EFFICIENT BAG-OF-FEATURE KERNEL REPRESENTATION FOR IMAGE SIMILARITY SEARCH

F. Precioso\(^1\), M. Cord\(^1\), D. Goris\(s\)\(^\ddagger\), N. Thome\(^1\)

\(^1\)LIP6, UMR CNRS 7606, UPMC - Sorbonne Universités
4 place Jussieu, 75005 Paris, France, lastname@lip6.fr
\(^\ddagger\)Yakaz
34 rue de clery, 75002 Paris, Paris, France, david@yakaz.com

ABSTRACT

Although “Bag-of-Features” image models have shown very good potential for object matching and image retrieval, such a complex data representation requires computationally expensive similarity measure evaluation. In this paper, we propose a framework unifying dictionary-based and kernel-based similarity functions that highlights the tradeoff between powerful data representation and efficient similarity computation. On the basis of this formalism, we propose a new kernel-based similarity approach for Bag-of-Feature descriptions. We introduce a method for fast similarity search in large image databases. The conducted experiments prove that our approach is very competitive among State-of-the-art methods for similarity retrieval tasks.

Index Terms— Bag-of-Features, similarity, kernels, Bag-of-Words, visual dictionary, image retrieval

1. INTRODUCTION

Complex descriptions representing each image by one set of vectors, called Bag-of-Features (BoF), showed to be powerful for image similarity search in image and video databases [1, 2]. Designing powerful similarities between images and ranking schemes on unordered sets of local features is a very challenging task.

As an alternative to standard voting scheme [1], Bag-of-Words models (BoW) based on visual dictionary computation have been developed both to overcome the amount of data to be processed [3]. Recently, alternatives very successful in categorization tasks, based on a generative coding of BoW, have been considered [4, 5].

Another popular alternative to the voting scheme that provides a similarity function on BoF are kernel on bags. Grauman and Darrell [6] proposed the pyramid matching kernel to compute the similarity between sets of PoIs. Their scheme provides good results, but is restricted to kernels which can be explicitly formulated as a dot product in an induced space. Other approaches attempting to go beyond explicit feature mappings belong to the class of kernels on sets of vectors, derived from kernels on vectors [7, 8]. These kernels have been successfully adapted for object retrieval [9, 10, 11]. The major advantage of such a framework is that it becomes possible to use unspecified kernel functions. However, the similarity computation rapidly becomes intractable when the size of the database increases.

In this paper, we propose a new approach based on kernel on bags, combining accuracy of kernel-based similarities and efficiency of dictionary-based approaches. We validate the relevance of the proposed kernel on two image datasets for semantical similarity search and near duplicate detection. Our formalism encompasses in a unified framework dictionary-based and kernel on set-based similarity functions. We evaluate the performances of our system both in terms of retrieval accuracy and computational efficiency, and compare our new kernel with State-of-the-art techniques (BoW approaches, kernel on bags, and super-vector coding [5]).

2. BOF SIMILARITY FOR IMAGE SEARCH

Let any image \(I\) be represented by a feature bag \(B\) composed of \(s\) unordered feature vectors \(b\) \(\in \mathbb{R}^d\): \(B = \{b\}_s\). Let \(\mathcal{F}\) be the database of images and \(\mathcal{F}\) the database of feature vectors \(b\). Let \(q\) be the query image represented by the bag \(B_q = \{b\}_q\).

In content-based image retrieval, if \(\text{score}_{ij} = \text{score}(B_i, B_q)\) denotes the similarity score between the query image \(I_q\) and the image \(I_j\), the search-by-similarity process in \(\mathcal{F}\) consists of comparing \(\text{score}_{ij}\) and \(\text{score}_{ip}\) for all \(i, p \in \{1, \ldots, n\}\) by increasing the gap between high and low matches:

\[
\text{score}_{ij}(B_q) = \Phi(B_q), \Phi(B_j) >
\]

This class of kernels has been later extended [9, 10] by raising the minor kernel \(k\) to the power of \(q\) in order to increase the gap between high and low matches:

\[
K_{\text{power}}(B_j, B_q) = \sum_{b_j \in B_j} \sum_{b_q \in B_q} \phi(b_j, b_q)
\]

However, computing this score between the query image \(I_q\) and all the images from the database \(\mathcal{F}\) is clearly computationally expensive if one thinks to the amount of combinations between all local descriptors of \(I_q\) and all local descriptors of all other images.
3. OUR ULTRA FAST KERNEL SCHEME

If some strategies have been developed to build kernels on bags using explicit mapping $\Phi$ [6, 12], we propose here a kernel on bags framework, called $UFast_K$, compliant with implicit mappings while remaining computationally tractable.

