

Incremental Kernel Learning for Active Image Retrieval without Global Dictionaries

Philippe-Henri Gosselin, Frédéric Precioso, Sylvie Philipp-Foliguet

▶ **To cite this version:**

Philippe-Henri Gosselin, Frédéric Precioso, Sylvie Philipp-Foliguet. Incremental Kernel Learning for Active Image Retrieval without Global Dictionaries. Pattern Recognition, 2011, 44 (10-11), pp.2244-2254. 10.1016/j.patcog.2010.12.006 . hal-00619211

HAL Id: hal-00619211
https://hal.science/hal-00619211

Submitted on 6 Sep 2011

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.

Incremental Kernel Learning for Active Image Retrieval without Global Dictionaries

P.H. Gosselin[*,a], F. Precioso[a], S. Philipp-Foliguet[a]

[a] *ETIS, CNRS, ENSEA, Univ Cergy-Pontoise, F-95000 Cergy-Pontoise*

Abstract

In content-based image retrieval context, a classic strategy consists in computing off-line a dictionary of visual features. This visual dictionary is then used to provide a new representation of the data which should ease any task of classification or retrieval. This strategy, based on past research works in text retrieval, is suitable for the context of batch learning, when a large training set can be built either by using a strong prior knowledge of data semantics (like for textual data) or with an expensive off-line pre-computation. Such an approach has major drawbacks in the context of interactive retrieval, where the user iteratively builds the training data set in a semi-supervised approach by providing positive and negative annotations to the system in the relevance feedback loop. The training set is thus built for each retrieval session without any prior knowledge about the concepts of interest for this session. We propose a completely different approach to build the dictionary on-line from features extracted in relevant images. We design the corresponding kernel function, which is learnt during the retrieval session. For each new label, the kernel function is updated with a complexity linear with respect to the size of the database. We propose an efficient active learning strategy for the weakly supervised retrieval method developed in this paper. Moreover this framework allows the combination of features of different types. Experiments are carried out on standard databases, and show that a small dictionary can be dynamically extracted from the features with better performances than a global one.

[*]Corresponding author

Email addresses: gosselin@ensea.fr (P.H. Gosselin), frederic.precioso@ensea.fr (F. Precioso), philipp@ensea.fr (S. Philipp-Foliguet)

Key words:
Multimedia retrieval, Interactive retrieval, Visual dictionaries, Machine learning, Kernel function

---

1. Introduction

Kernel-based methods for multimedia retrieval have shown their robustness for many tasks, in shape recognition [37], image retrieval [14], or event detection [29] for instance. Most methods first build a kernel function, usually from supervised data [5], then train a classifier such as Support Vector Machines (SVM). This procedure is relevant in the context of batch learning, when a large training set is provided. However, in the context of interactive retrieval, building an initial kernel before any retrieval session is a difficult task, since there is no initial training set. Indeed, in this specific context, the training set is iteratively built through interactions with the user who provides labels on unknown data. A classic strategy is to consider highly discriminant kernel functions, such as thin Gaussian kernels, which give the classifier the ability to shatter the database. The main drawback of such a strategy is the weak generalization capacity of the retrieval system.

The other solution we want to explore in this paper is to build the kernel function during the retrieval session and thus to adapt it to the current training set.

The framework proposed in this paper is general, but we will give applications for global descriptions of data using histograms. We will give examples of kernel designs for these histograms.

A common process when dealing with histograms is to first build a dictionary of visual words on a training set, using clustering methods, such as K-means [15] or randomized forests [28]. Each data in the database is then represented by a histogram on this visual dictionary. Then, a kernel function, RBF or Fisher kernel, is considered. Such strategies proved to be very powerful in a batch learning context, when a large training set is available [20, 33, 34]. In [32], Perronnin first proposed to build two kinds of visual dictionaries per category: one "universal" visual dictionary is build on the whole database with a Gaussian mixture model (GMM) without any supervision, while the other one dedicated to the category uses labelled training data to learn the GMM of that category. In [33], Perronnin et al. applied Fisher kernel on visual words modeled with a combination of "universal" and "categorical" GMMs, restricted to diagonal variance matrices for each component of the mixture. Deriving a diagonal approximation of the Fisher matrix of a GMM, they obtain a $(2d+1) \times k - 1$ dimensional vector representation of an image feature set, or $d \times k$ dimensional vector when considering only the components associated with either the means or the variances of

the GMM. In a very recent work [34], Perronnin et al. further improved the Fisher kernel in several ways. Inspired by these works, [20] Jegou et al. proposed a simpler approach in which a K-Means algorithm and a new descriptor aggregation approximate the universal GMM. This last method is a good approximation of Perronnin et al. sophisticated GMM models and produces comparable results.

However, in our context of interactive retrieval, we have very few assumptions both on the concepts the user is looking for and on the database which is completely unknown at the beginning. Hence, using a single dictionary computed for all concepts does not seem judicious, since concepts themselves are unknown.

A solution could be to compute a new dictionary based on the features from the images labelled by the user, at each feedback loop. One could, for example, consider very recent work of Mairal et al. [26] which iteratively builds a dictionary using on-line techniques in order to speed up the dictionary computation. However, this method is not adapted to the active learning context since all the histograms must be recomputed at each iteration with the new dictionary, and thus there is no strategy for selecting the optimal samples for annotation.

Another idea could be to incrementally design a kernel function directly on training data from base kernels (seen as weak-learners), using the boosting paradigm as in Crammer et al. approach [9]. However, as in the classic boosting algorithm [38], in order to determine penalties and weights involved in boosted kernel construction at each iteration, the current kernel has to be evaluated on all previously labelled data from the training set. The computation time of this evaluation increases along the interactive learning process and that is what we want to avoid.

In this paper we propose a general framework for interactive image retrieval with three main contributions. First we give a way to dynamically build the dictionary from visual words extracted from labelled images. Secondly we incrementally build the kernel function from the visual words added at each feedback step. The resulting kernel is the same as the one computed from scratch, except that the computational complexity is linear with respect to the size of the database. Thirdly, as this incremental kernel learning method allows an efficient active learning strategy, we propose a function which selects the images that contain the most interesting features. Section 2 gathers the works related to the three contributions, which are themselves presented in section 3. In section 4 we present an application of the method

using low-level features such as color or texture vectors. Finally, in sections 5 and 6, we present experimental results carried out on generalist databases. These results show that the proposed method performs better than the classic one where a global dictionary is used, even if the global dictionary is optimally tuned.

2. RELATED WORK

2.1. Iterative dictionary building

Our aim is to provide an algorithm able to build a single dictionary adapted to active learning, that is to say, able to be the most generative regarding the class the user wants to retrieve but also the most discriminative to all other possible classes. Furthermore, since we are considering active learning framework, our dictionary learning has to be fast and compliant with weakly-supervised context.

Dictionary learning algorithms received recently a lot of consideration mainly due to the increase of dataset size in classification context. If the interest is mainly focused on second-order iterative batch approaches for their computational speed compared to first order gradient descent methods [23], such methods cannot deal with very large training sets since the whole training set is required at each iteration to minimize the empirical risk (or more usually a surrogate function of this risk) [2]. In [27], the authors introduce a discriminative approach to supervised dictionary learning that effectively exploits the corresponding sparse signal decompositions in image classification tasks, and propose an effective method for learning a shared dictionary. The key idea is to learn simultaneously a single shared dictionary and models for different signal classes in a mixed generative and discriminative formulation.

A recent approach [26] has been proposed to iteratively build a dictionary by processing one element (or a small subset) of the training set at a time. As mentioned by the authors: such online approach *"is particularly important in the context of image and video processing [35], where it is common to learn dictionaries adapted to small patches, with training data that may include several millions of these patches (roughly one per pixel and per frame)"*.

