Fast shared boosting for large-scale concept detection [*]

Hervé Le Borgne [†] Nicolas Honnorat [‡]

Abstract

Visual concept detection consists in assigning labels to an image or keyframe based on its semantic content. Visual concepts are usually learned from an annotated image or video database with a machine learning algorithm, posing this problem as a multiclass supervised learning task. Some practical issues appear when the number of concept grows, in particular in terms of available memory and computing time, both for learning and testing. To cope with these issues, we propose to use a multiclass boosting algorithm with feature sharing and reduce its computational complexity with a set of efficient improvements. For this purpose, we explore a limited part of the possible parameter space, by adequately injecting randomness into the crucial steps of our algorithm. This makes our algorithm able to handle a problem of classification with many classes in a reasonable time, thanks to a linear complexity with regards to the number of concepts considered as well as the number of feature and their size. The relevance of our algorithm is evaluated in the context of information retrieval, on the benchmark proposed into the ImageCLEF international evaluation campaign and shows competitive results.

1 Introduction

Visual concept detection[1] is a pattern classification problem which consists of assigning one or multiple labels to an image or video keyframe, based on its semantic content. In practice, the term *concept* can recover at least two types of entities, namely the *object* and the *scenes*. When the concept is an object, its detection consists of identifying the presence of at least one instance of the considered category in the image (e.g there is "a car" in this image). For the scenes, the concept detection rather relates to a general atmosphere of the image (e.g detecting whether it represents an indoor or an outdoor scene, during day or night time, whether the image seems aesthetic or fancy...). In both cases, this is a very challenging task since it has to cope with variations and changes in viewpoint (rotation, zoom, translation), occlusions and lighting or image quality. Moreover, we are interested into the general category of the concept, not the particular instance that represent it in the considered image (e.g we are interested into detecting the presence of a car, not "the grey car of M. Smith"). Numerous approaches have been

[*] This is a pre-print of the accepted version of this article. The final publication is available at www.springerlink.com

[†] CEA, LIST, Laboratory of Vision and Content Engineering.herve.le-borgne@cea.fr

[‡] MAS Laboratory, Ecole Centrale de Paris. nicolas.honnorat@ecp.fr

[1] we use the terminology of ImageCLEF [5, 19]. This problem is also known as Generic Visual Categorization [8]

proposed into the vision community [9, 16, 26, 21, 12, 18], that usually posed the problem as a set of binary classification tasks, composed of two main steps, namely visual information (feature) extraction followed by supervised learning of each concept. The learned classifier is then applied to several locations and scales within a testing image and returns a confidence score on the presence of the considered concept (in case of scenes, it can usually be applied to a very restricted number of location, even eventually once to the global image [20]).

The first step of visual information extraction is out of the scope of this paper and we refer to [18] for a comprehensive presentation of the process of local feature extraction and to [15] for examples of global features that are used in the following work. Is is quite usual to distinguish global and local features. Global descriptors include histograms of visual attributes referring to the the color, texture and shapes of the image [20, 15] and descriptors specifically adapted to visual concepts [17]. Local features extraction consists of matching some local descriptors at points of interest to the closest visual words of a dictionary computed beforehand [4]. An alternative is to characterize an image with a gradient vector derived from a generative probability model [22].

The second step consists of subsequently feeding the image representation to a discriminative classifier. Although support vector machine are nowadays the most popular classification method used in the field of image classification [16, 18, 26, 12], multi-class variant of Boosting [23, 10, 11], Random forest [3] or Nearest Neighbors methods could be used as well, as proposed in this paper. These classifiers were initially designed to discriminate two classes only, that is to say positive from negative samples. However, in practice, one often want to be able to detect several concepts at the same time. Two well-known schemes are commonly used by the vision community to deal with such a multi-class classification problem. Considering a database of N images containing from 0 to C concepts, the "one versus all" approach (e.g used in [16, 18]) consists of leaning C binary classifiers considering one concept $i = 1, \dots, C$ as positive and all others as negative. The "one against one" approach (e.g use in [26, 12]) consists of learning $C(C-1)/2$ classifiers corresponding to all possible binary problems then combining their outputs with a majority vote. These two approaches are two extreme cases of the ECOC scheme [6]. In that case, each class is assigned to a unique binary string (named a codeword). During training for an example from class i, the desired outputs of these P binary functions are specified by the codeword for class i. New values of a test sample x are classified by evaluating each of the P binary functions to generate a p-bit string s. This string is then compared to each of the C codewords, and x is assigned to the class whose codeword is closest, according to some distance measure (such as a Hamming distance), to the generated string s. "One versus all" is then the special case for with codewords of size C with a unitary L_1 norm ($\|s\|_1 = 1$). Indeed, each class i is assigned to the codeword with 1 at the i^{th} position and 0 elsewhere. To generalize the "one against one" scheme, one need to use a t-uple $-1, 0, 1$ to code the strings, $-1, 1$ giving the class membership and 0 when the class is not considered by the current binary problem. Hence, the "one against one" scheme considers all $C(C-1)/2$ possible codewords such that $\|s\|_1 = 2$ i.e with one category at 1 and one other at -1 and all the others at 0.

