UNIVERSITÀ DI PARMA

UNIVERSITÀ DEGLI STUDI DI PARMA

Dottorato di Ricerca in Tecnologie dell'Informazione

XXXII Ciclo

Content-Based Image Retrieval for Visual Big Data Analysis

Coordinatore:

Chiar.mo Prof. Marco Locatelli

Tutor:

Chiar.mo Prof. Andrea Prati

Dottorando: *Federico Magliani*

Anni 2016 - 2019

Alla mia famiglia

Summary

List of Figures

List of Tables

Introduction

The Content-Based Image Retrieval [1] (CBIR) problem is a subtask of computer vision related to the classical information retrieval task. Moreover, with the growth of the available digital images on the Internet, thanks to cheaper and efficient digital cameras and smartphones, have become more accessible encouraging the study of this particular task more than before. The access to this huge quantity of data has allowed the creation of very large datasets, that brought with them lots of new challenges (for example: how to increase the quality of retrieval in dark or night scenarios, how to reduce the memory occupancy and in the end how to retrieve fastly the correct results on big datasets).

In a nutshell the objective of the task is to correctly retrieve and rank the similar images to the query one. In order to validate the results, a groundtruth with the similar images to retrieve is given. Of course, if the correct images will not be retrieved in the first ranking positions, so the final retrieval accuracy will be lower than in the perfect case (all the correct images retrieved in the first positions). The ranking of the images is possible due to the calculation of the distance between the query image and all the other images. Before to find the similar images and calculate the correct rank, it is mandatory to define a feature vector of each image. The feature vector allows to describe the image information in a unique global vector.

The problem is trivial for humans that simply execute this task through experience and semantic perception, but it is not so easy for a computer. This is known as semantic gap [2], which refers to the difference (or gap) between low-level image pixels and high-level semantic concepts. Thanks to the techniques for the creation of

the global feature vector that will be explained in the next sections, the computers are able to evaluate the similarity value between two images. The objectives of the different algorithms evaluated are several, where usually the most important is the retrieval accuracy (aka the number of images correctly identified) that needs to be as high as possible (in the perfect case equals to 1). On the other hand, the secondary targets are execution, retrieval time and memory occupancy that need to be as low as possible. The first one is relative to the time utilized for the creation of the descriptors, while the second one is the time necessary to search for similar images and the third one is related to the memory used during the process of retrieval: construction of all the descriptors and retrieval phase (calculation of the similarity value the descriptors and sorting of the ranking list).

Furthermore, the images may contain noisy patches (e.g. trees, person, cars, ...), be taken with different lightning conditions, viewpoints and resolution. In order to solve this problem it is crucial to develop algorithms and techniques with the objective of reducing the weight of the unnecessary patches of the images and that work well with a vast quantity of data. The choice of the feature detector and the feature aggregation methods is very important to start achieving the previous cited objective.

Mobile devices will play an important role in the future of CBIR systems. Mobile CBIR [3] needs to study and implement techniques for solving issues that are connected with the limited amount of hardware and memory on mobile devices. For example, storing all the images and the related descriptors for the retrieval can be problematic. In fact, a mobile device does not have the possibility to store millions of high resolution images. Usually, the solution is a client-server communication, in which the user takes a picture of a building/place and the smartphone calculates the descriptors and sends them to a server that will answer the correct name of the building and provide extra information. There are several applications of mobile CBIR systems: libraries and museum applications (which the final retrieved information can be implemented in an augmented reality scenario), fashion application for the search of certain clothes, advanced electronic tourist guides.

After introducing a simple pipeline for the resolution of the CBIR problem in Section 1.1 and presenting the basics for feature extraction and description in Section

1.2, the classical approaches for image description in the retrieval task are briefly introduced in Section 2, while Section 3 presents the comparison with the recent techniques based on deep learning. The typical problems of large-scale retrieval are depicted in Section 4. Furthermore, Section 5 introduces the graph theory and the diffusion approach with the focus on the efficient kNN graph construction. Experimental results on different datasets are then reported. Next, Section 6 describes the used image datasets, as well as the metrics used for the evaluation of the results and Section 7 reports some state-of-the-art results. Finally, conclusions are summarized at the end of the chapter.

Chapter 1

Content-Based Image Retrieval Task

I have no special talents. I am only passionately curious.

Albert Einstein

1.1 The Problem of Content-Based Image Retrieval

The proposed pipeline for the CBIR task is represented by a workflow graph in Fig. 1.1. The executed operations are:

1. At the beginning the input image is acquired. It is important to correctly import the image with the introduction of noise and to avoid to distort the image. Usually this operation is very fast and easy to do;

2. The first real step of the pipeline is the application of a feature detector and descriptor to the image, which allows to identify interesting pixels in an image. There are three different type of pixels: flat, edge and corner. An example of a flat pixel is one related to the sky in a picture. For our purposes it is not

important to detect and define it. Instead, the edge pixel represents a point on the image with some variations, but not on the edge direction. In the end, the most interesting pixel for our target task is the corner one. It identifies a point in which there are important colour variations compared to the neighbour points. There are different techniques for this step, that will be explained later in this chapter: the hand-crafted methods and the ones based on deep learning.

3. The extracted descriptors are local and need to be aggregated because there are a lot of points and it is difficult to compare all this sparse information. With a final global vector that uniquely represents the whole image, it is more easy to find a similar image in the dataset. There are several methods for different purposes (best retrieval accuracy, low retrieval time or less use of resources and available hardware). This approaches will be carefully explained in the next chapter.

4. A potential successive step is to compress the global descriptor in order to reduce the amount of data to be transmitted to the server and, therefore, the total retrieval time. However, this is a dangerous operation, because it can produce a reduction of retrieval accuracy.