However, Mairal et al. [26] approach suffers from excessive computational cost regarding active learning requirements. Indeed, during the dictionary update steps, the new dictionary D_t is computed by minimizing a surrogate cost function of the empirical cost function (under some constraints) using both the new annotated data $\mathbf{x}_t$ and its decomposition $\boldsymbol{\alpha}_t$ over the dictionary

D_{t-1} obtained at the previous iteration. These dictionary update steps can be speeded up using D_{t-1} as a warm restart to compute D_t. However, classical sparse coding steps require to compute the decomposition $\boldsymbol{\alpha}_i$ of $\mathbf{x}_i$ over the dictionary D_{i-1} and thus the minimization (computed by LARS algorithm) on all possible $\boldsymbol{\alpha}$ values in $\mathbb{R}^k$:

$$\boldsymbol{\alpha}_i \triangleq \underset{\boldsymbol{\alpha}\in\mathbb{R}^k}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{x}_i - D_{i-1}\boldsymbol{\alpha}\|_2 + \lambda\|\boldsymbol{\alpha}\|_1$$

where $\lambda \in \mathbb{R}$ is a regularization parameter.

Such computation is definitely not suitable to our context, where we target quasi real-time response from the system. Both the dictionary update and the sparse decomposition of the data over the dictionary have to be real-time compliant.

2.2. Kernel learning

In this paper, we focus on kernel-based learning techniques. The main motivation is that, once one has a kernel function, a lot of powerful learning methods can be used. This is also a good framework for reasoning and creating new methods. More specifically, we are interested in methods for learning kernels.

A first class of methods related to this task is distance learning techniques (which respects the triangle inequality), since kernels are easily built from distances. Such methods have been proposed for learning Mahalanobis distances:

$$d(\mathbf{x}, \mathbf{x}')^2 = \sum_r (\mathbf{x}_r - \mathbf{x}'_r)^\top A(\mathbf{x}_r - \mathbf{x}'_r) \tag{1}$$

with $\mathbf{x}$ and $\mathbf{x}'$ image indexes and $A \succeq 0$ a positive semi-definite matrix, that can be learnt using optimization [43, 11, 24] or boosting [17]. The method in [11] is interesting for our context, since the algorithm seems to be able to update matrix A for each new label. All values of matrix A are subject to change, and thus all distance values between labelled and labelled/unlabelled images must be recomputed.

A second class of methods learns a combination of kernel functions (a.k.a. multiple kernel learning), for instance using optimization [22, 1, 21]. In most cases, a linear combination of minor kernels is performed:

$$K(\mathbf{x}, \mathbf{x}') = \sum_j \beta_j k_j(\mathbf{x}, \mathbf{x}') \tag{2}$$

Weights β_j are computed according to the current labels, and to unlabelled data for semi-supervised algorithms. In many cases, the computation of weights β_j can be performed with complexity at most linear with the database size. All weights β_j are subject to change, and all kernel values between labelled and labelled/unlabelled images must be recomputed.

A third class of methods performs the learning of the Gram matrix $G_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$, for instance using boosting [9] or optimization [1]. With such methods, like the method we propose in section 3, we work on the row $\mathbf{r}_j = (G_{ij})_i$ of the Gram matrix which corresponds to the labelled images or on an approximation with eigen decomposition. That means that these methods directly produce the kernel values we are interested in. The method in [9] proposes a boosting algorithm for learning a linear combination of Gram matrices. This method is interesting since it creates weak hypothesis on the fly, during boosting rounds, which is similar to our aim of building a dictionary (and hence new indexes) during retrieval. However, all boosting rounds need to be run again at each feedback step.

Most of the kernel learning methods we presented in this section are able to learn new parameters for kernel building in reasonable time for our context, sometimes thanks to some modifications/approximations. Next to that an update of distance/kernel values must be performed with complexity $O(m \times n)$, with m the number of labelled images and n the number of images in the database. To the best of our knowledge, there is no method to perform these operations with a lower complexity. Furthermore, most of these methods do not directly deal with dynamic indexes, and thus full re-computations are required when indexes are changed, contrary to the method we propose in section 3.

2.3. Active learning

Many strategies have been proposed to select the best image to be labelled as positive or negative. They are usually described as "pessimistic" methods, "optimistic" methods, or as a combination of both.

"Pessimistic" methods will not necessarily lead to the fastest increase of performance at each feedback iteration, but they take a minimal risk. They ensure that a minimal performance is reached after a given number of labels. The most common technique of this class consists in selecting the documents the closest to the classifier boundary [7]. This strategy is used for instance with SVM [40], kNN [19] and boosting [25, 8], with extensions to deal with multiple image selection [3, 18] and with multi-label learning [42, 44]. It

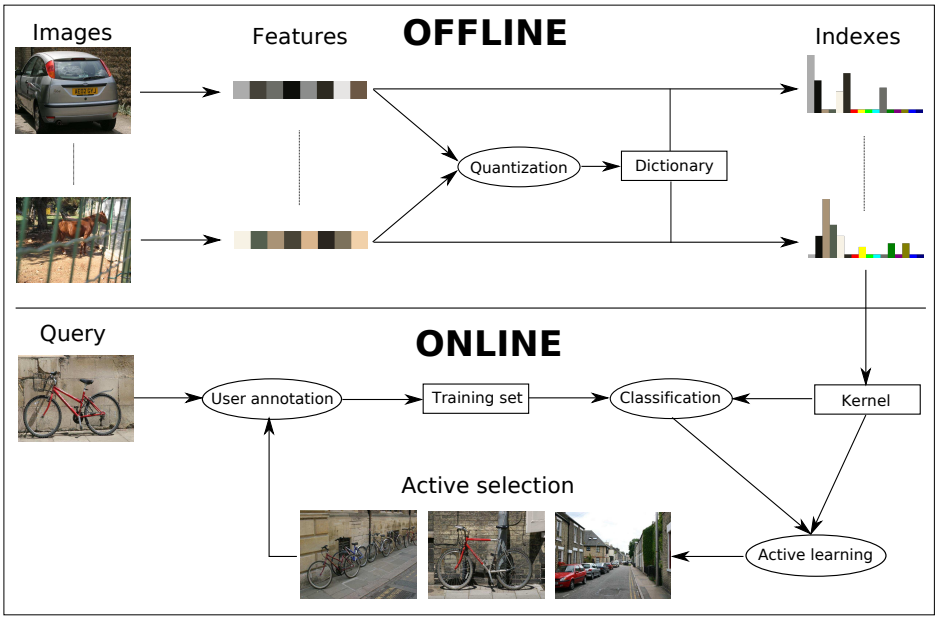
has been proven that selecting the most uncertain images is asymptotically the fastest strategy for dividing Version Space in the case of SVM [40] and boosting [25]. This strategy is optimal if one can select a document on the classifier boundary, but usually there is no document right on the boundary, and the constraint is relaxed by choosing a document near the boundary. The selection criterion can be the distance to the boundary [40], the entropy/probability [19], or samples where several classifiers agree/disagree [6]. That means that the theoretical optimum is never reached, but this strategy is still able to very well identify areas containing the most interesting documents to label, even if there are few labels (about 20–40) [40, 25]. Methods have been proposed to deal with the case of very few labels (2 or more) [13].

"Optimistic" active learning methods can lead to the fastest increase of performance, but they do not ensure that a minimal performance is reached after a given number of labels. Criteria of selection can be the minimization of expected classification error [36], the maximization of mutual information about the labels of the unlabelled documents [16], or the maximization of expected average precision [13]. The main drawback of these methods is that an unexpected label can lead to no improvement. For instance, let us consider the selection of an image that, once labelled as positive, would enhance a lot the retrieval. If the user labels this image as positive, the classification is much improved, whereas if it is labelled as negative, there is no improvement. Since the efficiency of these methods depends on all previous labels, the latter case may happen more frequently when there are very few labels.