An important drawback of these approaches to multi-concept detection (i.e multi-class classification) is that they bear poorly scalability, in particular in terms of computational complexity and memory needs. In the following subsection we present an approach based on boosting that partially respond to this problem. However, its learning complexity is still high and the learning duration becomes prohibitive when the number of classes increases. Then in section 2 we present our method to cope with this computational complexity. The main idea is to reduce the complexity by exploring a limited part of the possible parameter space. For this, we adequately injected randomness into the crucial steps of our algorithm. An experimental validation of our approach is given in section 3. We conclude in section 4.

1.1 Boosting and the multi-class problem

A classifier is qualified as *weak* when it performs only slightly better than random guessing. Boosting consists of building a strong classifier by iteratively adding weak classifiers that focus on the harder-to-learn parts of the learning samples distribution [23, 10]. An example of weak classifier is a regression "stump" of the form:

$$h_m(v) = a\delta(v^f > \theta) + b\delta(v^f \leq \theta) \tag{1}$$

where v^f is the f'th component of the feature vector v, θ is a threshold , δ the indicator function and (a,b) are the regression parameters. Many algorithms exists to build a strong binary classifier $H(v) = \sum h_m(v)$, which the sign gives the class label (± 1) of sample v. We refer to [11] for an exhaustive presentation of the boosting principles and algorithms.

To manage the multi-class problem, an original solution was proposed in [25], consisting of finding common features that can be shared across the classes, and training jointly the classifiers. Hence, it can potentially run faster (since it computes fewer features) and requires less data to train (since it can share data across classes). More precisely, in order to learn a strong classifier $H(v,c) = \sum_{m=1}^{M} h_m(v,c)$ (which the sign gives the class label (± 1) of sample v for each class c), it solves at each iteration m the following least squares problem:

$$J_{wse} = \sum_{c=1}^{C} \sum_{i=1}^{N} w_i^c \left(z_i^c - h_m(v_i,c)\right)^2 \tag{2}$$

where $w_i^c = e^{-z_i^c H(v_i,c)}$ are the weights for sample i and for the classifier for class c, and z_i^c are the labels (± 1)for sample i for class c. Let s^* the subset of classes that are shared; a stump is now defined as:

$$h_m^{f,\theta,s^*}\left(v_i^f,\ c\right) = \begin{cases} a & \text{if } v_i^f > \theta \text{ and } c \in s^* \\ b & \text{if } v_i^f \leq \theta \text{ and } c \in s^* \\ k^c & \text{if } c \notin s^* \end{cases} \tag{3}$$

where a, b and k^c are constants that are determined analytically, trying to minimize the weighted square error (2) for fixed f, θ, s^*. The strong classifier is then determined using algorithm 1. In its simplest version, it is quite slow since it requires fitting shared stumps and evaluate the error for all the $2^C - 1$ possible subsets of classes. The authors proposed to first select the class that has the best reduction of the error, then select the second class that has the best error reduction jointly with the previously selected class, then continuing this process by adding each time the best class among those not yet selected. This greedy search procedure drastically reduces the complexity of the algorithm since only $C(C-1)/2$ subsets are explored instead of $2^C - 1$. The computation of the shared regression stumps is accelerated thanks to a propagation of the computations from the leaves of the exploration graph to higher nodes (see [25] for details of the step (a) in Alg.1), but the parameters must still be estimated for each of the D dimensions

Algorithm 1 Multiclass boosting with feature sharing [25]. N is the number of training samples, M the number of boosting rounds and C the number of classes. In the simplest (but slowest) version of the algorithm, it explores all the possible subsets at each rounds: $s \in \mathscr{S} = 1, 2, \ldots, 2^C - 1$. v_i^f is the f'th feature of the i'th sample, $z_i^c = \pm 1$ are the labels for sample i related to class c, and w_i^c are the unnormalized sample weights that gives their relative importance into the training sample.

```
I. Initialize the weights $w_i^c = 1$ and set
   $H(v\ ,\ c) = 0, i = 1..N, c = 1..C$
II. Repeat for $m = 1, 2, \ldots, M$
  (A) Repeat for the possible features $f = 1 \ldots D$
    (1) Repeat for the possible thresholds $t = 1 \ldots N_\theta$
      (a) Precompute parameters for efficient computation
          of (i) and (ii)
      (b) Repeat for the possible subsets of classes $s \in \mathscr{S}$
        (i) Fit shared stump $h_m^s(v_i, c)$
        (ii) Evaluate the error $J_{wse}(s)$
  (B) Find best subset $s^* = \arg\min_s J_{wse}(s)$
  (C) Update the class estimates :
      $H(v_i, c) := H(v_i, c) + h_m^{s^*}(v_i, c)$
  (D) Update the weights : $w_i^c := w_i^c e^{-z_i^c h_m^{s^*}(v,c)}$
```