5. Recently, some graph based techniques are introduced for this problem in order to increase the final retrieval accuracy for some public datasets. The most famous and used algorithm is the diffusion approach, that will be evaluated later in this thesis focusing on the problem of the efficient kNN graph creation. This step is not mandatory, but is very useful for improving the final retrieval accuracy value. However, the application this kind of methods improve also the time needed for the retrieval of the correct results. It is important to evaluate the trade-off between the retrieval accuracy and the time needed for obtaining this kind of results.

6. The last operation is the query phase. After the feature extraction and the aggregation of all the images in the dataset, for every query image, the system checks and ranks the distances between the (compressed) query descriptor and

the descriptors of the database images. The results obtained are the images ranked based on the similarity with the query image. This step is usually executed in a brute-force way, that obtains the best results, but it needs a very long time, so there are several approximate techniques for solving this problem, that can be applied on large-scale retrieval scenarios.

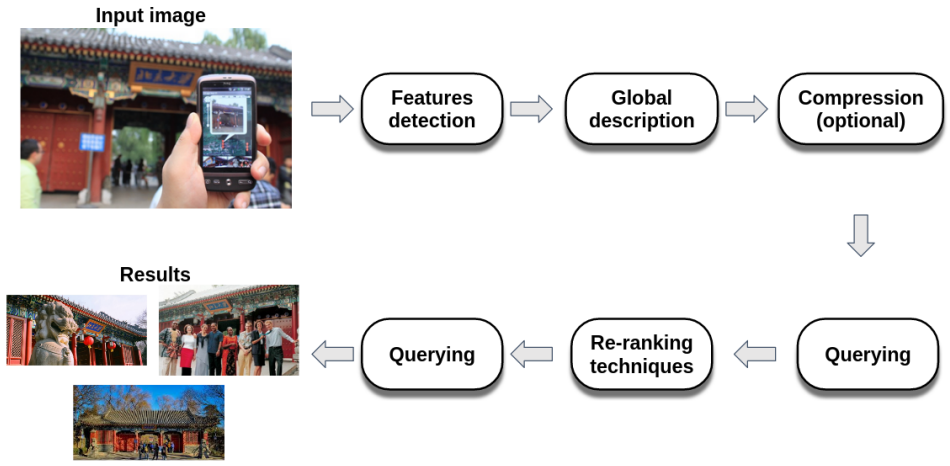

Figure 1.1: CBIR workflow: from image acquisition to the final ordered ranking list

1.2 Features Extraction and Description

As reported before, the importance of this phase is crucial for the construction of a good retrieval system. In order to extract discriminative information from an image, it is necessary to adopt a method for feature extraction and description. In the next subsections, hand-crafted and deep learning methods are presented, evaluating advantages and drawbacks. Following the past history in the literature, hand-crafted methods are presented first, followed by the more recent methods based on deep learning that demonstrate to extract different features compared to the previous methods.

1.2.1 Hand-Crafted Features

A feature is a point or a set of points (patch) in an image which contains relevant information. It is also called *local descriptor* and it contains geometric information (point coordinates and angle).

There are many algorithms that allow to solve this task. The most famous hand-crafted methods for feature extraction and description are SIFT [4] and SURF [5], actually patented, but there are several open source implementations. This means that the feature extraction pipeline is manually designed following several steps with the objective of extracting features that are invariant to translation, rotation, illumination, occlusion and scale.

SIFT (Scale Invariant Feature Transform) was proposed by David G. Lowe in 1999 [4]. It is invariant to rotation, scale, changing in lighting condition and affine transformation. Moreover, SIFT generates a large numbers of features that densely cover the image over the full range of scales and locations. Its main advantages are:

- *Locality*: features are local, therefore robust to occlusion and clutter (no prior segmentation);
- *Distinctiveness*: individual features can be matched to a large database of objects;
- *Quantity*: many features can be generated for even small objects;
- *Efficiency*: close to real-time performance;
- *Extensibility*: can easily be extended to a wide range of differing feature types, with each adding robustness.

On the other hand, SURF (Speeded Up Robust Features) was proposed by H. Bay *et al.* [5] and it is a variation of SIFT that claims to be faster and more invariant than SIFT itself. The main steps of SURF are:

1. Detection of keypoints using the determinant of the Hessian matrix;

2. Detection of the most relevant keypoints in the scale-space. It is similar to what SIFT do, but it uses also a non-maximal suppresion;

3. Detection of the orientation of the keypoints;

4. Generation of the descriptor.

This feature detector and descriptor techniques was used a lot in several computer vision tasks.

1.2.2 Deep Learning Features

More recently, deep learning techniques allowed to extract features from Convolutional Neural Networks (CNNs) that are more discriminative than the one extracted from hand-crafted methods. The deep learning approach is most used in computer vision task because the CNN-based architectures improve the results in an overwhelmed way and, thanks to modern GPU, it is possible to run CNNs in an acceptable running time. The advent of high-level software library applied on deep learning architecture as Keras [1] or PyTorch [2] has allowed to make easy the use of this kind of methods. Now, the majority of researchers in the field of artificial intelligence and computer vision use approaches based on neural networks and deep learning for the resolution of their problems obtaining excellent results.