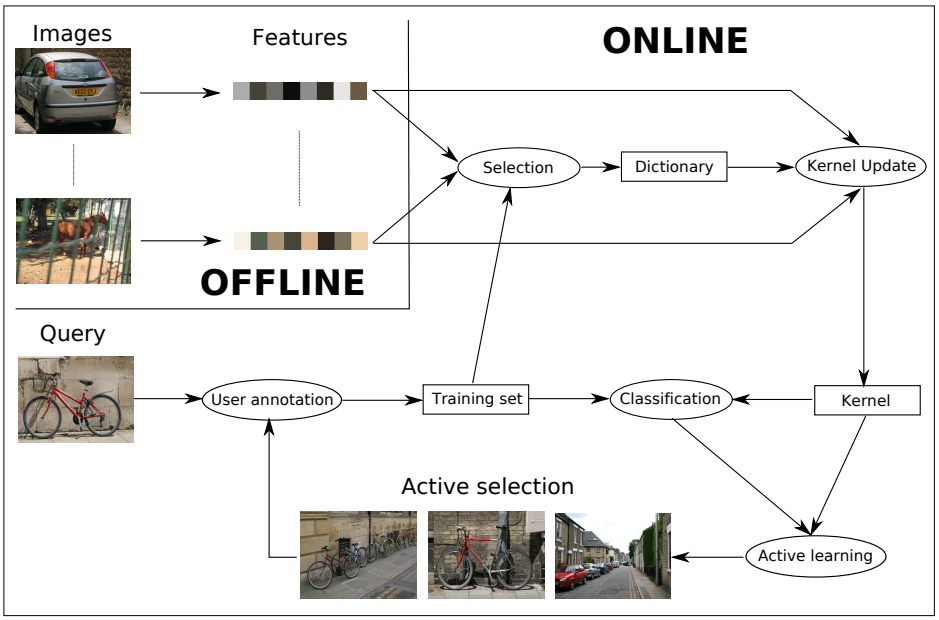
Considering advantages and drawbacks of these two classes of active learning methods, the combination of them is interesting [13]. Uncertainty-based methods perform well when selecting the interesting areas of document space. Hence, one can preselect the images the closest to the classifier boundary. Selecting within these preselected images (thanks to an uncertainty criterion) is subject to the inaccuracy of classifier boundary, especially with few labels. Then, an optimistic strategy can be used to select within the preselected images. When using such a combination of strategies, highly promising images can be selected, while ensuring a minimal improvement.

3. PROPOSED FRAMEWORK

In this section, we present the general framework to build a dictionary of visual words and to learn a kernel function simultaneously during a retrieval



(a) Classic learning scheme with a global dictionary



(b) Proposed learning scheme with a dynamic dictionary

Figure 1: Learning schemes for interactive image retrieval.

session.

In Fig. 1(a) we present a classic learning scheme for interactive retrieval. It is composed of two main stages: off-line and on-line stages. The off-line stage is performed once and only once before any retrieval session. During this stage, features are first extracted from images (in the example, colours). Then a dictionary of features is computed, generally using K-Means. Since there is no label, the usual criterion for word computation is the distortion (or equivalent), which aims at minimizing the difference between a feature and its corresponding word. Then, features are projected onto the dictionary in order to get indexes, such as histograms. The on-line stage is performed each time a new retrieval session begins. Retrieval session is initialized with some images of the category (in the example of Fig. 1, a single image of bicycle), an the database is classified according to these first labels. Then the active learner selects unlabelled images that, once labelled, will enhance the most the classification of the database. The user labels these images, which are added to the current training set. These steps are repeated until the user is satisfied or tired.

In Fig. 1(b) we present the learning scheme we propose to use in this paper. The main difference is that we move all the processes related to the dictionary from the off-line stage to the on-line stage. Such a change allows to perform the dictionary computation with the information brought by user labels. Even if we have very few labels in the interactive retrieval, this information can still slightly improve the relevance of the dictionary for the currently searched category. Of course, as we focus on real time image retrieval, full re-computation of indexes is not an acceptable solution. In our case, we use kernels for classification, and hence we only compute what is required to update the previous kernel to the new one.

3.1. Dynamic dictionary

In this section, we assume that each image i is represented by a set $P_i = \{\mathbf{p}_{ri}\}_r$ of features $\mathbf{p}_{ri}$ in feature space $\mathbb{P}$. For instance, P_i can be the set of the most representative color vectors of image i. These features are precomputed for all images of the database, or computed on the fly for query images outside the database.

The first aim of the method is to build dictionary $D_L = \{\hat{\mathbf{p}}_l\}_{l\in[1,L]}$ of features $\hat{\mathbf{p}}_l$, but with the constraint that they are features from images of the database. We call words the features selected to enter the dictionary. The first advantage of such a strategy is that all possible words (and latter

minor kernels) are known before any retrieval session, and thus allow the use of cache algorithms to save computational time.

At the beginning of a retrieval session, the dictionary D_0 is empty. Then, at each feedback step, we add to the dictionary features from images labelled as positive. They are added as soon as they are different from the ones already in the dictionary. As we will see with experiments, positive labelled images seem to contain most of the interesting features.

We denote $g_D(\mathbf{p})$ a feature selection function which returns 1 if feature $\mathbf{p}$ can be added to dictionary D, and 0 otherwise. Examples of such functions are presented in section 4.

3.2. Incremental kernel learning

For each new word $\hat{\mathbf{p}}_l$, we build a minor kernel function $k_{\hat{\mathbf{p}}_l}(P_i, P_j)$, which is the similarity between image i and image j relatively to $\hat{\mathbf{p}}_l$. We sum all these minor kernels to get the kernel according to current dictionary D_L:

$$K_L(P_i, P_j) = \sum_{l=1}^{L} k_{\hat{\mathbf{p}}_l}(P_i, P_j) \tag{3}$$

with L the number of words in dictionary D_L.

The incremental computation of this kernel is straightforward:

$$K_{L+1}(P_i, P_j) = K_L(P_i, P_j) + k_{\hat{\mathbf{p}}_{L+1}}(P_i, P_j) \tag{4}$$

If $\mathcal{P}$ is the space of feature sets, minor kernels $k_{\hat{\mathbf{p}}_l}(P_i, P_j)$ are computed using evaluation function $e_{\hat{\mathbf{p}}_l} : \mathcal{P} \to \mathbb{R}$ defined for feature $\hat{\mathbf{p}}_l$ and similarity function δ by :

$$k_{\hat{\mathbf{p}}_l}(P_i, P_j) = \delta(e_{\hat{\mathbf{p}}_l}(P_i), e_{\hat{\mathbf{p}}_l}(P_j)) \tag{5}$$

The simplest evaluation function $e_{\hat{\mathbf{p}}_l}$ equals 1 if $\hat{\mathbf{p}}_l$ is in P_i and 0 otherwise. This formula can be used for dictionary of visual words, where one only checks whether keyword $\hat{\mathbf{p}}_l$ belongs to image P_i. In the case where an image is represented by a set of colours, $e_{\hat{\mathbf{p}}_l}(P_i)$ can return the number of pixels whose color is close to $\hat{\mathbf{p}}_\mathbf{l}$.

The similarity function δ compares the evaluation of images i and j regarding feature $\hat{\mathbf{p}}_l$. This function must be chosen so that the expansion of function K_L in Eq. (3) is a kernel function. Thus function $e_{\hat{\mathbf{p}}_l}$ does not need to satisfy any mathematical property to lead to a kernel function.