More precisely, our kernel, generalizing eq.(1), is defined by:

$$UFast_K(B_j, B_q) = \sum_{b_{r,q} \in B_q} \sum_{b_{r,j} \in B_j} k(b_{r,q}, b_{r,j}) \Phi(b_{r,q}, b_{r,j})$$

where $\Phi$ is a function associated to a quantifier $Q$ of the local descriptor space. By this way, many irrelevant matches in eq.(4) may be discarded, so that the whole similarity get closer to the exact match kernel between sets of points. This scheme exploits the fact that a large number of $\Phi$ values are zero over the dataset $\mathcal{F}$ to optimize the computation of the similarity $UFast_K$ using indexing structures.

We use here the strategy developed for BoW: the quantifier $Q$ maps a local descriptor $x \in \mathbb{R}^d$ to the integer index $m \in \llbracket 1, |W| \rrbracket$ of the closest codeword $w_m$, among the codebook $W$, to the descriptor $x$. $\Phi$ is defined as the indicator function:

$$\Phi(x, y) = I_{Q(x)=Q(y)}$$

The algorithm 1 details the steps of our similarity search over $\mathcal{F}$.

Algorithm 1

1. Database image similarity $score_j$ are initialized to 0.
2. for all query image feature $b_{r,q}$ do
3. a fast $\Phi$ search on all features of the dataset $\mathcal{F}$ quantized in the same cluster as $b_{r,q}$
4. for all $I_j$ such that a feature $b_{r,j}$ has been found do
5. increase the similarity by $score_j := score_j + k(b_{r,j}, b_{r,q})$
6. end for
7. end for

As one can see in step 3 of the algorithm, fast strategies may be implemented to boost the search using indexing structure. Inverted files strategies may be used with classical clustering techniques like K-means if the data are sparse [13]. Among all techniques, we have focused in our work on the powerful Locality-Sensitive Hashing scheme (LSH) which achieves this search step with a computational complexity sublinear with the amount of local descriptors in $\mathcal{F}$. A cluster (or a bucket) of LSH is hence defined as the intersection of $L$ hash tables, each one concatenating $H$ hash functions.

The breakdown in complexity of $UFast_K$ compared to exhaustive kernels on bags lies in the restriction of all combinations of minor kernel computations inside the $M$ clusters (or buckets in LSH) which can be emphasized by rewriting $UFast_K$:

$$UFast_K(B_j, B_q) = \sum_{m=1}^{M} \kappa(min_m(B_j), min_m(B_q))$$

where $min_m(B_j) = \{b_{r,j} \in B_j | Q(b_{r,j}) = m\}$.

Another interest of our $UFast_K$ is to reduce the amount of noise by taking into account only neighbors of each query local descriptor while the exhaustive kernel on bags accumulates small similarity values corresponding to very dissimilar PoIs between compared BoFs. This is illustrated on Fig.(1). Indeed, for the class cars, almost all objects have the same pose and they cover most of the image, thus most local descriptors are semantically relevant. In such a case, similarity search results are equivalent for exhaustive kernel on bags, Fig.1(a), and for our Ultra Fast kernel scheme, Fig.1(b). Our scheme provides thus highly reduced computational complexity for no accuracy loss. If the class of objects contains more appearance variations, as for bus, our Ultra Fast kernel provides even better results than exhaustive kernel, comparing results of Fig.1(d) with Fig.1(c)), because the amount of semantically useless local descriptors is reduced by the clustering defining $min_m$ representation of BoF.

In our framework, any minor kernels $k$ in eq.(4) may be used. We focus on corresponding $\kappa$ functions (eq.(6)) in the next section.

![Fig. 1. VOC2006: 5 300 images, 260 PoI / image, 1.4 Million of color SIFT (384 dimensions)](image)
4. CHOICES FOR $\kappa$

We introduce in our formalism different choices for $\kappa$ leading to different classical BoF similarity measures. We thus encompass in one scheme all BoF similarities, including our Ultra Fast scheme, highlighting computational complexity and accuracy differences.

4.1. linear kernel

The standard similarity measure between image $I_q$ and query image $I_q$, when they are represented by Bow vectors, is the inner product of these vectors. By defning the kernel on mini Bags:

$$\kappa(mini_m(B_q), mini_m(B_q)) = \|mini_m(B_q)\| \cdot \|mini_m(B_q)\|$$

(7)

in our formalism (eq.(6)), $U\text{Fast}_K$ corresponds to the standard inner product between BoW vectors.