The VGG16 architecture [6], shown in Fig. 1.2, is one of the most popular CNN-based architecture used for image classification. It is composed by:

- in input layer that will elaborate the image and resize it to the input shape accepted by the network, for every channel (R, G, B). The input shape is an important factor for a CNN architecture. Usually, by maintaining high resolution the final classification is improved, but requires more time. In the figure above the input image has a shape equals to (224,224,3);

[1] `https://keras.io/`
[2] `https://pytorch.org/`

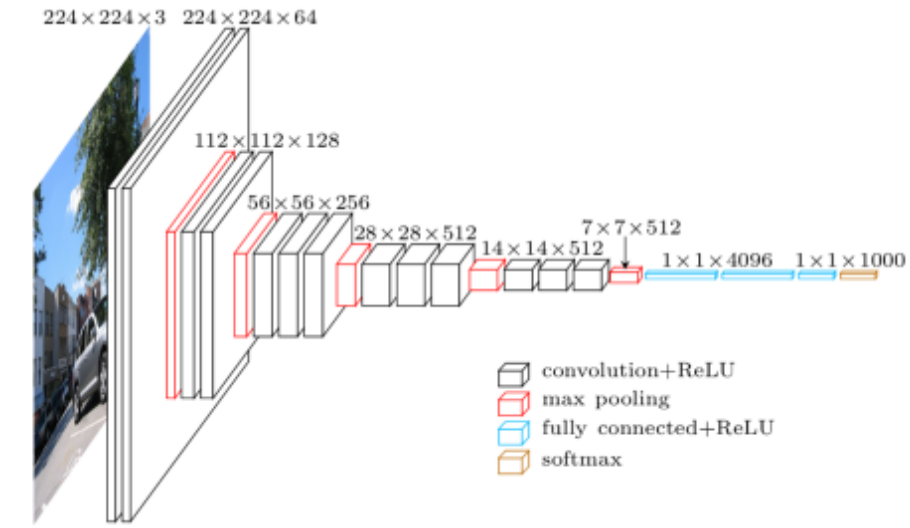

Figure 1.2: VGG16 basic architecture

- convolutional layers will apply a convolutional filter on the image. In a CNN-based architecture, there are several convolutional layers of different dimensions. The idea is to apply at the beginning of the network big convolutional filters and towards the end of the network use small ones. In the figure, the first convolutional layer produces an image with a shape of (224,224,64) and the end one produce an image with shape of (14,14,512). This process allows to train the network in order to learn at the beginning some simple patches of the image and in the end to understand very little details, that can be useful for the CBIR task;
- as an activation layer, VGG16 uses ReLU that will apply an element-wise activation function, such as the $\max(0,x)$ thresholding at zero. This leaves the size of the volume unchanged and allows to treat only positive elements;
- pooling layers will perform a downsampling operation along the spatial dimensions (width, height). There are different types of pooling, the most common used are max and average pooling;

- the last section of the network is composed by fully-connected layers and a softmax layer that will compute the class scores of the different categories. For our target this part of the network is useless and so it is removed because we are not interested to classify the image between different classes, but only to extract discriminative features for retrieval purposes.

CNNs are used in CBIR during the feature extraction phase because the features obtained from an intermediate layer of these architectures (often called *CNN codes*) are more discriminative than the features extracted by SIFT or SURF. It is possible to affirm this because the performance between the same embedding algorithm used with SIFT/SURF features and the ones obtained with the CNN codes are different and in particular the second ones are better than the first ones. An example of feature maps that it is possible to extract from the VGG16, previously showed, has a shape of (14,14,512), that represents the penultimate feature maps before the fully connected layer. Differently from the hand-crafted methods, the features extracted from CNNs are densely extracted and they can catch more details on the images through the high number of convolutional layers. Furthermore, the absence of a dense structure, required for the hand-crafted methods and the high parallelization of the process on GPUs [7] for the feature extraction step on neural networks allows to fastly extract of CNN codes than the ones obtained by the hand-crafted feature descriptors.

Thanks to the knowledge gained while solving one problem and applying it to a different, yet related, one through the process of *transfer learning*, has allowed to extract CNN codes from pre-trained networks. There are several different pre-trained CNN architectures (e.g., VGG16, GoogLeNet [8], ResNet [9], ...) that allow to easily extract features from their layers. The choice of the layers depends on the type of problem and the selected network. Obviously, the deeper the network is, the better the results obtained by the extracted features are. However, it is also true that the deeper the network, the harder the training phase is, since more hyperparameters need to be learnt, requiring more training data and time to converge.

Chapter 2

Classical Approaches: from BoW to VLAD

The question of whether a computer can think is no more interesting than the question of whether a submarine can swim.

Edsger Wybe Dijkstra

2.1 Bag of Words

Bag of Words (BoW) [10] was the first algorithm implemented using local descriptors for solving the problem of CBIR. This technique is composed by three different steps:

1. the creation of a vocabulary of visual words using a clustering algorithm applied on the database images. It is usual to apply K-means for the clustering step. It consists in the creation of some partitions of the elements in a predefined number of clusters (based on the similarity of the visual descriptors);

2. each image is described in terms of occurrence of these words (bag of words model). For each image, it is calculated the distance between the image de-

scriptor and each word of the vocabulary and then the bag of words model is updated;

3. a distance between each pair of bag of words descriptors is calculated. After that the ranking is obtained.

Figure 2.1: Example of features for Bag of Words vectors

The idea behind this technique is the use of a vocabulary, as it can be seen in Fig. 2.1, which is composed by the center feature of every cluster (centroid) obtained by the application of the K-means algorithm. Thanks to this approach it is possible to create a descriptor of an image based on the most important and discriminative features.

Though quite simple, BoW has achieved good results in CBIR, at the cost, however, of a large consumption of memory.