of the features and N_θ possible thresholds θ. In the end, the total complexity of the algorithm for N training samples labeled on C classes is:

$$\mathscr{C}^0 = O\left(MDN_\theta \left[NC + C^3\right]\right) \tag{4}$$

Another algorithm of shared boosting was presenting in [24], showing similar limitations in terms of learning complexity.

2 Fast learning for multi-class boosting

According to (4), the shared multi-class boosting algorithm exhibits a cubic growth rate with the number of classes, which makes it unsuited to this scalability. This section presents an extension of a previous work in which we proposed several strategies to overcome this limitation [14]. The general idea developed in this section is to reduce the complexity by injecting randomness into the steps of the algorithm that are the most time consuming, in order to explore a limited part of the possible space.

Choosing a random value can be considered as reasonable in some cases, as L. Breiman shown with bagging [2] and random forest [3]. Bagging, that is based on bootstrapping [7], consists of a random selection (with replacement) of the training set to learn several classifiers then combining them by simple majority voting. A close idea proposed by Ho with the Random Subspace Method [13] is to select randomly a subset of features in the training set to learn classifiers, then combing them by majority voting as well.

In the case of Boosting, classifiers and training sets are obtained in a deterministic way. The proposed algorithm (see Alg. 2) introduces random selection of the parameter space of the boosting process, i.e in the choice of the classifiers parameters. Instead of trying “all possible settings” for these parameters, we either determine a reasonable

Algorithm 2 Fast multiclass boosting with feature sharing. N is the number of training samples, M the number of boosting rounds and C the number of classes. In the simplest (but slowest) version of the algorithm, it explores all the possible subsets at each rounds: $s \in \mathscr{S} = 1, 2, \ldots, 2^C - 1$. v_i^f is the f'th feature of the i'th sample, $z_i^c = \pm 1$ are the labels for sample i related to class c, and w_i^c are the unnormalized sample weights that gives their relative importance into the training sample.

```
I. Initialize the weights $w_i^c = 1$ and set
    $H(v\ ,\ c) = 0, i = 1..N, c = 1..C$
II. Repeat for $m = 1, 2, \ldots, M$
  $(A)$ Repeat for $p = 1 \ldots N_f$
    (a) Choose one feature $f_p$ in $1 \ldots D$
    (b) Choose one threshold $t_p$ in $1 \ldots N_\theta$
    (c) Precompute parameters for efficient computation
        of (ii) and (iii)
    (d) Repeat for $k = 1 \ldots N_k$
      (i) Choose a subset of classes $s_{k,p} \in \mathscr{S}$
      (ii) Fit shared stump $h_m^{f_p, t_p, s_{k,p}}(v_i^{f_p}, c)$
      (iii) Evaluate the error $J_{wse}(f_p, t_p, s_{k,p})$
  $(B)$ Find best weak classifier $h_m^{f^\star, t^\star, s^\star} = \arg\min_{f,t,s} J_{wse}(f, t, s)$
  $(C)$ Update the class estimates :
      $H(v_i, c) := H(v_i, c) + h_m^{f^\star, t^\star, s^\star}(v_i, c)$
  $(D)$ Update the weights : $w_i^c := w_i^c e^{-z_i^c h_m^{f^\star, t^\star, s^\star}(v, c)}$
```

value or, when no a priori choice seems better than an other, we pick a limited set of random values, compute a performance criterion and finally keep the best one. Accordingly to the boosting principle, the proposed algorithm builds a strong classifier by iteratively adding weak classifiers. At a given step, the learning samples are weighted such that those which are wrongly classified become dominant in the computation of the new weak classifier. In our case we use shared stumps defined according to Equation (3). Hence, at each step, three parameters must be determined : a feature number f, a threshold t, and a sharing subset of class s.

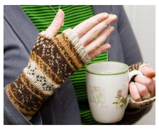
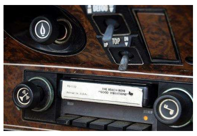
The choice of the feature number f_p (step (a) of Alg. 2) can not be done fully randomly. Hence, we chose to use the Fisher criterion on a random sharing of the class space, repeating this process N_f times. Fist, a subset of class s is chosen randomly. Let us note $\bar{s}$ the complement of s, $m_s(f)$ the mean value of the image feature vectors v_i^f that belongs to one of the classes of s and $v_s(f)$ their variance. The final choice is then :

$$f_p = \max_{f \in 1 \ldots D} \frac{[m_s(f) - m_{\bar{s}}(f)]^2}{v_s(f) + v_{\bar{s}}(f)} \tag{5}$$

It only requires to pre-compute the sum (and its square) for each feature only once, then the mean and variance can be computed faster at each of the N_f random choices.