4.2. Shifted-linear kernel

In order to refne the similarity between BoWs and not anymore counting only local descriptors falling into the same cluster, one can account for the position in the feature space of each local descriptor with respect to the center of its assigned cluster:

$$\kappa(mini_m(B_q), mini_m(B_q)) = \sum_{b_{r,q} \in B_q} \sum_{b_{s,j} \in B_j} k^\phi(b_{r,q}, b_{s,j}) f_{LSH}(b_{r,q}, b_{s,j})$$

(8)

We show in appendix A that this representation corresponds to the Vector of Locally Aggregated Descriptors (VLAD) proposed by Jegou et al. [5].

4.3. $\chi^2$-gaussian kernel

We propose in this paper to preserve all the accuracy of considering a $\chi^2$-gaussian kernel as minor kernel $\kappa$:

$$\kappa(mini_m(B_q), mini_m(B_q)) = \sum_{b_{r,q} \in B_q} \sum_{b_{s,j} \in B_j} k^\phi(b_{r,q}, b_{s,j}) f_{LSH}(b_{r,q}, b_{s,j})$$

(9)

Based on the properties of LSH, we can easily prove that our $U\text{Fast}_K$, with the above $\kappa$ function, is indeed a Mercer kernel by demonstrating the positive defnite property of its Gram matrix $\kappa_{ij} := \langle k^\phi(b_{r,i}, b_{s,j}) f_{LSH}(b_{r,i}, b_{s,j}) \rangle_{ij}$.

The computational complexity of kernel on bags given in eq.(1) is $O(n^2)$ with $n$ the number of local descriptors in each image while the complexity of our Ultra Fast Scheme is $O(n \cdot M) = O(n \cdot \sqrt{M})$ with $M$ the number of clusters. Classically $M$ and $n$ are of the same magnitude order, leading to a complexity of approximately $O(n)$.  

5. EXPERIMENTS

We perform search-by-similarity on the VOC2006 image database [14] and on a 100K image dataset (INRIA Holidays+Flickr [5]). In all experiments, the minor kernel between local descriptors is the $\chi^2$-gaussian kernel at power $q = 5$. Mean Average Precision (MAP) is used to compare retrieval performances. We consider performances computed by considering as relevant, any image in the same class as the current $I_q$. This approach allows to focus on semantical similarity aspects: the similarity between any pair of images from the same class should be high, and low otherwise, and the VOC2006 classes are defned by highly semantic concepts as bottle, bus, cars ... so that each class always contains very different objects.

Bags of SIFT are extracted for all images. For $U\text{Fast}_K$, we use the standard implementation $E^2\text{LSH}$ from [15] ($R = 225$, $L = 100$, $H = 20$). We compare $U\text{Fast}_K$ with $K_{\text{power}}$ (eq.(3)) and the Bag-of-Words method with the standard parameter settings [16, 17]: database SIFT vector $K$-means clustering, tf-idf weighting scheme, and Euclidean distance.

MAP and time results (expressed in terms of ratio for kernel approaches) are reported on Tab.1 with 4 dictionary sizes: 100, 200, 300 and 1000 for BoW. First, let say that the MAP scores are not very high because the similarity search is evaluated here without any learning. Far away from near-duplicate or copy-detection, the evaluation here really focuses on highly semantical similarity purpose. It highlights the ability of the methods to get high similarity scores for categories of objects and not only for the same object taken under various points of view.

<table>
<thead>
<tr>
<th>Method</th>
<th>MAP %</th>
</tr>
</thead>
<tbody>
<tr>
<td>BoW</td>
<td></td>
</tr>
<tr>
<td>10K words</td>
<td>41.4</td>
</tr>
<tr>
<td>20K words</td>
<td>44.6</td>
</tr>
<tr>
<td>200K words</td>
<td>54.9</td>
</tr>
<tr>
<td>VLAD (from [5])</td>
<td></td>
</tr>
<tr>
<td>16 words, D=2048</td>
<td>49.6</td>
</tr>
<tr>
<td>64 words, D=8192</td>
<td>52.6</td>
</tr>
<tr>
<td>$U\text{Fast}_K$</td>
<td>76</td>
</tr>
</tbody>
</table>

Table 2. Comparison of the accuracy of $U\text{Fast}_K$ with State-of-the-art approaches on INRIA Holidays dataset (1491 images) authors in [18, 5] on the 1491 image of INRIA Holidays dataset with the same experimental protocol. These results show that our method is more accurate than BoW approaches which quantize local features but consider only hard-assignment and even more accurate than VLAD which integrates a normalized linear kernel on shifted vectors.