Several embedding techniques have been proposed to overcome the weakness of the BoW approach: Hamming Embedding [11], VLAD [12], Fisher Vector [13]. As Bag of Words models, also Hamming Embedding is based on local descriptors, but these descriptors are binarized according to the similarity and to a fixed threshold. The transformation in binary codes reduces the retrieval time and also the memory used for the storing of the descriptors of the images. Jegou *et al.* [14, 15] implemented different techniques for improving the accuracy of the Hamming Embedding. These methods obtained excellent results, but with a large usage of memory. Fisher Vector [13] represents a more complex solution of VLAD vectors. VLAD [12] descriptors are very interesting for our research and they will be explained in details in the next subsection.

2.2 VLAD and locVLAD

Vector of Locally Aggregated Descriptors (VLAD) is a simplification of the Fisher Vector and therefore it is the most used between the two. Thanks to the idea behind this algorithm that is very simple VLAD became used and famous. It allows to obtain good results also with a small descriptor and a small amount of memory.

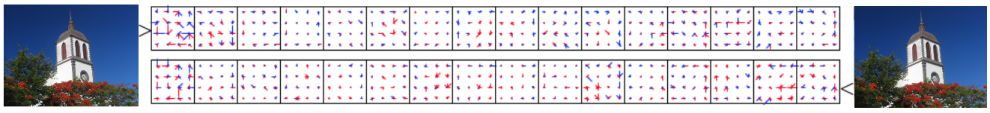

Figure 2.2: VLAD descriptors of similar images

The VLAD vector $\mathbf{v}$, represented in graphical way in Fig. 2.2, is obtained by computing the sum of residuals that is the difference between the feature $\mathbf{x}$ and the cluster center $\mu_\mathbf{i}$:

$$\mathbf{v}^i = \sum_{\forall \mathbf{x} \in X : q(\mathbf{x}) = \mu_i} \mathbf{x} - \mu_i \tag{2.1}$$

Then the concatenation of the $\mathbf{v}^i$ features is executed, resulting the unnormalized VLAD vector $\mathbf{v}$. Unfortunately, this vector can contains duplicated features that

make the recognition harder. The normalization of the vector $\mathbf{v}$ can help to solve this problem. There are many types of normalization:

- Signed Square Rooting normalization [16]: the vector is normalized using the following function $sign(x_i)\sqrt{|x_i|}$;
- Residual normalization [17]: an independent residual L_2 norm is calculated for every cluster center;
- Z-score normalization [18]: residual normalization is applied, then the mean is subtracted from each vector and each one is divided by the standard deviation. From extensive testing, it resulted that the Z-score normalization produces the best results in terms of accuracy;
- Power normalization [19]: the following function $sign(x_i)|x_i|^{\alpha}$, where $\alpha = 0.2$, is usually applied;

Finally, in order to remove noisy values all the vectors are further L_2 normalized:

$$\hat{\mathbf{v}} = \frac{\mathbf{v}}{||\mathbf{v}||} \tag{2.2}$$

After the development of VLAD descriptors, several variants of VLAD to outcome the weakness of this global descriptor and hence the final accuracy in the retrieval are proposed: CVLAD [20], CEVLAD [21], HVLAD [22], FVLAD [23], locVLAD [24] and gVLAD [18]. These methods apply modification during the aggregation phase: CVLAD and gVLAD exploit geometric information of the features, CEVLAD uses entropy of the local descriptors, HVLAD and FVLAD adopt multi vocabulary and finally locVLAD reduces noisy features, giving more importance to features located in the central part of the query images.

Now, we want to focus more on locVLAD (location-aware VLAD) algorithm because it is a contribution during my PhD work. It is a variant of VLAD that with the reduction of the influence of features found at the borders of the image wants to improve the final retrieval performance. It is not a simple removal of border features, but it is a fusion of different VLAD descriptors. The locVLAD descriptor corresponds

to the average between the VLAD vector calculated on the whole image (v) and the one executed considering the images cropped of a certain percentage ($v_{cropped}$):

$$v_{locVLAD} = \frac{v + v_{cropped}}{2}$$

It is important to note that the locVLAD is calculated only on test phase and for the database images the normal VLAD descriptors are calculated. It is quite straightforward (but important) to notice that, by repeating the whole pipeline, the feature set and, therefore, the VLAD vectors will be different.

The parameters to account are two: the average weights for the two vectors v and $v_{cropped}$, and the cropped borders percentage. Regarding the former, we performed different tests and realized that the best results are obtained by an equal weight for the two vectors, as shown in the previous equation. The value of the second parameter depends to the image resolutions. The idea behind our proposal is that the useful features for the retrieval task are usually located near the center of the image and the distractors are at the border of the image. This statement is not always true, i.e. some features at the image borders might be useful. For this reason we average both the cropped and not-cropped VLAD descriptors, creating a robust version of VLAD descriptor to noisy features. As said before, locVLAD algorithm is not applied to the database images because experiments demonstrate that applying it also on the database images decreases the final retrieval accuracy.

Chapter 3

Deep Learning Approach: Transfer Learning and Fine-Tuning

Ask not what your country can do for you; ask what you can do for your country.

John Fitzgerald Kennedy

3.1 Deep Learning for Image Retrieval

In the previous chapter we experimentally demonstrate that the feature obtained from intermediate layers of CNN-based architecture are better than the hand-crafted ones. So, here and in the next chapters we will use only this kind of features. Then, the focus will be moved to the embedding layer. In the following lines the most famous and used methods based on CNN codes will be explained.

NetVLAD [25] introduced a VLAD layer in the end of a CNN that allows to create VLAD embeddings. Ng *et al.* [26] extracted the features from intermediate layers of CNN for embedding them in VLAD descriptors. More sophisticated embedding

techniques are introduced to improve the final accuracy of the retrieval system: CCS strategy [27] combined hand-crafted features and different CNN codes, OC [28] extracted features only in some important image regions that are related to detected objects, in the end R-MAC [29] proposed a new method for the embedding phase, similar in the spirit to VLAD that reaches outstanding results in several public image datasets.