3.3. Active learning strategy

In this section, we present a method to perform active learning within our kernel learning framework. We denote by $\mathbf{y} = (y_i)_i$ the label vector, with $y_i \in \{-1, 0, 1\}$ the label of image i (1 for relevant images, -1 for irrelevant images, and 0 when there is no label).

Active learning aims at selecting the unlabelled images which, once labelled by the user and thus added to the training set, will improve at most the classifier. This can be expressed as the minimization of a function $a(i)$ over unlabelled images :

$$i^\star = \underset{\{i|y_i=0\}}{\operatorname{argmin}}\, a(i) \tag{6}$$

$i^\star$ is the index of the selected image. A common active learning function selects the images the closest to the classifier boundary. When the classifier is SVM, $a(i)$ can be the distance of image i to the separating hyperplane. In this case, we have the SVM_{active} method proposed by Tong [40].

Because of the inaccuracy of the classifier boundary with very few labels, we apply a correction using the method in [13]. Then, we preselect the 100 images the closest to this corrected boundary. The selection within these 100 images uses the following active learning method.

For each feature $\mathbf{p}$ in unlabelled images, we compute the kernel target alignment between the labels and the kernel $k_\mathbf{p}$ (*cf.* Eq. (5)). This criterion, proposed by Cristianini [10], is a measure of similarity between a kernel and a set of labels. The higher the alignment is, the higher the kernel $k_\mathbf{p}$ "fits" the category represented by the labels $\mathbf{y}$:

$$A_\mathbf{y}(k_\mathbf{p}) = \frac{\langle k_\mathbf{p}, \mathbf{y}\mathbf{y}^\top \rangle}{\sqrt{\langle k_\mathbf{p}, k_\mathbf{p} \rangle \langle \mathbf{y}\mathbf{y}^\top, \mathbf{y}\mathbf{y}^\top \rangle}} \tag{7}$$

with $\langle ., . \rangle$ the dot product.

Since labels do not change during active learning selection, the active selection function can be written as:

$$a(t) = \underset{\substack{\mathbf{p} \in P_t \\ g_D(\mathbf{p}) = 1}}{\operatorname{argmin}} \frac{\sum_i \sum_j y_i y_j \delta\left(e_\mathbf{p}(P_i), e_\mathbf{p}(P_j)\right)}{\sqrt{\sum_i \sum_j \delta\left(e_\mathbf{p}(P_i), e_\mathbf{p}(P_j)\right)}} \tag{8}$$

with $g_D(\mathbf{p})$ a feature selection function which returns 1 if feature $\mathbf{p}$ can be added to dictionary D, and 0 otherwise (see section 4.1).

3.4. Bound the dictionary

In order to save memory, we propose a method to bound the size of the dictionary. The main idea is to compute the kernel target alignment for each word $\hat{\mathbf{p}}_l$ of the dictionary, and only to use the words that maximize the alignment.

Considering the kernel target alignment computation time, since we have very few labels, their re-computation is not a problem. For enabling a word, the update is the same as before (*cf.* Eq. (4)). For disabling a word $\hat{\mathbf{p}}_l$, the update is the following:

$$K_{L+1}(P_i, P_j) = K_L(P_i, P_j) - k_{\hat{\mathbf{p}}_{L+1}}(P_i, P_j) \tag{9}$$

At the end of the section 5.2.2 we present experiments which evaluate the performance according to the maximum number of words in the dictionary.

4. APPLICATION

In this section, we present an application of the method for dynamic histograms.

4.1. Dynamic Histograms

In this application, descriptors are vectors, such as colour or texture. The aim of this application is to iteratively build the kernel function on histograms during the retrieval.

With classic histogram-based features, a dictionary is built for the whole database, before any retrieval session. This is performed using some parametrization, for instance the number of words in the dictionary. As we will show in experiments, performance for each retrieval session depends on this parametrization. With the method we proposed in section 3.1, we get rid of global dictionaries.

Before any retrieval session, we extract and quantize the descriptors within an image using K-Means with Euclidean distance d. In the following experiments, we keep 64 descriptors per image. Then, we build for each image i a set P_i of features $\mathbf{p}_{ri}$ with:

$$\mathbf{p}_{ri} = (\mathbf{f}_{ri}, h_{ri}, \theta_{ri}) \tag{10}$$

where $\mathbf{f}_{ri}$ is the center of cluster r computed by K-Means, h_{ri} the number of vectors in cluster r, and θ_{ri} the distance of $\mathbf{f}_{ri}$ to the closest cluster center $r' \neq r$.

These image feature sets $P_i = \{\mathbf{p}_{ri}\}_i$ are computed off-line.

During the on-line retrieval, if D denotes the current dictionary, we only add features from images labelled as relevant if they are not similar to any feature of D. To consider two features $\mathbf{p}$ and $\hat{\mathbf{p}}$ as similar, we check if thresholds θ and $\hat{\theta}$ are close enough and if descriptors $\mathbf{f}$ and $\hat{\mathbf{f}}$ are close enough according to these thresholds. In this case, the feature selection function g_D is:

$$g_D(\mathbf{p}) = \begin{cases} 1 & \text{if } \forall \hat{\mathbf{p}}_l = (\hat{\mathbf{f}}_l, \hat{h}_l, \hat{\theta}_l) \in D,\ \frac{|\hat{\theta}_l - \theta|}{\hat{\theta}_l + \theta} > 0.8 \text{ or } d(\hat{\mathbf{f}}_l, \mathbf{f}) > \hat{\theta}_l \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

Such a feature selection function can add clusters with similar centers but with different sizes, since parameter θ can be seen as the size of the hypersphere around the cluster center. That means that, for instance, a first center can focus on a very specific blue color, while another one focuses on all blue hues. Then, if the user is looking for a car with a very specific blue color, the first cluster is the most interesting. In another case, if the user is looking for blue skies, the second cluster is the most interesting.

Evaluation value $e_{\hat{\mathbf{p}}_l}(P_i)$ of each image P_i according to the new word $\hat{\mathbf{p}}_l$ first aims at checking if one of the descriptors of image i is close to word $\hat{\mathbf{p}}_l$. In this case, $e_{\hat{\mathbf{p}}_l}$ returns the number of descriptors of P_i close to $\hat{\mathbf{p}}_l$:

$$e_{\hat{\mathbf{p}}_l}(P_i) = \begin{cases} h_{r^\star i} & \text{if } d(\hat{\mathbf{f}}_l, \mathbf{f}_{r^\star i}) \leq \hat{\theta}_l \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

with $r^\star = \text{argmin}_r\, d(\hat{\mathbf{f}}_l, \mathbf{f}_{ri})$.

In the example of color histograms, $g_W(\mathbf{p})$ equals 1 if color $\mathbf{f}$ is far enough to each color already in D, and $\mathbf{x}_i = (e_{\hat{\mathbf{p}}_l}(P_i))_l$ is similar to the histogram of image i regarding dictionary D.

4.2. Incremental kernel

We present in this section two similarities δ (*cf.* Eq. (5)) that respectively lead to a triangular and a Gaussian kernel (*cf.* Eq. (3)).

4.2.1. Triangular χ^1 incremental kernel

If the evaluation function always returns positive values, the following function δ can be used, with $(x, y) \in \mathbb{R}^2$:

$$\delta_{\chi^1}(x, y) = -\frac{|x - y|}{x + y} \tag{13}$$

This function is very interesting since it is invariant to scale:

$$\forall \alpha > 0,\ \delta_{\chi^1}(\alpha x, \alpha y) = \delta_{\chi^1}(x, y)$$

Thus, the method is invariant to the magnitudes of evaluation function values. For instance, even if a first evaluation function $e_{\hat{\mathbf{p}}_l}$, corresponding to a first feature $\hat{\mathbf{p}}_l$, returns values around 10^{-5} and another one $e_{\hat{\mathbf{p}}_k}$, corresponding to feature $\hat{\mathbf{p}}_k$, returns values around 10^5, both feature spaces still have the same weight in kernel K_L.