The threshold t_p of the stump (step (b) of Alg. 2) is chosen randomly, within the range of data for the previously chosen feature number. The process is repeated N_f times, in the same loop as the choice of feature number. We verified that in practice, it does not deteriorate the performances of the algorithm significantly, in comparison with an exhaustive search on all the possible values for this threshold (see section 3.2.

Figure 1: Example of pictures and their annotated concepts.

			
No Visual Season	No Visual Season	No Visual Season	No Visual Season
Indoor	Indoor	Outdoor	Outdoor
Day	No Visual Time	Night	Day
Neutral Illumination	Neutral Illumination	Neutral Illumination	Neutral Illumination
No Blur	Partly Blurred	No Blur	Partly Blurred
No Persons	No Persons	Single Person	No Persons
		Citylife	Animals
	Still Life		Aesthetic Impression

To determine the final best stump/weak classifier, we also pick randomly N_k possible subsets of classes $s_{k,p} \in \mathscr{S}$ (step (d) of Alg. 2) and retain the weak classifier (defined by a feature number $f^\star$, a threshold $t^\star$ and a class subset $s^\star$) that has the lowest cost $J_{wse}(f^\star, t^\star, s^\star)$ among the $N_f \times N_k$ possibilities:

$$h_m^{f^\star,t^\star,s^\star} = \underset{\substack{p=1\ldots N_f \\ k=1\ldots N_k}}{\arg\min} J_{wse}(f_p, t_p, s_{k,p}) \tag{6}$$

We then update the class estimate by adding the chosen weak classifier (stump) to the strong classifier and finally update the sample weights before the next step. Considering that the complexity of step (step (c) of Alg. 2) is $O(NC)$, the final complexity of our algorithm is thus:

$$\mathscr{C}^1 = O\left(M\left[NDC + N_f[\alpha D + \varepsilon + NC + \beta N_k]\right]\right) \tag{7}$$

where α, ε and β are some constants corresponding to the unitary cost of respectively steps (a), (b) and (d).

One could be surprised that the finally chosen share set s^* may be different from that one that has been used to compute the Fisher criterion at step (a). In fact, the cost of fitting a shared step is much lower than the one to compute the Fisher criterion ($\beta << \alpha$) thus it makes sense to do it. Moreover, in practice, we noticed it improves the performances of the algorithm.

3 Experimental validation

We are interested in the problem of visual concept detection. For the experiments presented here we used the ImageCLEF VCDT experimental paradigm. The vcdt08 database [5] contains 1827 images for training and 1000 images for testing, with 17 concepts to detect. The vcdt09 database [19] has 53 concept categories, 5000 learning images and 3000 testing ones. The categories are not exclusive, i.e one image can contain several of the concepts (see figure 1).

The global feature we use are described in [15] and consist of a color histogram taking into account the spatial arrangement of pixels (*cime* descriptor, size 64) and a

Table 1: Average learning time (second / experiment) averaged across 20 experiments (plus standard deviation), on vcdt08 for different values of N_f and N_k, at $M = 100$ and $M = 1000$ iterations. One experiment only was conducted for [25]: it has constant performances for a given database.

		$N_f = 5$	$N_f = 50$	$N_f = 500$
$M = 100$	$N_k = 10^2$	4.9 ± 0.04	6.7 ± 0.2	23.3 ± 0.7
$M = 100$	$N_k = 10^3$	5.0 ± 0.04	7.3 ± 0.1	25.0 ± 0.4
$M = 100$	$N_k = 10^4$	5.8 ± 0.1	13.6 ± 0.4	87.2 ± 0.6
$M = 100$	$N_k = 10^5$	11.6 ± 0.1	73.6 ± 0.4	681 ± 3.2
$M = 100$	[25] with greedy heuristic			73.2
$M = 100$	[25] without heuristic			11966
$M = 1000$	$N_k = 10^2$	45.5 ± 0.2	56.2 ± 0.5	161.9 ± 4.9
$M = 1000$	$N_k = 10^3$	46.1 ± 0.2	62.4 ± 0.5	220.8 ± 4.1
$M = 1000$	$N_k = 10^4$	52.0 ± 0.3	121.5 ± 0.6	812.4 ± 6.2
$M = 1000$	$N_k = 10^5$	111.0 ± 0.3	711.6 ± 3.3	6802 ± 7.9
$M = 1000$	[25] with greedy heuristic			718.9
$M = 1000$	[25] without heuristic			131905