3.2 R-MAC

Considering an input image I of size W_i x H_i, the output of the activations of the convolutional layer is a 3D tensor of W_i x H_i x K dimensions, where K is the number of feature channels. The spatial resolution W_i x H_i depends on the network architecture, the layer examined, and the input shape.

The 3D tensor of responses is represented as a set of 2D features channel responses $X = \{X_i, \quad i = 1, \ldots, K\}$ where X_i is the 2D tensor representing the responses of the i^{th} feature channel over the set Ω of valid spatial locations, and $X_i(p)$ is the response at a particular position p. Therefore, the feature vector is constructed with a spatial max-pooling over all locations:

$$f_\Omega = [f_{\Omega,1}, \ldots, f_{\Omega,i}, \ldots, f_{\Omega,K}]^T \quad with \quad f_{\Omega,i} = \max(X_i(p)) \tag{3.1}$$

This representation does not encode the location of the activations due to the max-pooling operated over a single region of size W x H. For solving this problem the Regional MAC (R-MAC)[29] was introduced.

The regions are defined on the CNN response maps. Square regions at L different scales are sampled. At the first level ($l = 1$), the region size is determined to be as large as possible (its height and width are both equal to $\min(W,H)$). At every successive level l, $l \times (l+1)$ regions of width $2 \cdot \min(W,H)/(l+1)$ are uniformly samples (see Fig. 3.1).

Then, the feature vector associated with each region is calculated, and it is post-processed with L_2-normalization, PCA-whitening and further L_2-normalization. At

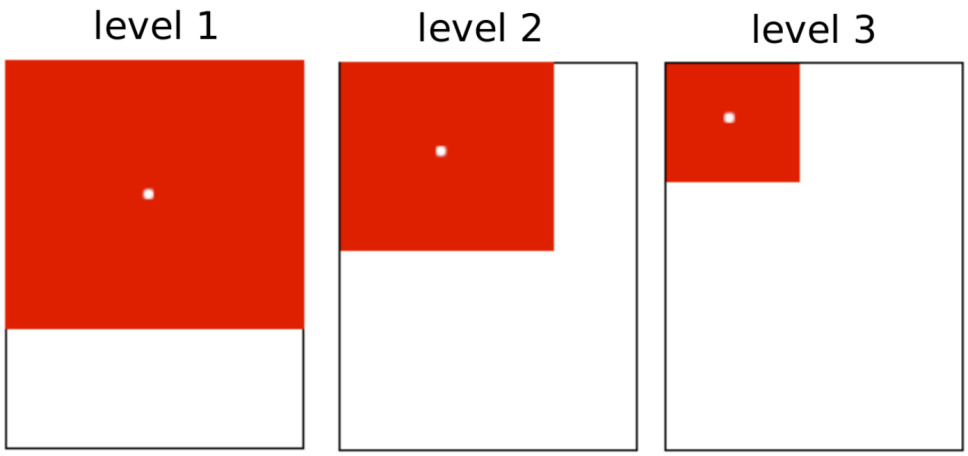

Figure 3.1: Grid detector for R-MAC descriptor

the end, the regional feature vectors are combined into a single image vector by sum-pooling and L_2-normalization.

Several new implementations of R-MAC are proposed due to the success of this feature vector:

- Gordo *et al.*[30] proposed to feed in the network three resolutions of the same image preserving the aspect ratio. Then, the three computed R-MAC descriptors are sum-pooled in a final R-MAC descriptor (M-R R-MAC that stands for Multi-Resolution R-MAC) because two or more versions of the same image with different resolutions will not produce the same output descriptor;

- Superior retrieval perfomance has been demonstrated with the use of VGG19 and ResNet101 [30] for the feature extraction phase instead of VGG16;

- Seddati *et al.* [31] proposed to execute sum-pooling before calculating R-MAC descriptors. The output feature maps of the images of three different resolutions are rescaled and summed up into one single 3D feature tensor that encodes activations at different scales;

- The application of a Region Proposal Network (RPN) [30] can enhance the detection step and reduce the impact of background on R-MAC representations;
- Gordo *et al.* [30] tried to fine-tune VGG16 and ResNet50 with different loss function on a similar big dataset to the one used for the retrieval, obtaining excellent results that are now the state of the art for several datasets.

The choice of the datasets used in fine-tuning is important because you need to learn discriminative CNN features useful for your retrieval data. The pre-trained CNN model is competent in discriminating images of different object/scene classes because it was trained on ImageNet [32] that is composed by many different classes related to several objects as cars, bikes, dogs, cats, airplanes and so on. So, it may be less effective to identify images that fall in the same class but represent different instances. In practice, very few people train a CNN from scratch (random initialisation) because it needs a lot of time and computational resources and it is rare to get enough data for an effective training. Therefore, using pre-trained network weights as initialisations or a fixed feature extractor, helps in solving most of the problems at hand. The CNN architectures for fine-tuning tasks can be subdivided in two types:

- Classification-based network: Babenko *et al.*[33] used fine-tuning to classify landmarks;
- Verification-based network: Gordo *et al.*[30] used a *Siamese network* with a triplet loss, producing better results than the application of Classification-based network, thanks to the three inputs of the network: one for a relevant example (I^+), one for a non-relevant example (I^-) and the last for the query image (I_q), as showed in Fig. 3.2.

