows to extract is equal to the number of image pixels. The annotation is thus linear in the size of the image.

2.2.2 Subwindow Classification Model with Multiple Outputs (or SCMMO)

We now describe a second approach whose main difference with the previous one is that the subwindows classifier is extended so as to predict the class of all subwindow pixels.

Learning sample generation. Random subwindows of size $w \times h$ are extracted from training images as in the previous method. The difference is that the full annotation is associated to each subwindow.

Learning. Our idea here is to build a classifier able to predict at once a class for each pixel of a subwindow (instead of only the central pixel). Several extensions of decision trees have been proposed to handle multiple outputs (eg. (Geurts et al., 2006b)). These methods differ from the standard tree-growing method in the score measure used to select splits and in the way predictions are computed at tree leaves.

The score measure that we propose here is the average of the Gini entropy based score for each output pixel. More precisely, we have:

$$score(S,T) = \frac{1}{w.h}\sum_{k=1}^{w*h} score^k(S,T) \quad (4)$$

$$score^k(S,T) = G_C^k(S) - G_{C|T}^k(S) \quad (5)$$

$$G_C^k(S) = \sum_{i=1}^{m} \frac{n_{i.k}}{n_{...}}(1-\frac{n_{i.k}}{n_{...}}) \quad (6)$$

$$G_{C|T}^k = \sum_{j=1}^{2}\sum_{i=1}^{m} \frac{n_{ijk}}{n_{...}}(1-\frac{n_{ijk}}{n_{.j.}}), \quad (7)$$

where the parameters have the same meaning as in the previous section with the addition of the index $k \in 1,\ldots,wh$ that references the subwindow pixels.

Once the tree is built, class probability estimates at each subwindow position could then be obtained at each leaf simply by computing class frequencies over all subwindows that reach the leaf. However, the size of such a probability vector is equal to the number of pixels times the number of classes and thus the storage of these vectors at each leaf node could require a lot of memory space. To reduce this requirement, in our implementation, we have decided to keep track only of the majority class of each pixel together with its confidence as measured by its frequency. This simplification leads to memory requirements independent of the number of classes[2].

Prediction. The annotation process again consists in extracting subwindows and in testing each one of them with all the trees. To get a prediction for each pixel, these subwindows need at least to cover the whole image. However, we observed that the more subwindows are extracted, the better the predictions. Thus, in our experiment, we extracted all possible, overlapping, subwindows.

By testing each subwindow with all the trees, a set of tuples *(class number,rate of confidence)* is assigned to each pixel of the initial image. From this set of predictions, a vector of size m (number of different classes) is computed for each pixel: its i-th element ($i = 1,\ldots,m$) is the sum of the confidence numbers which are associated to the i-th class among all predictions. The predicted class for the considered pixel is the one that receives the highest overall confidence.

[2]We run several experiments keeping the full class probability vectors at tree leaves but no significant difference was observed.

Table 1: Main properties of the four datasets

DB	# Im.	Im. size	Protocol	# Subwin. per image
Retina	50	$[630\times420, 1386\times924]$	LOO	2000
Bronchia	8	752×574	LOO	10000
Corel	100	180×120	2×(60/40)	1650
Sowerby	104	96×64	2×(60/44)	1650

Computational complexity. With respect to the SCM approach, the complexity at the learning stage is multiplied by the number of output pixels. This leads to a complexity of $O(M(wh)^{\frac{3}{2}}N\log_2 N)$ to grow M trees from N subwindows of size $w\times h$. As in SCM, the computation of a prediction for a subwindow with M trees requires $O(M\log_2 N)$ tests. The number of subwindows is equal to $(w_i - w + 1)(h_i - h + 1)$, where w_i and h_i are respectively the width and the height of the image to annotate.

3 EXPERIMENTS

In this section, we provide experiments on four databases. After a description of the problems, we compare the two methods that are proposed and conclude the section with a discussion of computing times and the influence of the main parameters of the algorithms.

3.1 Databases

Experiments are conducted on four datasets briefly described below. The main properties of these datasets are summarized in Table 1. Examples of images are given in Figure 1.