combination of texture and color descriptors (*tlep*, size 576). We also considered the use of local descriptors by computing SIFT features then a classical K-means to create a visual dictionary [26], resulting into bag-of-sift (*BoSIFT* descriptor, size $K = 5000$). Points of interest are detected with a Harris-Laplace detector in addition to 1000 points randomly distributed over the image. This last refinement aims to better describe the concept that concerns the wholes images, their general aspect or their atmosphere. Finally, we considered the descriptor *BoSURF* based on a dense grid (every 6 pixels) of local SURF feature [1]. To create the final descriptors of fixed size, we computed three dictionaries of size $K = 50$; $K = 500$ and $K = 5000$, using the same K-means algorithm as for the bag-of-sift dictionary.

The performance of the algorithm is measured with the equal error rate (EER) that indicates that the proportion of false acceptances is equal to the proportion of false rejections. The lower the equal error rate value, the higher the accuracy of the system. All experiments were conducted on one core of a Intel Q9400 CPU @ 2.66 GHz with 4 GB memory.

3.1 Exp 1: parameter choice

At the core of Alg. 2, there are two parameters related to random choices: N_f is the number of random choices on the feature number and N_k is the number of random choices of class sharing subsets to fit the stump and compute the total cost. From equation (7), one can see they influence the complexity, but can also influence the performances of the algorithm. This experiment aims at studying this influence.

In practice we used the vcdt08 benchmark ($N = 1827$, $C = 17$) with the global feature only ($D = 640$). We measured the performances at $M = 100$ and $M = 1000$ iterations. We conducted the experiment 20 times and reported the average learning time in table 1 and the average EER in table 2 for different values of N_f and N_k. The standard deviation was computed over the 20 experiments.

As expected, the computation time increases with N_k and N_f. However, for a given set of parameters, the time is quite constant since the standard deviation is very low in

Table 2: Average EER (multiplied by 100) on vcdt08 for different values of N_f and N_k (the lower the better), at $M = 100$ and $M = 1000$ iterations. The average and standard deviation was computed across 20 experiments.

		$N_f = 5$	$N_f = 50$	$N_f = 500$
$M = 100$	$N_k = 10^2$	28.3 ± 0.5	26.8 ± 0.6	26.2 ± 0.4
$M = 100$	$N_k = 10^3$	27.1 ± 0.4	26.3 ± 0.4	26.0 ± 0.4
$M = 100$	$N_k = 10^4$	26.6 ± 0.4	25.8 ± 0.4	25.7 ± 0.4
$M = 100$	$N_k = 10^5$	26.6 ± 0.5	25.7 ± 0.4	25.6 ± 0.4
$M = 100$	[25] with greedy heuristic			26.0
$M = 100$	[25] without heuristic			25.8
$M = 1000$	$N_k = 10^2$	24.6 ± 0.3	24.1 ± 0.4	24.3 ± 0.3
$M = 1000$	$N_k = 10^3$	24.3 ± 0.3	24.2 ± 0.4	24.3 ± 0.4
$M = 1000$	$N_k = 10^4$	24.1 ± 0.3	24.0 ± 0.3	24.1 ± 0.4
$M = 1000$	$N_k = 10^5$	24.1 ± 0.4	24.0 ± 0.3	24.1 ± 0.4
$M = 1000$	[25] with greedy heuristic			24.8
$M = 1000$	[25] without heuristic			24.7

comparison with the average learning time (table 1).

The same experiment was conducted with the algorithm of [25] that obtained an EER of 26.0 in 73.2 seconds at $M = 100$ and 24.8 in 719 seconds at $M = 1000$. With the parameters $N_f = 50$ and $N_k = 10^4$ we obtain similar or better performances five to six times faster (13.6 sec. at $M = 100$ and 122 sec at $M = 1000$). Moreover, from equation (4) and (7) we induce that this difference would grow with the number of concepts (C) to detect, showing the interest of our algorithm for large-scale visual concept detection (see also section 3.5). One can also notice that the couple $(N_f = 50, N_k = 10^4)$ seems sufficient on this experiment since multiplying them by ten leads to similar results. It may be different with larger feature descriptors (i.e when D increases).

Globally, the performances improve when N_k and N_f grow. Because of the randomness injected into the algorithm, the results vary from one experiment to another. This can be seen as a drawback in comparison with [25] that has non-varying performance. However, our method can also be seen as advantageous since it allows to discover some parameter configurations that are better than those of the greedy algorithm. At convergence ($M = 1000$), the parameters $(N_f = 50, N_k = 10^4)$ allows to obtain significantly better results (from 24.8 to 24.0 ± 0.3).