The triplet loss of the three-stream Siamese (Verification-based) network can be defined in this way:

$$L(I_q, I^+, I^-) = \frac{1}{2} \max\left(0, m + ||q - d^+||^2 - ||q - d^-||^2\right) \tag{3.2}$$

where q is the query R-MAC descriptor, d^+ is the R-MAC descriptor of the relevant image and d^- is the R-MAC descriptor of the non-relevant example, m is a scalar

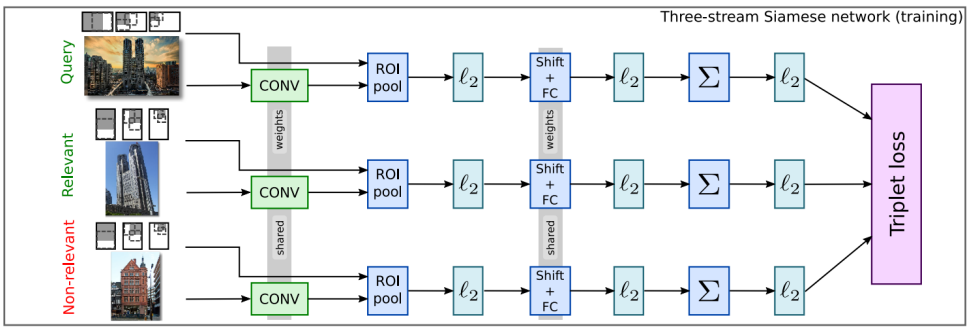

Figure 3.2: Three-stream Siamese network used for the fine-tuning process

that controls the margin. The relevant and non-relevant example are randomly picked and feed in the input of the network based on the ground truth of the dataset. The gradient is back-propagated as in a common neural network, but in this case through the three streams of the network, and the convolutional layers. The advantage of this architecture is that the back-propagation is executed also on the "PCA" layers, the shifting and the fully connected layer. This allows to the network to learn how to execute feature extraction step, PCA and shifting.

The optimizer used is SGD [34] (Stochastic Gradient Descent) with a learning rate equal to 10^{-3} and momentum equal to 5×10^{-5}. SGD is a simplification of gradient descent. Each iteration it estimates this gradient on the basis of a single randomly picked example instead of computing the gradient exactly.

This approach offers several advantages. First and foremost, it directly optimizes a ranking objective. Second, it can train the network using images at the same (high) resolution of test time. Last, learning the optimal "PCA" can be seen as a way to perform discriminative large-margin metric learning [35], in which one learns a new space where relevant images are closer.

3.2.1 R-MAC+

I proposed some improvements on the R-MAC method implemented by Gordo *et al.* [36] and I created the R-MAC+ pipeline [37].

Symbol	Definition	l=0	l=1	l=2	l=3
W	width of the feature maps	-			
H	height of the feature maps	-			
xRegions	regions along the x axis	1	1 or 2	3	2
yRegions	regions along the y axis	0	1 or 2	2	3
widthRegion	width of the region	W	$\min(W,H)$	$\lceil \frac{2}{l+1} \cdot \min(W,H) \rceil$	
heightRegion	height of the region	H	$\min(W,H)$	$\lceil \frac{2}{l+1} \cdot \min(W,H) \rceil$	
ΔW	region stride on the x axis	$W/xRegions$			
ΔH	region stride on the y axis	$H/yRegions$			

Table 3.1: Summary of notation and explanation of region detector algorithm.

First, I proposed to create R-MAC descriptors from images resized in a different way than the original multi-resolution approach. The R-MAC+ descriptors are composed by the sum-pooling of three different R-MAC descriptors obtained by: the original input image, the increased image (+25%) and the reduced image (-25%), by retaining the aspect ratio of the images. This strategy should allow to increase the dimensions of the feature maps in order to have more features and therefore local maxima than the previous multi-resolution R-MAC. This approach is connected to the new region detector, that detects a reduced number of regions (15) instead of the 20 of the original one proposed by Tolias *et al.* [29].

Second, a new faster grid mechanism for the detection of the regions is proposed because less regions are detected and then computed. The new region detector is structured on three levels, in which each grid detector has a different scale. All the symbols used and their values are explained in Table 4.1. In Fig. 3.3 are reported the different regions detected. The first level ($l = 0$) of the region detector is represented by an orange region covering all the image, that of course has the sizes of the image. Then, in the second level ($l = 1$) 2 blue square regions are used, arranged along the largest size of the image. The number of *xRegions* and *yRegions*, in the case of $l = 1$, are determined in the Equations 3.3 and 3.4.

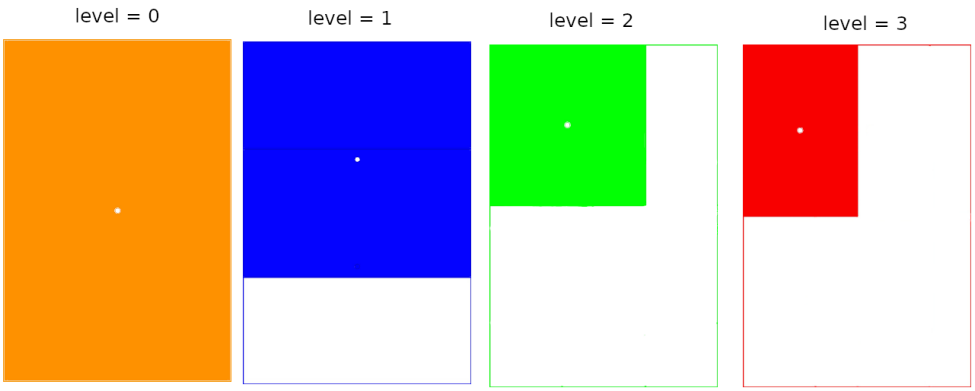

Figure 3.3: New grid detector adopted for the R-MAC+ descriptors creation

$$xRegions = \begin{cases} 1 & \text{if W < H} \\ 2 & \text{otherwise} \end{cases} \tag{3.3}$$

$$yRegions = \begin{cases} 2 & \text{if W < H} \\ 1 & \text{otherwise} \end{cases} \tag{3.4}$$