This function leads to a triangular kernel with χ^1 distance:

$$\begin{aligned} K_L(P_i, P_j) &= \sum_{l=1}^{L} k_{\hat{\mathbf{p}}_l}(P_i, P_j) \\ &= -\sum \delta_{\chi^1}(e_{\hat{\mathbf{p}}_l}(P_i), e_{\hat{\mathbf{p}}_l}(P_j)) \\ &= -d_{\chi 1}(\mathbf{x}_i, \mathbf{x}_j) \end{aligned}$$

with d_{χ^1} the χ^1 distance between two vectors, and $\mathbf{x}_i = (e_{\hat{\mathbf{p}}_l}(P_i))_l$.

4.2.2. Gaussian L^2 incremental kernel

The method can also be used to incrementally build a Gaussian L^2 kernel, using the following function δ, with $(x, y) \in \mathbb{R}^2$:

$$\delta_{L^2}(x, y) = -\frac{1}{2\sigma^2}(x - y)^2 \tag{14}$$

If we compute the exponential value of kernel $K_L(P_i, P_j)$, we get a Gaussian L^2 kernel:

$$\begin{aligned} K'_L(P_i, P_j) &= \exp\left(K_L(P_i, P_j)\right) \\ &= \exp\left(\sum_l \delta_{L^2}(e_{\hat{\mathbf{p}}_l}(P_i), e_{\hat{\mathbf{p}}_l}(P_j))\right) \\ &= \exp\left(-\frac{1}{2\sigma^2} d_{L^2}(\mathbf{x}_i, \mathbf{x}_j)\right) \end{aligned}$$

with d_{L^2} the L^2 distance, and $\mathbf{x}_i = (e_{\hat{\mathbf{p}}_l}(P_i))_l$.

Algorithm 1 Evaluation protocol for one category

Given: category C, training set A, test set T, number of feedbacks F, number of labels per feedback s

For each image c of $C \cap A$ do

- Create a new retrieval session with no labels
- Initialization:
  - Label c as positive
  - $I_0^\star = s$ closest images to c in A
  - Label images of $I_0^\star \cap C$ as positive, otherwise as negative
- For f from 1 to F do
  - Train a new classifier using current labels
  - Sort images of T using the classifier
  - Compute Precision/Recall curve $pr(f, c)$ on T
  - $I_f^\star = s$ images selected in A using active learning
  - Label images of $I_f^\star \cap C$ as positive, otherwise as negative

Output: Average Precision/Recall curves $PR(f) = \underset{c}{\text{average}} \Big(pr(f, c) \Big)$

5. EXPERIMENTS ON INTERACTIVE RETRIEVAL

5.1. Evaluation Protocol

We set up an evaluation protocol to estimate the average performance one can expect when starting an interactive retrieval session with a random image. This is performed by the simulation of retrieval sessions for each category, where the user labels the images selected by the active learning technique. A Precision/Recall curve is computed at each feedback step, and then averaged over all retrieval sessions for the same category. Then, the average quality of the ranking is computed with the usual criterion of Average Precision, which is used for example in TRECVID evaluation campaign [30]. At last, in order to have a global quality measure of our system, we compute the Mean Average Precision (MAP) on all categories. Labelling is

only performed on the train set, and performance is computed on the test set. Details of the evaluation protocol are shown in Algorithm 1.

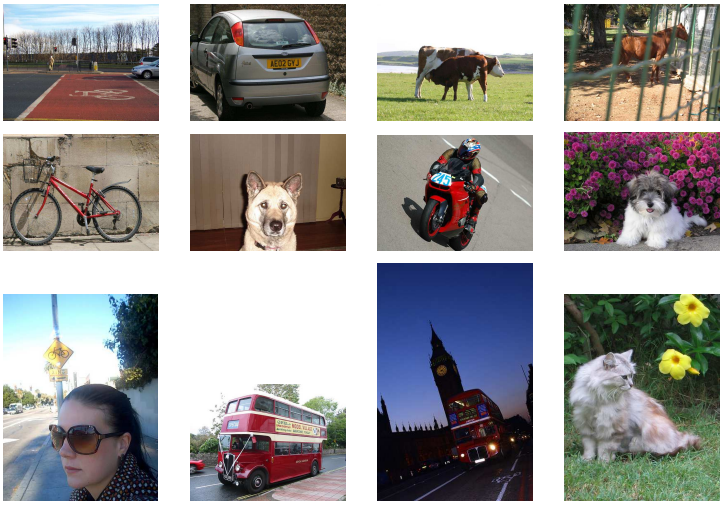


Figure 2: Images from VOC 2006 database.

5.2. Results on VOC2006

We carried out experiments on the VOC 2006 database [12] which contains images belonging to 10 categories. It is split into 1,277 train images and 2,686 test images. Images from this database are shown in Fig. 2. Let us note that we do not follow any of the PASCAL evaluation protocols since we are interested in interactive image retrieval.

We compared two histogram-based methods, one using global dictionaries, and the other one using our dynamically built histogram. In both cases we used $L^\star a^\star b^\star$ colours, textures from quaternion wavelets [4], and SVM to classify the database according to the current labels. Results per category are presented in Table 1 for 10 feedback steps, and results according to each feedback step are presented in Figure 3.

5.2.1. Global dictionary

The first method precomputes a global dictionary with a fixed number of color and texture clusters. We use dictionary sizes of 16, 32, 64, 96, 128, 160, 192, 224 and 256, and consider all combinations of two features with

different dictionary sizes. The kernel function is a triangular kernel with χ^1 distance:

$$K_{\chi^1}(\mathbf{x}, \mathbf{y}) = -\sum_r \frac{|\mathbf{x}_r - \mathbf{y}_r|}{\mathbf{x}_r + \mathbf{y}_r}$$

The active learning method is the one of [14]. This is an enhancement of Tong's SVM_{active} method[40], a reference strategy for interactive retrieval. These two papers compare several interactive retrieval methods and show the large increment brought by active learning strategies.

We display three of the 81 combinations of dictionaries in Table 1 : 64 colours/64 textures, 256 colours/256 textures, and 64 colors/160 textures. The last one is the combination which gives the best overall MAP among the 81 combinations. Performance can be very different from one parametrization to another, for instance for the "bicycle" category, it goes from 32% to 46%. This shows that the performance is highly dependent on the parametrization of the global dictionaries.

Fig. 4 shows the average precision for several combinations of dictionary sizes. One can see that results are not regular according to dictionary sizes. This is due to the learning of dictionary: even if we use a K-Means algorithm which gives almost stable dictionaries [31], they are still subject to some variations. However, one can see that there is emerging combinations. Let us note that this behaviour is the same for all categories.

5.2.2. Dynamic Histograms

The second method is the one we proposed in section 4.1 with a triangular kernel with χ^1 distance (*cf.* Eq. 13). Results are shown in the fourth column of Table 1. With this method, no parametrization is needed. On average, our method performs better than the method with global dictionaries, even if global dictionaries are perfectly tuned. That means that, whatever the cross validation performed to optimally tune global dictionaries, performance is better with our dynamic dictionaries.

Let us note that our method is less effective for two categories ("cow"and "sheep"). This can be explained by the fact that these categories can be retrieved with few features (color and texture of grass for instance). That also shows that the method builds discriminative kernels, which is consistent with interactive retrieval, where the user expects to retrieve the images he labelled as positive. That means that the method takes little risk, and selects the features which increase at most the similarity between positive labels and

decrease as much as possible the similarity between negative labels. Since only few labels are available for training, in special cases such as "cow"and "sheep"the method does not detect generic features which are relevant for a large category.