Retina. This dataset consists of 50 microscope images of vertical sections through cat retinas[3]. Each pixel must be classified in either the *retina class* (more precisely the retina outer nuclear layer) or the *background class.*

Bronchial. The second database is the Bronchial Image Database[4], which contains 8 color images from

[3]`http://www.bioimage.ucsb.edu/research/retina_seg.php`

[4]`http://users.info.unicaen.fr/~lezoray/Databases.php`

bronchial cytology. The pixels of these images have to be classified into one of the three following classes: cell *nucleus*, cell *cytoplasm* and *background*.
Corel. The Corel database [5] contains 100 images of wild life. Seven classes are defined: *rhino/hippo*, *polar bear*, *vegetation*, *sky*, *water*, *snow*, and *ground*.
Sowerby. This database [6] contains 104 images of landscape with roads, trees, etc. The pixels must be classified in 7 classes: *sky*, *vegetation*, *road marking*, *road surface*, *building*, *street object* and *car*.

3.2 Accuracy results

In this section, we compare three methods in terms of accuracy: the two approaches presented in the previous section and, as a baseline, a method called PCM (for pixel classification model) that only uses the information about the pixel itself to predict its class, neglecting its surrounding context. Both SCM and SCMMO reduce to this baseline when the size of the subwindows is fixed to 1 ($w = h = 1$).

Results are reported in Table 2. The test protocol is given in the fourth column of Table 1. For the first two datasets, we used leave-one-out cross-validation. For Corel, we use 60 images for training and 40 images for testing and we randomized our tests twice (with different LS and TS). We use the same protocol for Sowerby with 60 and 44 images respectively for training and testing. In each case, we built 20 trees. The number of subwindows extracted per learning image (in the fourth column of Table 1) was chosen such that the learning sample contains about 100000 subwindows. We restrict ourselves to square subwindows ($w = h$) and considered four different subwindow sizes on each problem, $w \in \{1, 5, 10, 20\}$. In Table 2, we only report the best results over these sizes, together with the optimal size (w^*). For an image, the error rate is measured as the ratio between the number of incorrectly classified pixels and the total number of pixels of this image. Results in Table 2 are average error rates over all images in the test sample. Visual examples of predictions can be found in (Dumont et al., 2009).

From Table 2, it is clear that SCM and SCMMO are more efficient than the baseline pixel classifier. We also observe that SCMMO is almost always significantly better than SCM. These results are not surprising. Since PCM and SCM classify every pixel independently of the classification of its neighbors, their predictions can be very discontinuous over the image. This is well illustrated by the very noisy appearance of the PCM and SCM annotations (Dumont et al., 2009). On the other hand, SCMMO produces smoother predictions for every subwindow as it tries to predict all subwindow pixels at once. Furthermore the final pixel classification is obtained by averaging the predictions over all subwindows that cover that pixel, which further reduces the variance of the annotation.

Table 2: Accuracy results

Database	PCM	SCM (w^*)	SCMMO (w^*)
Retina	12.54%	7.53% (5)	7.56% (5)
Bronchial	3.96%	3.42% (10)	3.13% (10)
Corel	53.71%	49.43% (5)	36.01% (20)
Sowerby	17.5%	14.96% (5)	10.93% (5)

3.3 Comparison with state-of-the-art

In this subsection, we compare our results to other results presented in the literature on each dataset:
Retina: (Bertelli et al., 2007) obtained F-measures[7] on this problem ranging from 0.873 to 0.883. We obtained the following F measures: 0.769 for PCM, 0.869 for SCM, and 0.870 for SCMMO. Note however that Bertelli et al's test protocol is significantly different from ours (they used one reference image for learning and all 50 images for testing).
Bronchial: (Meurie et al., 2003) reports error rates for each class separately using 4 images for training and 4 images for testing. They obtained 29.1% for the *cytoplasm* class and 12.6% for the *nucleus* class. For comparison, SCMMO gives 23.86% for the *Cytoplasm* class and 23.1% for the *Nucleus* class.
Corel: State-of-the-art results (He et al., 2004; He et al., 2006; Shotton et al., 2006; Verbeek and Triggs, 2007) range from 20% to 33.1% error rates, evaluated on one run using the same number of images for learning and testing as ours.
Sowerby: Error rates reported in the literature ((He et al., 2004; He et al., 2006; Shotton et al., 2006; Verbeek and Triggs, 2007)) range from 10.5% to 17.6% error rates. Note that taking a larger K for the Extra-Trees method and extracting 5 times more subwindows, it is possible to decrease our error rate down to 10.3% (see Section 3.5 for a discussion of these parameters).