In the third step ($l = 2$), 6 green rectangulars are adopted in order to cover entirely the image. Finally, in the last step ($l = 3$), 6 red rectangulars are used. The value of *heightRegion* and *widthRegion* are equal to $\lceil \frac{2}{l+1} \cdot \min(W,H) \rceil$, but if the regions do not cover entirely the image, the width and the height of the regions change. These values are calculated following the Equations 3.5 and 3.6.

$$heightRegion = \begin{cases} \lceil H/yRegions \rceil & \text{if } (heightRegion \cdot yRegions) < H \\ heightRegion & \text{otherwise} \end{cases} \tag{3.5}$$

$$weightRegion = \begin{cases} \lceil W/xRegions \rceil & \text{if } (weightRegion \cdot xRegions) < W \\ weightRegion & \text{otherwise} \end{cases} \tag{3.6}$$

The regions are arranged based on ΔW along the horizontal axis and on ΔH along the vertical axis. This means that the image will be entirely covered because the ΔW and ΔH are related to the dimensions (W,H) and to the regions $(xRegions, yRegions)$.

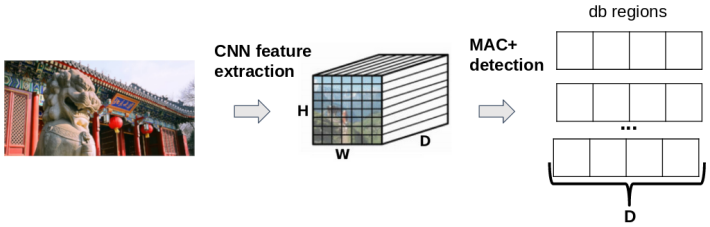

Figure 3.4: Pipeline for the creation of "db regions"

During the creation process of regional descriptors the grid detector is moved to cover entirely the image in order to create different descriptors for each image, as showed in Fig. 3.3. Furthermore, it is important to partially overlap some areas of the image, at different level of the grid detector algorithm, to give more importance to small patches of images in the regional descriptors creation phase.

The grids used in the proposed region detector are squares and rectangulars of different sizes, as explained in the Table 4.1.As a result, this method produces a lower number of regions than the original one, allowing to reduce the time spent in the creation of R-MAC+ descriptors, without loss in the final accuracy.

The third improvement introduced is a novel retrieval method. Usually, after the creation of all R-MAC descriptors, for each query image a similarity ranking is constructed through the sorting of database images based on their L_2 distances from the query descriptor. The new retrieval method proposes to modify the descriptors evaluated in the retrieval phase. For the query images it proposes to use R-MAC+ descriptors and for the database images "db regions" are suggested. The "db regions", represented in Fig. 3.4, are the MAC+ descriptors obtained during the process of creation of R-MAC+ descriptors of the database images. They represented the maxima of the different regions detected with the grid mechanism. Of course, they are related only to the database images.

The "db regions" are used in the retrieval phase instead of R-MAC+ descriptors of database images because a small part or a region of the image can well represent the entire image. This is due to the features extracted from CNN and max-pooling. Besides, sum-pooling and L_2 normalization reduce the value of features of the re-

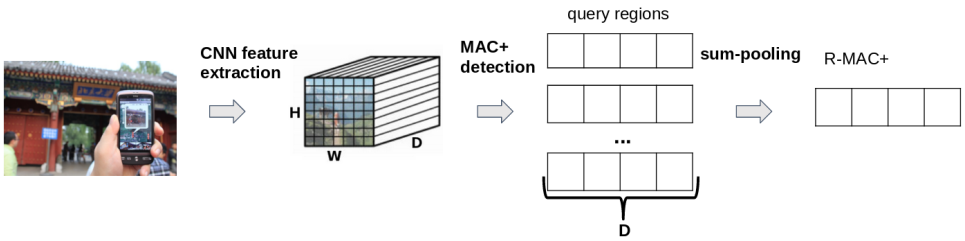

Figure 3.5: Pipeline of creation of R-MAC+ descriptors for the query images

gions, but this does not happen with the proposed retrieval strategy. Instead, on the query side, R-MAC+ descriptors are used as represented in Fig. 3.5, because it is preferrable to have an unique query descriptor for the retrieval phase, otherwise if the "query regions" are used there would be confusion or mismatching.

The advantage of this approach consists in comparing different "db regions" of the database images with the descriptor of the query image, choosing the one that obtains the best similarity or the minimum L_2 distance to the query descriptor. If the multi-resolution approach is adopted, there will be 45 regions for each database image to check for the retrieval, otherwise only 15 regions. This requires more time, but with an important extra boost on the final accuracy retrieval performance.

3.3 Dense-Depth locVLAD

Since VLAD-based embedding works better with dense representations, we introduced a novel representation scheme.