We also carried out experiments to evaluate the performance of dynamic dictionaries when their sizes are bounded. The results are shown in Table 2. We tried several maximal numbers of words : 16, 32, 64, 96, and 128. Let us notice that, without bounding, the size of each dictionary goes up to 200 with 51 labels. Results are almost stable for bounding values higher than 96 words, and even with few words, we still have good performance.

Fig. 5 shows the average computational time of each feedback step using dynamic dictionaries, when running the system on a single core of an Intel Core 2 Duo with 2GB of memory. The first feedback step is the longest to compute, since all visual words are new, and their evaluation and kernel values need to be computed. For the next feedback steps, only the evaluation and kernel values for the new visual words are computed, the previous values being stored into memory. In order to store these values up to the available memory, we use classic caching algorithms. In order to get average computation time for a single search, we empty caches at the beginning of each retrieval session during these experiments. In the other case, where caches are shared among users, the average time for one feedback step is around 10 milliseconds, the same computational time as for static dictionaries.

	Dictionary sizes			
Category	64/64	256/256	64/160	Dynamic
bicycle	43	45	44	**48**
bus	48	57	56	**64**
car	65	69	70	**79**
cat	28	28	30	**31**
cow	**47**	35	40	41
dog	22	23	23	**25**
horse	22	24	23	**25**
motorbike	46	55	50	**61**
person	33	35	34	**37**
sheep	48	38	**48**	36
all	40	41	42	**45**

Table 1: Average Precision(%) on the test set of VOC2006 database. Initialization with 1 image, 10 feedbacks, 5 labels per feedback. Except for the last column, results are shown for global dictionaries. The two numbers in column heads (for instance '160/64') are the sizes of the color and texture dictionaries. The last column shows results using the method proposed in this paper, where dictionaries are built during the retrieval.

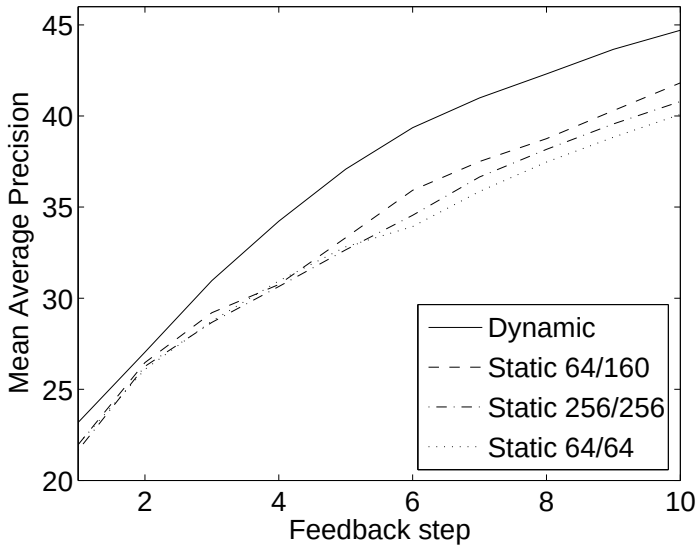
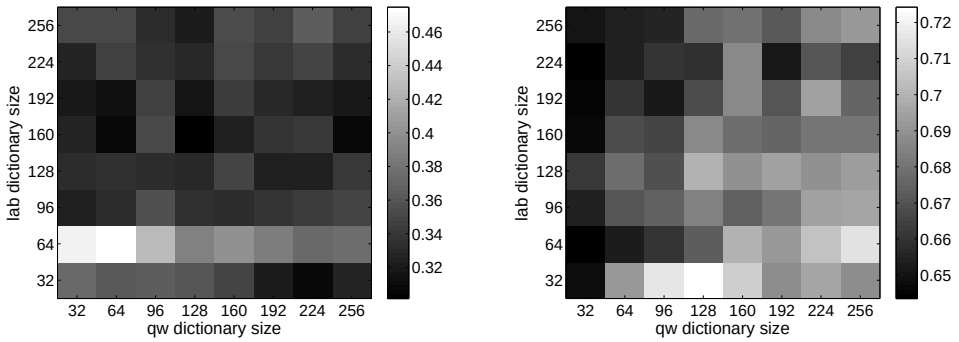


Figure 3: Mean Average Precision(%) on the test set of VOC2006 database, according to each feedback step. Initialization with 1 image, 10 feedbacks, 5 labels per feedback. "Dynamic" refers to the proposed method, "Static X/Y" refers to a global dictionary with "X" color and "Y" texture words.



(a) Category "cow" (b) Category "car"

Figure 4: Average Precision on the test set of VOC 2006 database. Each square is the result for a combination of two dictionaries. A lighter square means a better performance than a darker one.

	Max. number of words					
Category	16	32	64	96	128	$+\infty$
bicycle	38	41	46	47	47	**48**
bus	42	50	60	63	62	**64**
car	69	73	76	77	79	**79**
cat	25	28	28	30	30	**31**
cow	25	29	33	37	38	**41**
dog	22	22	24	24	25	**25**
horse	18	21	22	23	24	**25**
motorbike	45	53	57	58	60	**61**
person	31	33	34	36	36	**37**
sheep	29	33	35	35	36	**36**
All	34	38	41	43	44	**45**

Table 2: Average Precision(%) using the proposed method on the test set of VOC2006 database, according to the maximum number of words allowed in dynamic dictionaries. For experiments corresponding to the last column, the average number of words in each dictionary is between 150 and 200. Initialization with 1 image, 10 feedbacks, 5 labels per feedback.

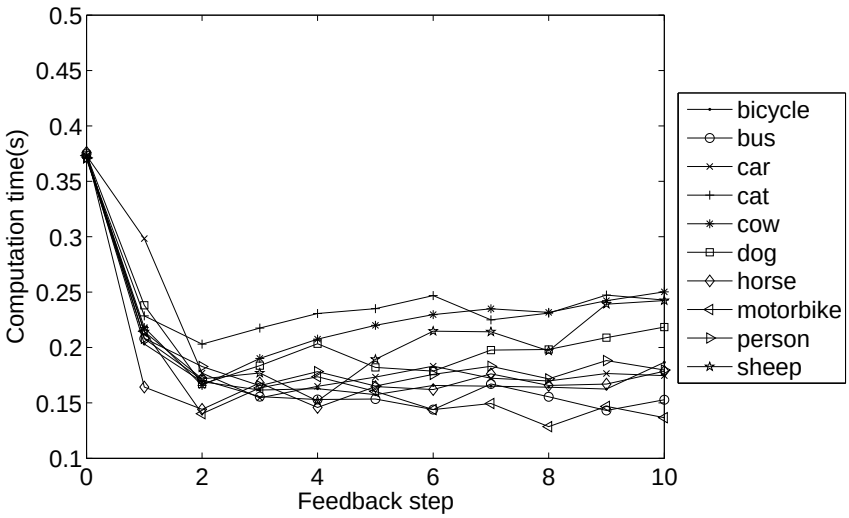


Figure 5: Average computational time of each feedback step with dynamic histograms.



Figure 6: Images from the Web.

5.3. Results on Images from the Web

We carried out experiments on the database presented in [39], which contains images belonging to 18 categories. We only considered non-abstract images labelled as "good" or "ok", which leads to a total of 10610 images. In comparison to VOC2006 images, most images have a poor quality (low resolution and high compression). Some images from this database are presented in Fig. 6. The experimental protocol is the same as the one we used for VOC2006, except that performance is measured on the train set (there is no test and validation sets).