3.2 Exp 2: random selection of threshold

As explained before, it is usually worth to choose the threshold (step II.A.b of Alg. 2) fully randomly. To prove this, we conducted the same experiment as before (section 3.1), with the parameters ($M = 1000$, $N_f = 50$ and $N_k = 10^4$), but determined values of thresholds were used. The thresholds were chosen according to order statistics computed on learning data. Preliminary experiments (not reported here) showed this is much more efficient than a linear distribution between the minimal and maximal value, since it divides data into $N_{th} + 1$ essentially equal-sized data subsets. Several numbers of threshold N_{th} were tested $(1, 3, 5, 10, 20)$. Hence, when $N_{th} = 1$ the threshold is the median, for $N_{th} = 3$ we add the first and third quartiles, and so on. The experiments were repeated 20 times. The average EER and the standard deviation across the 20 experiments was reported into table 3.

Choosing $N_{th} = 1$ threshold (i.e the median) gives similar results in term of computation time but the performance are much poorer than those obtained with our algorithm. With $N_{th} = 20$ the performances become comparable but the computation time for learning is much larger. Hence, choosing randomly the threshold can be considered as a good strategy most of the time. In the following, this is the only strategy used.

Let note that the variance of the results is similar when the threshold is chosen randomly or according to order statistics. This suggests that this variance is mainly due to the randomness injected to determine the sharing subset of classes s.

3.3 Exp 3: convergence

This experiment aims at determining the number of iteration necessary to reach the convergence. It was conducted on the vcdt09 benchmark ($N = 5000$, $C = 53$) with features of different sizes D. We fixed the parameters of the algorithm to ($N_f = 50$ and $N_k = 10^4$). The number of iteration varied from $M = 25$ to $M = 1000$ and we computed the EER every 25 iterations. Results are reported on Fig. 2.

With the CIME descriptor ($D_{cime} = 64$) the algorithm converges around 200 iterations, that is to say between two and three D_{cime}. We see similar behavior with the other global descriptor TLEP, for which the algorithm converges in about D_{tlep} iterations. For the local SIFT-based descriptor the convergence is relatively even faster since the algorithm is converging around 700 iterations with a descriptor of size $D_{BoSIFT} = 5000$. However the raw performance with this descriptor is quite disappointing as it remains below the others. We obtained better performances with the bag-of-SURF descriptor that converges in 400 iteration for the three visual dictionary sizes considered ($D_{BoSURF} = 5000$,$D_{BoSURF} = 500$ and $D_{BoSURF} = 50$). The performances are slightly better with the dictionary of size 500.

As mentioned below, the scope of this paper focuses on the learning method and does not concerns the quality of the descriptors. It seems nevertheless obvious that better local descriptors needs to be used for a powerful application. The interesting point of this experiment is to show that our algorithm converges in a few hundreds of iterations even when the descriptor size reaches several thousands.

3.4 Exp 4: influence of the number of classes

This experiment aims to study the influence of the number of classes. For two different descriptors, tlep and bag-of-SIFT, we selected ten times a restricted number of classes among the 53 available in vcdt09. The choice was made randomly. Then we learned and tested on this set of classes with the parameters ($N_f = 50, N_k = 10^4$) and computed the average and standard deviation of the equal error rate resulting from these experiments (see figure 3).

Table 3: Learning time (second / experiment) and Equal Error Rate (EER) averaged across 20 experiments (plus standard deviation), on vcdt08 for N_{th} thresholds placed at order statistics. Parameters are ($M = 1000$, $N_f = 50$ and $N_k = 10^4$) and we recall the result with random choice (see table 2 and 1).

N_{th}	1	3	5	10	20	1 (rand)
EER	26.4 ± 0.4	24.8 ± 0.4	24.6 ± 0.4	24.4 ± 0.3	24.1 ± 0.4	24.0 ± 0.3
Time	127.2 ± 1.1	272.7 ± 1.6	420.2 ± 5	781.8 ± 5.1	1522 ± 4.9	121.5 ± 0.6