In most of the cases, SCMMO results are thus comparable to those obtained by the state-of-the-art. Only on the Corel problem, are our results significantly less good than others. However, it has to be

[5] `http://www.cs.toronto.edu/~hexm/label.htm`
[6] `http://www.cs.toronto.edu/~hexm/label.htm`

[7] *F-Measure* is defined in (Bertelli et al., 2007) as the harmonic mean of precision (p) and recall (r): $F = 2pr/(p+r)$.

noted that exactly the same method was applied here on the four different problems without any specific adaptation while other methods are usually evaluated on less diverse types of images. Suggestions to further improve the results of the proposed method on specific problems are given in Section 4.

3.4 Computing Times

The annotation with our method is quite fast. Computing times are obviously a function of the size of the image to annotate. In our case, the complexity is minimal as it is linear with respect to the image size. Table 3 gives the duration of the building of 20 trees and of the annotation of an image in the case of the Sowerby and Retina databases[8]. The dimension of the subwindows is 5×5. The Sowerby database contains very small images (94×64) while Retina contains larger ones (752×574 minimum). For Retina, the annotation times are given for one image of size 990×660.

For comparison, computing times for He et al.'s method on Sowerby are 24h for training and 30s per image prediction (mentioned by (Shotton et al., 2007)), TextonBoost needs 5h/10s (Shotton et al., 2006) or 20m/1.1s (Shotton et al., 2007), (Verbeek and Triggs, 2007) requires 20m/5s, while our best method, SCMMO, needs about 6m for learning and less than 1s per image for annotation.

Note that the method offers several possibilities to further reduce training and prediction times. To build trees, we could consider using a smaller subset of attributes at each split (parameter K). In the next subsection, we will also highlight the fact that a smaller number of trees could be used. To reduce the computing times at the prediction stage, we could consider subsampling subwindows in test images, in principle at the expense of classification accuracy. Some preliminary results show that it is possible to reduce this sampling rate at least by a factor of two without significant increase in error rates. It is also worth noting that our method is highly parallelizable. Indeed, tree induction, subwindow extraction and their propagation in trees are processes that could be run independently and their results subsequently aggregated.

3.5 Influence of Method Parameters

Our methods essentially depend on four parameters: the subwindow size $h \times w$, the number of trees M, the number N of extracted subwindows to build up the training set, and the Extra-Tree parameter K that filters test during tree induction.

[8]The method is implemented in JAVA and runs on a 2.4Ghz processor

Table 3: Training and prediction times on Retina and Sowerby (20 trees)

Dataset	Sowerby		Retina	
Method	Training	Prediction	Training	Prediction
PCM	10.53s	0.15s	5.34s	15.19s
SCM	76.63s	0.12s	41.22s	17.64s
SCMMO	315.68s	0.26s	125.5s	26.98s

Table 4: Influence of the Extra-Tree K parameter with SCMMO (100 000 training subwindows, 20 trees).

Database	$K = 1$	$K = \sqrt{m}$	$K = m$
Retina	8.22 (58s)	7.56 (125s)	7.23 (1219s)
Bronchial	3.27 (134s)	3.13 (994s)	3.16 (24835s)
Corel	38.00 (284s)	36.01 (9115s)	36.79 (18714s)
Sowerby	11.67 (108s)	10.93 (316s)	10.49 (3581s)

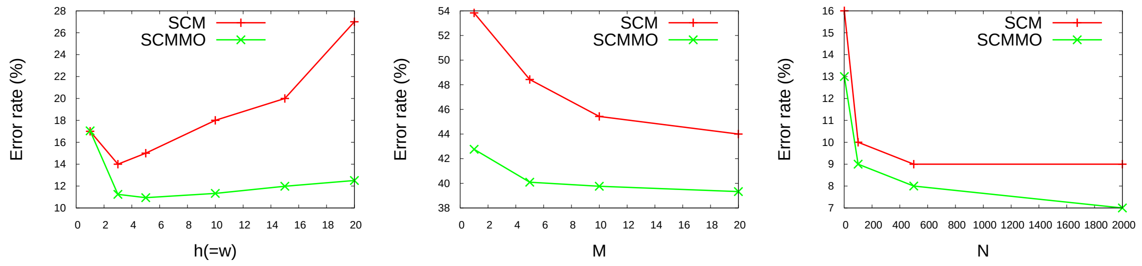
The influence of the size of the extracted subwindows is significant. In most of the cases, we observed that there exist an optimal size, which is problem dependent. This is illustrated for the Sowerby database in Figure 2.

Concerning the number of trees, since each tree of the ensemble classifier is an independent realization of the random tree growing process, the more trees are grown, the better is the accuracy of the ensemble classifier. This is clearly illustrated in Figure 2 on the Corel problem. In most cases, the error converges however quite rapidly. With SCMMO, 10 trees give already quite good results. Notice that more trees are however needed with SCM since this latter does not benefit from the additional averaging effect of multiple subwindows voting for a same pixel as in the case of SCMMO.