Results are presented in Table 3. Let us first remark that we have lower results than with the previous database and the same features, certainly because of the poor quality of images. We first carried out experiments with global dictionaries of color and textures. We considered all combinations of one color dictionary and one texture dictionary with 64, 128, 256 and 512 words. The first column presents Average Precision using the worst combination, the second one a combination one may choose considering the database size, and the third column the best combination. Again, we observed large differences from one global dictionary tuning to another. It seems that a small color dictionary and a large texture dictionary is the most suited tuning for that database. The last column presents Average Precision using the proposed method, which does not require any global dictionary tuning. As

for VOC2006, the proposed method gives the best Mean Average Precision.

	Dictionary sizes			
Category	512/64	256/256	128/512	Dynamic
airplane	14	18	**20**	19
beaver	6	**10**	9	7
bikes	20	20	20	**23**
boat	18	23	25	**30**
camel	13	15	14	**19**
car	26	34	38	**38**
dolphin	17	20	**23**	20
elephant	12	15	16	**17**
giraffe	18	**21**	20	21
guitar	26	26	23	**27**
horse	14	15	15	**17**
kangaroo	8	8	**9**	7
motorbikes	13	16	18	**19**
penguin	8	10	**11**	10
shark	**20**	19	18	19
tiger	11	17	**19**	19
wristwatch	18	21	23	**32**
zebra	25	44	50	**56**
all	16	20	21	**22**

Table 3: Average Precision(%) on the database with images from the Web. Initialization with 1 image, 10 feedbacks, 5 labels per feedback. Except for the last column, results are shown for global dictionaries. The two numbers in column heads (for instance '128/512') are the sizes of the color and textures dictionaries. The last column shows results using the method proposed in this paper, where dictionaries are built during the retrieval.

Approach	Unsupervised		Supervised	Supervised+Unsupervised	
Features	lab/qw(a)	VLAD(b)	lab/qw(c)	lab/qw+vlad(d)	GMMs(e)
bicycle	90	90	91	95	94
bus	95	95	96	97	98
car	93	94	95	96	97
cat	80	89	87	91	93
cow	82	92	92	93	94
dog	73	81	82	87	87
horse	84	87	86	92	92
motorbike	93	92	95	97	96
person	77	81	81	87	86
sheep	88	92	90	94	95
all	85	89	89	93	93

Table 4: Area under ROC Curve(%) on the test set of VOC2006 database. lab = $L^\star a^\star b^\star$ colors, qw = quaternionic wavelet textures, VLAD = Vector of Locally Aggregated Descriptors with Harris-Laplace/rgSIFT.

6. EXPERIMENTS ON BATCH LEARNING

In this section, we present results for batch learning, where the goal is to build classifiers for a set of categories thanks to a large training set. In order to compute theses results, we followed the protocol named "comp1" in PASCAL Challenge for VOC 2006 [12]. Results are computed using the Matlab code provided with the database, which measures performance using the Area Under the ROC Curve (AUC). Results are displayed in Table 4 and detailed in the next sections.

6.1. Unsupervised visual dictionary

We have first carried out experiments using a global visual dictionary of colours and textures. This dictionary is created using K-means on the training set. We have tested several dictionary sizes, and it appears that a dictionary of 256 colours and 256 textures provides the best results, displayed in column (a) of Table 4. We used a SVM classifier with $C = 1$ and a Gaussian kernel with a χ^1 distance and $\sigma = 0.5$. One classifier is trained for each category. This method gets an average AUC of 85% and a MAP of 56%.

We have also carried out experiments using the state of the art approach: Vectors of Locally Aggregated Descriptors (VLAD) [20], which is derived from the method based on bag of features and Fisher Kernels proposed in [33]. VLAD approach aggregates keypoint descriptors into a single vector in

a more efficient way than bag of features. We have integrated this aggregation scheme using keypoints detected with Harris-Laplace and described with rgSIFT [41]. We have tested several dictionary sizes (from 8 to 64), and it appears that a dictionary of 32 visual words provides the best results, displayed in the second column of Table 4. The training procedure is the same, except that we used a Gaussian kernel with L^2 distance better adapted to VLAD descriptor. This method gets an average AUC of 89% and a MAP of 66% (column (b) of Table 4).

From this first experiments, it seems that the color and texture features are not as relevant as VLAD descriptor regarding classification of VOC 2006 data. In particular, the classifier using the global visual dictionary based on VLAD significantly outperforms colour and texture visual dictionary description on dog, cat or cow categories.

6.2. Supervised visual dictionary

In order to clearly illustrates the power of our supervised dynamic dictionary approach, we used it for batch learning and compared classification results with the previous ones on global dictionaries. However, using our method to build a visual dictionary while remaining in a classic batch learning context is not really relevant. Indeed, the dictionary is traditionally built on all the features in the images labelled as positive, which would make our incremental (dynamic) learning process useless. Furthermore, the evaluation of the kernel target alignment requires $O(n^2)$ operations, with n the number of labels. In the context of batch learning, where training sets usually contain thousands of labels, these evaluations become intractable.

In order to circumvent these issues, we use our incremental method simulating retrieval sessions on the training sets, thus we build the visual dictionary by creating new features in an interactive retrieval framework. The idea is the following: if we store the classifier created at the end of a retrieval session, we get a good function to evaluate the relevance of any image in the neighborhood of the initial query image. Then, if we repeat such a process for each image of a category, we can represent any image with a "semantic descriptor" made of all the outputs from the classifiers of simulated retrieval sessions. This semantic descriptor based on supervised features relies on a category or a concept. A detailed presentation of this process is given in Algorithm 2. At last, we train a SVM classifier on the semantic descriptor corresponding to the category to retrieve.

Results with this approach are displayed on the column (c) of Table 4. This method achieves an average AUC of 89% and a MAP of 66%. If we compare these results to the one of the global dictionary of colours and textures, we can see the improvement of the learning process we propose, since both methods use the same low-level features. Our new process is at least as good as a global dictionary based on VLAD descriptor and even outperforms this state of the art method on the critical dog, cat and cow categories.

6.3. Combined supervised and unsupervised visual dictionaries

Finally, in order to compare somewhat fairly our approach to the current best method on PASCAL Challenge [33], we have carried out experiments using the concatenation of all our supervised features and the global dictionary based on VLAD descriptors. This approach is an approximation of the original method from Perronnin et al. [33] who creates a large vector for each image: one part being related to a global ("universal" in [33]) visual dictionary based on an unsupervised GMM trained on all data, and the other part is related to a supervised (category) visual dictionary using Gaussian Mixture Models trained on each category. Inspired by these works, [20] Jegou et al. proposed a simpler approach in which K-Means algorithm applied to VLAD descriptors approximates the universal visual dictionary and gets performance comparable to Perronnin et al. ones. We then concatenate the global VLAD vectors to our supervised "semantic" descriptors. In order to compare Perronnin et al. approach with our method, we have reported the results of [33] in the column (e) of Table 4.

We have tuned the SVM and Gaussian kernel parameters following a two-folds cross validation between train and validation sets. Results are shown in the column (d) of Table 4. This method gets an average AUC of 93% and a MAP of 74%.

Performance is about the same, and thus illustrates the ability of our method to get very good results with less sophisticated image representations.

Algorithm 2 Supervised feature creation for one category using an interactive retrieval technique.

Given: category C, training set A, number of feedbacks F, number of labels per feedback s

For each image c of $C \cap A$ do

- Create a new retrieval session with no labels
- Initialization:
  - Label c as positive
  - $I_0^\star = s$ closest images to c in A
  - Label images of $I_0^\star \cap C$ as positive, otherwise as negative
- For f from 1 to F do
  - Train a new classifier f_c using current labels
  - $I_f^\star = s$ images selected in A using active learning
  - Label images of $I_f^\star \cap C$ as positive, otherwise as negative

Output: Supervised feature vectors for images $(I_i)_i$: $S_i^C = \Big(f_c(I_i)\Big)_{c \in C}$

7. CONCLUSION

In this paper, we introduced a framework to learn a kernel function during an interactive retrieval. This framework does not need any assumptions on the features, since the main requirement is an evaluation function which does not have to satisfy any special mathematical property. Furthermore, this framework allows the combination of features of different types, through evaluation functions adapted to each kind of feature (*cf.* Eq. (13)).