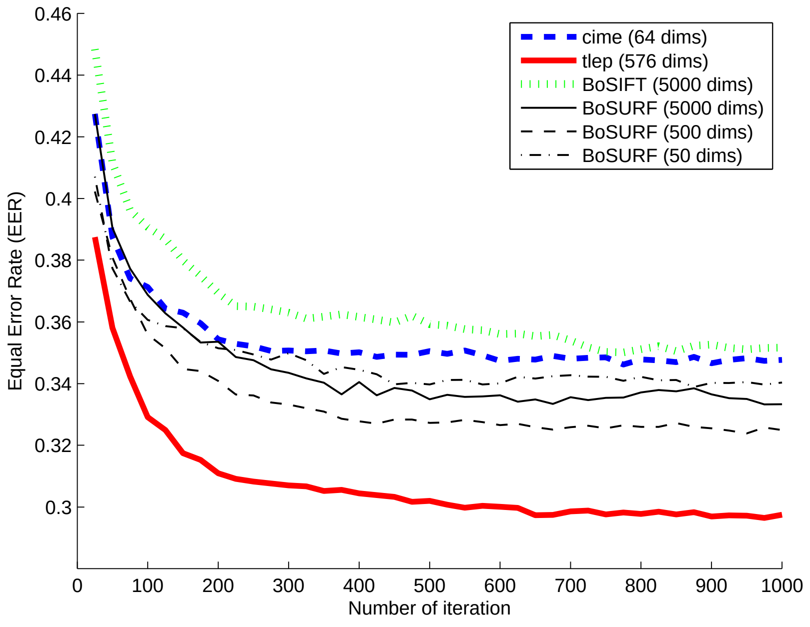


Figure 2: EER on vcdt09 according to different descriptor sizes and number of iteration.

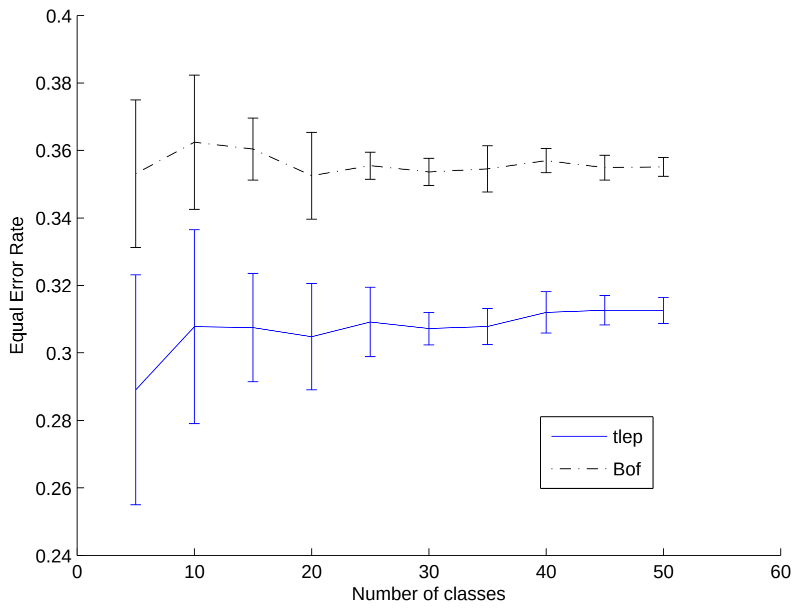


Figure 3: EER on vcdt09 according to the number of classes considered, for two descriptors.

Table 4: EER ($\times 100$) on vcdt09 for our method and one-versus-all approach. Or method was repeated 20 times and the results are the average and standard deviation over the 20 runs

	tlep	BoF
our method	31.4 ± 0.2	36.94 ± 0.3
one-versus-all	46.05	48.64

The decay of the standard deviation simply reflects the class variety in terms of classification difficulty drops down when more classes are took into account. The interesting result is that the average EER value is almost the same whatever the value of the number of classes, showing the remarkable stability of the algorithm even when the number of classes grows.

3.5 Exp 5: interest of sharing classes

In this experiment, we compared our algorithm to a one-versus-all approach on the same feature (since the goal is to compare the learning algorithm, independently of the used features). We thus run our algorithm in a one-versus-all approach, with he parameters $(N_f = 50, N_k = 10^4)$ and $M = 200$ iterations for each classifier. The results (table 4) are drastically in advantage of our method (same parameters but $M = 200$ for all the classes, making it much faster).

In fact, such a difference can be explained in the context of concept detection. Indeed, a given image can contain several concepts, and some of them can overlap. Hence, during the learning of concept A, images containing concept B can be both in the positive and the negative set. Excluding such images from the learning set would lead to reduce the number of learning images for concept A. The problem is that the number of images to learn each concept would drop down when the number of classes increases. It is worth stating that when one want to detect a large number of concept simultaneously, the probability that the concepts overlap inevitably increases. Sharing classes thus becomes increasingly relevant in the case of a large number of (overlapping) concepts to detect.

4 Conclusion

We proposed a fast learning algorithm based on shared boosting and showed its interest in the context of visual concept detection. We adequately injected randomness into the crucial steps of our algorithm in order to reduce the exploration of its parameter space. We conducted several experiments on the ImageCLEF benchmarks 2008 and 2009 that showed the efficiency of our method in the context of visual concept detection.

It is much faster than other learning algorithms while maintaining a similar or better accuracy. In particular, in the context of visual concept detection for which several concepts can be present within an image, our approach allows to keep a significant quantity of learning images. For a given database, the accuracy is stable when the number of classes grows. Its complexity grows linearly with the number of classes to detect, making it suitable to tackle with large-scale visual concept detection problems. We also showed that our algorithm usually converges in a few hundreds of iterations even when the descriptor size reaches several thousands. This is quite promising for the use of a large number of visual features, in particular when one considers that the complexity of learning increases, as well, linearly with the total size of features.