Concerning the number of subwindows extracted from the images in order to create the learning sample, like the number of trees, the larger it is, the better. This is illustrated in Figure 2 in the case of the Retina database. Notice that good results are typically obtained by using a learning sample containing about 100 000 subwindows. Usually, this parameter should be fixed to the maximal possible value given the available computer resources.

The Extra-Tree parameter K influences both accuracy and computing times (lower values of K require less score computations). Table 4 shows results in terms of error rates and computing times for three values of K: $K = 1$, $K = \sqrt{m}$, and $K = m$, where $m = 3wh$ is the total number of attributes. Setting K to m improves accuracy on Retina and Sowerby. The default setting $K = \sqrt{m}$ is however very close in terms of ac-

Figure 2: Left: error rate versus width (= height) of the subwindows on the Sowerby database. Middle: error rate versus number of trees on the Corel database. Right: error rates versus total number of learning subwindows on the Retina database.



curacy and brings an important improvement of computing times. Taking $K = 1$ deteriorates accuracy with respect to the two other settings but the resulting error rate might still be considered acceptable for some applications where low computing times are important. Note that this value of K amounts at building the trees in an unsupervised way (ie., not looking at the outputs when selecting the tests).

4 CONCLUSION AND FUTURE WORK

In this paper, we have introduced and compared two generic image annotation methods: SCM and SCMMO. We have shown that the idea of (Marée et al., 2005) for image classification using random subwindows and ensemble of randomized trees can be extended for pixel-wise labelling. In particular, our extension of extremely randomized trees to deal with multiple output prediction opens the door to a wide range of potential applications. Indeed, from our experiments on four distinct databases, SCMMO appears to be the most interesting one among the two proposed methods: SCMMO is almost always significantly better than SCM and by using it, we obtained good results on three problems among four. We deem that the main merit of our approach is its good overall performance while remaining conceptually very simple and keeping computing times very low. Our approach also requires less tuning than others which makes it an excellent candidate as an off-the-shelf method even if it does not reach state-of-the-art results on each and every problem. Moreover, it might be exploited as a first, fast step for image annotation as its output predictions might be post-processed by various techniques at the local and/or global level.

Although we focused on the design of a generic method, possible improvements or extensions of this framework are numerous and might be considered to improve results when dealing with a specific problem. First, as proposed by (Marée et al., 2005) in the context of image classification, it could be possible to introduce more robustness into the pixel classification models by applying random perturbations (rotation, scaling, etc.) to the subwindows or original images before the learning stage. Computing features at multiple scales or orientations or over various filter responses (like in (Dollar et al., 2006)) instead of using feature vectors composed by raw pixel values might also help to achieve a higher level of invariance and robustness to disturbances that might be required for a given application. However, such an extension would require additional computational costs.

There exist also several further degrees of freedom in the way subwindows are extracted. For instance, we could generate several models exploiting each a different size of input/output subwindows. This would allow to combine into the final model local and global information from the images. For example, some preliminary results show that combining three models built on subwindows of sizes 5×5, 2×30 and 30×2 reduces the error rate on Corel dataset with SCMMO downto 33.89%. Optimizing the number of models and the subwindow sizes might further improve the accuracy on a given problem.

Also, our multiple output model can be cast as a special case of the generic method proposed in (Geurts et al., 2006b) to handle complex kernelized output spaces. We believe that it would be interesting to exploit this method with more *ad hoc* kernels for image annotation, for example to take more explicitly into account correlations among neighboring pixel classes or any other domain specific knowledge.

Comparison with other recent image annotation methods using randomized trees should also be investigated (Shotton et al., 2008; Schroff et al., 2008).

Finally, other computer vision applications of random subwindows and tree-based ensemble methods with high-dimensional output spaces might be investigated, e.g. the detection of edges (Dollar et al., 2006).

ACKNOWLEDGEMENTS

PG and MD acknowledge the support of the Fonds National de la Recherche Scientifique (FNRS, Belgium). This paper presents research results of the Belgian Network BIOMAGNET (Bioinformatics and Modeling: from Genomes to Networks), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office. The scientific responsibility rests with the authors.

REFERENCES

Amit, Y. and Geman, D. (1997). Shape quantization and recognition with randomized trees. *Neural computation*, 9:1545–1588.