One application of this framework is the ability to build an optimal dictionary during the retrieval, instead of the tiresome step of cross validation, usually performed to build a global dictionary (hopefully the best one). Moreover, we showed on VOC 2006 database, that this automatically built dictionary has better performances than the best global dictionary, while still being compatible with an on-line use.

Another interest of our framework is that it dynamically builds the kernel function as new words are added to the dictionary. We also introduced an efficient active learning technique for this framework, based on the kernel target alignment.

This framework can be used with more complex features, for instance with keypoints or bags of features or even graphs, as long as an evaluation function can be defined. We are also working on extensions of this framework for long-term or collaborative learning, where the information given by past users is used to optimize the whole retrieval system.

References

[1] F. Bach. High-dimensional non-linear variable selection through hierarchical kernel learning. Technical report HAL 00413473, September 2009.

[2] L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems*, volume 20, pages 161–168, 2008.

[3] K. Brinker. Incorporating diversity in active learning with support vector machines. In *International Conference on Machine Learning*, pages 59–66, February 2003.

[4] W. L. Chan, H. Choi, and R. Baraniuk. Quaternion wavelets for image analysis and processing. In *IEEE International Conference on Image Processing (ICIP)*, volume 5, pages 3057–3060, October 2004.

[5] Y. Chen and J.Z. Wang. Image categorization by learning and reasoning with regions. *International Journal on Machine Learning Research*, 5:913–939, 2004.

[6] J. Cheng and K. Wang. Active learning for image retrieval with co-SVM. *Pattern Recognition*, 40:330–334, 2007.

[7] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, pages 15(2):201–221, May 1994.

[8] B. Collins, J. Deng, K. Li, and L. Fei-Fei. Towards scalable dataset construction: An active learning approach. In *European Conference on Computer Vision (ECCV)*, 2008.

[9] K. Crammer, J. Keshet, and Y. Singer. Kernel design using boosting. In *In Advances in Neural Information Processing Systems 15*, pages 537–544. MIT Press, 2003.

[10] N. Cristianini, J. Shawe-Taylor, A. Elisseff, and J. Kandola. On kernel target alignement. In *Neural Information Processing Systems*, Vancouver, Canada, December 2001.

[11] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *International Conference on Machine Learning (ICML)*, volume 227, Corvalis, Oregon, 2007.

[12] M. Everingham, A. Zisserman, C. K. I. Williams, and L. Van Gool. The PASCAL Visual Object Classes Challenge 2006 (VOC2006) Results. http://www.pascal-network.org/challenges/VOC/voc2006/results.pdf.

[13] P.H. Gosselin and M. Cord. Active learning methods for interactive image retrieval. *IEEE Transactions on Image Processing*, 17(7):1200–1211, 2008.

[14] P.H. Gosselin, M. Cord, and S. Philipp-Foliguet. Combining visual dictionary, kernel-based similarity and learning strategy for image category retrieval. *CVIU, special issue on Similarity Matching in Computer Vision and Multimedia*, 110(3):403–417, 2008.

[15] K. Grauman and T. Darell. The pyramid match kernel : Discriminative classification with sets of image features. In *IEEE International Conference on Computer Vision (ICCV)*, Beijing, China, October 2005.

[16] Y. Guo and R. Greiner. Optimistic active learning using mutual information. In *Joint Conference on Artificial Intelligence*, pages 823–829, Hyderabad, India, 2007.

[17] T. Hertz, A. Bar-Hillel, and D. Weinshall. Learning distance functions for image retrieval. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 570–577, Washington, DC, June 2004.

[18] S. C.H. Hoi, R. Jin, J. Zhu, and M. R. Lyu. Semi-supervised SVM batch mode active learning with applications to image retrieval. *ACM Transactions on Information Systems (TOIS)*, 27(3):1–29, July 2009.

[19] P. Jain and A. Kapoor. Active learning for large multi-class problems. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 762–769, 2009.

[20] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *IEEE Conference on Computer Vision & Pattern Recognition*, June 2010.

[21] T. Joutou and K. Yanai. A food image recognition system with multiple kernel learning. In *International Conference on Image Processing (ICIP)*, Cairo, Egypt, September 2009.

[22] G. R. G. Lanckriet, N. Cristianini, N. Bartlett, L. El Ghaoui, and M. I. Jordan. Learning the kernel matrix with semi-definite programming. In *International Conference on Machine Learning*, Sydney, Australia, 2002.

[23] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems*, volume 19, pages 801–808, 2007.

[24] J.E. Lee, R. Jin, and A.K. Jain. Rank-based distance metric learning: an application to image retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2009.

[25] X. Li, L. Wang, and E. Sung. Improving adaboost for classification on small training sample sets with active learning. In *The Sixth Asian Conference on Computer Vision (ACCV)*, Korea, 2004.

[26] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *International Conference on Machine Learning (ICML)*, 2009.

[27] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Supervised dictionary learning. In *Advances in Neural Information Processing Systems*, volume 21, pages 1033–1040, 2008.

[28] F. Moosmann, E. Nowak, and F. Jurie. Randomized clustering forests for image classification. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 30, september 2008.

[29] A. Nasser, D. Hamad, J.-L. Rouas, and S. Ambellouis. The use of kernel methods for audio events detection. In *International Conference on Information and Communication Technologies: From Theory to Applications, 2008.*, pages 1–6, April 2008.

[30] NIST. TREC Video Retrieval Evaluation Campain. http://www-nlpir.nist.gov/projects/trecvid/.

[31] G. Patanè and M. Russo. The enhanced LBG algorithm. *Neural Networks*, 14(9):1219–1237, November 2001.

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

[33] F. Perronnin and C. R. Dance. Fisher kernels on visual vocabularies for image categorization. In *IEEE Conference on Computer Vision & Pattern Recognition*, June 2007.

[34] F. Perronnin, Y. Liu, J. Sanchez, and H. Poirier. Large-scale image retrieval with compressed fisher vectors. In *IEEE Conference on Computer Vision & Pattern Recognition*, June 2010.

[35] M. Protter and M. Elad. Image sequence denoising via sparse and redundant representations. *IEEE Transactions on Image Processing*, 18:27–36, 2009.

[36] N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *International Conference on Machine Learning*, 2001.

[37] H. Sahbi. Kernel PCA for similarity invariant shape recognition. *Journal of Neurocomputing*, 70:3034–3045, november 2006.

[38] R. E. Schapire. The strength of weak learnability. In *Machine Learning*, volume 5(2), pages 197–227, 1990.

[39] F. Schroff, A. Criminisi, and A. Zisserman. Harvesting Image Databases from the Web. In *Proceedings of the 11th International Conference on Computer Vision, Rio de Janeiro, Brazil*, 2007.

[40] S. Tong and D. Koller. Support vector machine active learning with application to text classification. *Journal of Machine Learning Research*, pages 2:45–66, November 2001.

[41] K. E. A. van de Sande, T. Gevers, and C. G. M. Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

[42] S. Vijayanarasimhan and K. Grauman. What's it going to cost you?: Predicting effort vs. informativeness for multi-label image annotations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2270–2277, 2009.

[43] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *Neural Information Processing Systems*, Vancouver, Bristish Columbia, December 2002.

[44] D. Zhang, F. Wang, Z. Shi, and C. Zhang. Interactive localized content based image retrieval with multiple-instance active learning. *Pattern Recognition*, 48(2):478–484, 2010.