5.2. Performances on 100K dataset (INRIA Holidays+Flickr)

We now compare $U\text{Fast}_K$ with State-of-the-art approaches for image retrieval task on large databases. These databases contains 1491 images for INRIA Holidays, at which are added 100,000 “distractors” images from Flickr to reach a 100K dataset [5]. We have used the experimental setup of [5] and the local features publicly available from authors’ webpage. The comparison here is more near duplicate detection oriented. Tab.2 shows State-of-the-art results provided by
Tab. 3 presents the results for experiments on the 100K (INRIA Holidays+Flickr) dataset conducted as defined by the experimental protocol in [5]. The results given for VLAD are actually extracted from the curves in [5] and based on details provided by the authors. We can thus see that for about the same time of computation we have a relative gain of 10% in MAP. These results show that the non-linear mapping used in UFastKL is more accurate than the linear one used in original VLAD for that database for comparable running times.

<table>
<thead>
<tr>
<th>Method</th>
<th>MAP</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VLAD (from [5])</td>
<td>64</td>
<td>39</td>
</tr>
<tr>
<td>UFastKL</td>
<td>43</td>
<td>12</td>
</tr>
</tbody>
</table>

Table 3. Performances on 100K dataset of UFastKL vs VLAD

6. CONCLUSION

In this paper, we focus on designing a kernel-based similarity approach to find a trade-off between powerful data representation and efficient similarity scheme. We propose a new kernel on Bag-of-Features, the Ultra Fast kernel, based on the kernels on sets of vectors derived from kernels on vectors. Our framework encompasses dictionary-based and kernel on set-based similarity functions, including recent Fisher representation simplification. We have evaluated our method in a semantic similarity search context and for a near copy detection task showing the good tradeoff between accuracy of exhaustive kernel on Bags and efficiency of Bow representation, our kernel provides. The designed kernel fulfilling Mercer’s conditions, we have hence integrated it in a SVM framework to evaluate its performances in an active learning context [19] and we are currently working on evaluating it for large scale classification task.

A. VLAD IS A LINEAR-SHIFTED KERNEL ON BAGS

Starting from eq.(8), we can write:

\[ \kappa_{\text{mini}}(B_q, B_j) = \sum_{b_q \in B_q} \sum_{b_j \in B_j} \langle b_q - w_m, b_j - w_m \rangle \]

\[ = \sum_{b_q \in B_q} \sum_{b_j \in B_j} \langle b_q, b_j \rangle - \sum_{b_q \in B_q} \langle b_q, w_m \rangle - \sum_{b_j \in B_j} \langle b_j, w_m \rangle + \sum_{b_q \in B_q} \sum_{b_j \in B_j} \langle b_j, b_q \rangle \]

\[ = \langle \phi_m(b_q), \phi_m(b_j) \rangle \]

with \( \phi_m \) a mapping function from \( \mathbb{R}^d \) to \( \mathbb{R}^{d \times M} \) such that:

\[ \phi_m(x) = x \otimes U_m - w_m \otimes U_m \]

(11)

where \( U_m \), a vector of \( M-1 \) zeros and 1 at the \( m^{th} \) component, \( m = Q(x) \) as defned for eq.(5) and \( w_m \) is the codeword assigned the vector \( x \).

With the notations of [5], \( i^{th} \) component of the VLAD descriptor associated to \( m^{th} \) cluster is defned by:

\[ v_{m,i} = \sum_{x \in B_q : Q(x) = m} x_i - w_{m,i} = \sum_{x \in B_q : Q(x) = m} \phi_m(x) \]

(12)

where \( x_i \) and \( w_{m,i} \) respectively denote the \( i^{th} \) component of the local descriptor \( x \) and its assigned visual word \( w_m \). We can finally rewrite our \( \kappa \) as:

\[ \kappa_{\text{mini}}(B_q, B_j) = \langle v_{m,q}, v_{m,i} \rangle \]

(13)

In [5], the VLAD vector \( v_i \) is further \( L^2 \)-normalized which leads, instead of considering \( UFastKL \) with the above \( \kappa \), to consider the normalized kernel:

\[ UFastKL(B_q, B_j) = \frac{UFastKL(B_q, B_j)}{\sqrt{UFastKL(B_q, B_q)UFastKL(B_j, B_j)}} \]

B. REFERENCES