Bertelli, L., Byun, J., and Manjunath, B. S. (2007). A variational approach to exploit prior information in object-background segregation: application to retinal images. In *ICIP*.

Borenstein, E. and Malik, J. (2006). Shape guided object segmentation. In *CVPR*, pages 969–976.

Bosch, A., Zisserman, A., and Munoz, X. (2007). Image classification using random forests and ferns. In *Proc. ICCV*.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA.

Cour, T. and Shi, J. (2007). Recognizing objects by piecing together the segmentation puzzle. In *CVPR*.

Dollar, P., Tu, Z., and Belongie, S. (2006). Supervised learning of edges and object boundaries. In *CVPR*, pages 1964–1971.

Dumont, M., Marée, R., Wehenkel, L., and Geurts, P. (2009). `http://www.montefiore.ulg.ac.be/services/stochastic/pubs/2009/DMWG09/`.

Ernst, D., Marée, R., and Wehenkel, L. (2006). Reinforcement learning with raw image pixels as input state. In *Proc. International Workshop on Intelligent Computing in Pattern Analysis/Synthesis*, volume 4153 of *Lecture Notes in Computer Science*, pages 446–454.

Geurts, P., Ernst, D., and Wehenkel, L. (2006a). Extremely randomized trees. *Machine Learning*, 36(1):3–42.

Geurts, P., Wehenkel, L., and d Alché-Buc, F. (2006b). Kernelizing the output of tree-based methods. In *ICML*, pages 345–352.

He, X., Zemel, R., and Carreira-Perpinan, M. (2004). Multiscale conditional random fields for image labelling. In *CVPR*, volume 2, pages 695–702.

He, X., Zemel, R., and Ray, D. (2006). Learning and incorporating top-down cues in image segmentation. In *ECCV*, pages 338–351.

Jodogne, S., Briquet, C., and Piater, J. (2006). Approximate policy iteration for closed-loop learning of visual tasks. In *Proc. of the 17th European Conference on Machine Learning*, volume 4212, pages 222–226.

Lepetit, V. and Fua, P. (2006). Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1465–1479.

Marée, R., Geurts, P., Piater, J., and Wehenkel, L. (2005). Random subwindows for robust image classification. In *CVPR*, volume 1, pages 34–40.

Marée, R., Geurts, P., Visimberga, G., Piater, J., and Wehenkel, L. (2003). An empirical comparison of machine learning algorithms for generic image classification. In *Proc. SGAI-AI*, pages 169–182.

Marée, R., Geurts, P., and Wehenkel, L. (2007). Content-based image retrieval by indexing random subwindows with randomized trees. In *Proc. ACCV*, volume 4844 of *LNCS*, pages 611–620.

Meurie, C., Lebrun, G., Lezoray, O., and Elmoataz, A. (2003). A comparison of supervised pixels-based color image segmentation methods. application in cancerology. *WSEAS Transactions on computers, special issue on SSIP'03*, 2:739–744.

Moosmann, F., Triggs, B., and Jurie, F. (2006). Randomized clustering forests for building fast and discriminative visual vocabularies. In *Neural Information Processing Systems (NIPS)*.

Nowak, E. and Jurie, F. (2007). Learning visual similarity measures for comparing never seen objects. In *Proc. IEEE CVPR*.

Nowak, E., Jurie, F., and Triggs, B. (2006). Sampling strategies for bag-of-features image classification. In *Proc. ECCV*, pages 490–503.

Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. (2007). Object retrieval with large vocabularies and fast spatial matching. Proc. IEEE CVPR.

Schroff, F., Criminisi, A., and Zisserman, A. (2008). Object class segmentation using random forests. In *Proc. of the British Machine Vision Conference (BMVC)*.

Shotton, J., Johnson, M., and Cipolla, R. (2008). Semantic texton forests for image categorization and segmentation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Shotton, J., Winn, J., Rother, C., and Criminisi, A. (2006). Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *ECCV*, pages I: 1–15.

Shotton, J., Winn, J., Rother, C., and Criminisi, A. (2007). Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling appearance, shape and context. In *International Journal on Computer Vision*.

Verbeek, J. and Triggs, B. (2007). Scene segmentation with conditional random fields learned from partially labeled images. In *Proc. NIPS*.

Yin, P., Criminisi, A., Winn, J., and Essa, I. (2007). Tree-based classifiers for bilayer video segmentation. In *Proc. CVPR*.

Yu, S. X. and Shi, J. (2003). Object-specific figure-ground segregation. In *CVPR*, volume 2, pages 39–45.