Last, our work explores the benefit of injecting randomness into the choice of the boosting parameters. One of the main challenges in future work will consist of reducing the variance from one run to another or at least being able to control it in function of the "amount" of randomness injected. More generally, this principle that has previously been applied to data or feature choice, is quite promising and could be explored beyond allowing faster learning.

5 acknowledgements

We would like to thank the two reviewers for their fruitful comments. We acknowledge support from the ANR (project Yoji) and the DGCIS for funding us through the regional business cluster Cap Digital (projects Géorama and Roméo).

References

[1] Bay, H., Ess, A., Tuytelaars, T., Gool, L.V.: Surf: Speeded up robust features. Computer Vision and Image Understanding **110**(3), 346–359 (2008)

[2] Breiman, L.: Bagging predictors. Machine Learning **2**(24), 123–140 (1996)

[3] Breiman, L.: Random forests. Machine Learning **1**(45), 5–32 (2001)

[4] Dance, C., Willamowski, J., Fan, L., Bray, C., Csurka, G.: Visual categorization with bags of keypoints. In: ECCV International Workshop on Statistical Learning in Computer Vision (2004)

[5] Deselaers, T., Hanbury, A.: The visual concept detection task in imageclef 2008. In: CLEF working notes, pp. 531–538. Aarhus, Denmark (2008)

[6] Dietterich, Bakiri: Solving multiclass learning problems via ecocs. Journal of AI Research (2), 263–286 (1995)

[7] Efron, B., Tibshirani, R.J.: An Introduction to the Bootstrap. Chapman & Hall, New York (1993)

[8] Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html

[9] Fergus, R., Perona, P., Zisserman, A.: Object class recognition by unsupervised scale-invariant learning. In: IEEE Conference on Computer Vision and Pattern Recognition (2003)

[10] Freund, Y.: Boosting a weak learning algorithm by majority. Information and Computation (121), 256–285 (1995)

[11] Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: A statistical view of boosting. Annals of Statistics (28), 337–407 (2000)

[12] van Gemert, J.C., Geusebroek, J.M., Veenman, C.J., Smeulders, A.W.M.: Kernel codebooks for scene categorization. In: European Conference on Computer Vision (2008)

[13] Ho, T.K.: The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence **20**(8), 832–844 (1998)

[14] Honnorat, N., Le Borgne, H.: Accélérer le boosting avec partage de caractéristiques. In: CORESA, pp. 203–208. Toulouse, France (2009)

[15] Joint, M., Moëllic, P.A., Hède, P., Adam, P.: Piria: a general tool for indexing, search and retrieval of multimedia content. In: SPIE ElectronicImaging. San Jose, California, USA (2004)

[16] Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: IEEE Conference on Computer Vision and Pattern Recognition (2006)

[17] Le Borgne, H., Guérin Dugué, A., O'Connor, N.: Learning midlevel image features for natural scene and texture classification. IEEE T. on Circuits and Systems for Video Technology **17**(3), 286–297 (2007)

[18] Marszałek, M., Schmid, C., Harzallah, H., van de Weijer, J.: Learning object representations for visual object class recognition. In: Visual Recognition Challenge workshop, in conjunction with ICCV (2007)

[19] Nowak, S., Dunker, P.: Overview of the clef 2009 large scale visual concept detection and annotation task. In: CLEF working notes. Corfu, Greece (2009)

[20] Oliva, A., Torralba, A.: Modeling the shape of the scene: a holistic representation of the spatial envelope. International Journal of Computer Vision **3**(42), 145–175 (2001)

[21] Perronnin, F.: Universal and adapted vocabularies for generic visual categorization. IEEE T. on Pattern Analysis and Machine Intelligence **7**(30), 1243–1256 (2008)

[22] Perronnin, F., Dance, C.: Fisher kernels on visual vocabularies for image categorization. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8 (2007)

[23] Schapire, R.: Strength of weak learnability. Machine Learning (5), 197–227 (1990)

[24] Shotton, J., Winn, J., Rother, C.: A. criminisi.textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. International Journal of Computer Vision **1**(81), 2–23 (2009)

[25] Torralba, A., Murphy, K.P., Freeman, W.: Sharing visual features for multiclass and multiview object detection. IEEE T. on Pattern Analysis and Machine Intelligence **5**(29), 854–869 (2007)

[26] Zhang, J., Marszałek, M., Lazebnik, S., Schmid, C.: Local features and kernels for classication of texture and object categories: a comprehensive study. International Journal of Computer Vision **2**(73), 213–238 (2007)