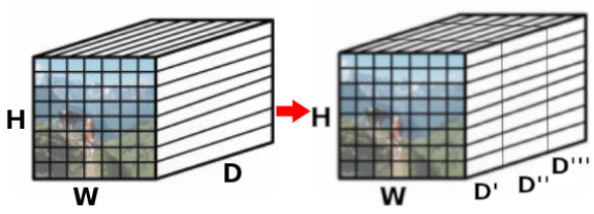

Figure 3.6: Dense-depth detector

The features extracted from mixed8 are grouped into a larger set of features of lower dimensionality in order to augment the VLAD descriptive quality. Given a feature map of dimension $W \times H \times D$ (W = width, H = height and D = depth), we split it along the depth axis in order to obtain a high number of features of lower dimension, as it can be seen in Fig. 3.6. Splitting along D allows to maintain the geometrical information of the feature maps, because features that have different position on $H \times W$ are not aggregated. Following this method, the number of descriptors changes from $H \times W$ to $H \times W \times \frac{D}{split\,factor}$. The $split\,factor$ indicates the dimension of every single descriptor obtained after the split along the depth axis. This value needs to be a trade-off between the number of features and their discriminative quality. As an example, a feature map of 8x8x1280, with $split\,factor = 128$, will be transformed in a set of $8 * 8 * 10 = 640$ descriptors of $128D$. Thereinafter, we will refer to this as *Dense-Depth Representation* (DDR).

After the extraction, the CNN codes are normalized using the root square normalization described in [38].

We will demonstrate in the experiments that the newly proposed DDR achieves higher performance.

Chapter 4

Large-Scale Retrieval

On any given Sunday you're gonna win or you're gonna lose. The point is, can you win or lose like a man?

Tony D'Amato

4.1 The Indexing Problem

The problem of Content-Based Image Retrieval has reached awesome results in small-scale datasets. When the algorithms are tested on large scale, some issues, irrelevant with small amount of data, quickly become fundamental for the performance of the system. The pipeline needs to maintain high the trade-off between effectiveness and efficiency for the retrieval on big datasets. Searching the most similar images in a big dataset is a costly operation, because exhaustive search has a complexity of $O(n)$, where n is the number of images in the dataset. For this reason on a big dataset, the Nearest Neighbor (NN) search becomes infeasible, due to the well-known problem of the curse of dimensionality. On the other hand, it is preferrable to apply an Approximate Nearest Neighbors (ANN) search. ANN returns a point that has a distance from the query equals to at most c times the distance from the query to its nearest points, where $c = (1+\varepsilon) > 1$. ANN are implemented in indexing systems with the

objective to speed up the access to the data, because during the query phase, the retrieval is executed only on a subset of database elements. Thanks to these methods, the complexity in the search can be reduced up to $O(\log(n))$.

In the following lines some of the most famous and used indexing methods will be explained.

Product Quantization (PQ) [39] decomposes the space into a Cartesian product of low dimensional subspaces and quantizes each subspace separately. A vector is represented by a short code composed of its subspace quantization indices.

PQ works as follow:

- the input vector x is split into m distinct subvectors $u_j, 1 \leq j \leq m$ of dimension $D^* = D/m$, where D is a multiple of m;
- the subvectors are quantized separately using m distinct quantizers;
- a given vector x is therefore mapped as follows:

$$\underbrace{x_1,\ldots,x_{D^*}}_{u_1(x)},\ldots,\underbrace{x_{D-D^*+1},\ldots,x_D}_{u_m(x)} \rightarrow q_1(u_1(x)),\ldots,q_m(u_m(x)) \tag{4.1}$$

  where q_j is a low-complexity quantizer associated with the j^{th} subvector;
- the index set $\mathscr{I}_j$, the codebook $\mathscr{C}_j$ and the corresponding reproduction values c_j are associated with the subquantizer q_j;
- the codebook is therefore defined as the Cartesian product:

$$\mathscr{C} = \mathscr{C}_1 \times \cdots \times \mathscr{C}_m \tag{4.2}$$

  where a centroid of this set is the concatenation of centroids of the m subquantizer.

In Permutation-Pivots [40] method, database and query elements are represented by permutations of a set of reference objects. Firstly, the reference objects are selected, in the basic version of the algorithm they are randomly chosen. Secondly, for every descriptor (database and query), the distance between the k reference objects

is calculated and is related to the data type. Usually, the Euclidean distance is used. After that, the sorting step is executed and it consists in sorting these distances with the target to creat a initial ranking, that will used in the creation of the final vector, used in the retrieval step. At the end, in order to improve the retrieval results it is possible to apply a re-ranking method based on the original distance. The advantages of this final step are manifold: it is simple to implement, easy to understand and also it produces good results.

Locality-Sensitive Hashing (LSH [41]) is an hashing function that allows to create an index of the descriptor allowing the creation of an indexing system. The main principle of LSH is that it projects points that are close to each other into the same bucket with high probability. It does not mean that similar elements will be in the same bucket, but the probability of this event is high and it is related to the similarity between the two elements. There are many different types of LSH, such as E2LSH [42], multi-probe LSH [43], and many others.

LSH has been first introduced by Indyk and Motwani in [41] and is defined as follows [44]: a family of hash functions $\mathscr{H}$ is called (R, cR, P_1, P_2)-sensitive if for any two items $\mathbf{p}$ and $\mathbf{q}$

- if $dist(\mathbf{p},\mathbf{q}) \leq R$, $Prob[h(\mathbf{p}) = h(\mathbf{q})] \geq P_1$
- if $dist(\mathbf{p},\mathbf{q}) \geq cR$, $Prob[h(\mathbf{p}) = h(\mathbf{q})] \leq P_2$

with $c > 1$, $P_1 > P_2$, R is a distance threshold, and $h(\cdot)$ is the hash function used.

The hash function used in LSH is a scalar projection:

$$h(\mathbf{x}) = sign(\mathbf{x} \cdot \mathbf{v}) \tag{4.3}$$

where $\mathbf{x}$ is the feature vector and $\mathbf{v}$ is a vector with the components randomly selected from a Gaussian distribution $\mathscr{N}(0,1)$.

The process is repeated L times, with different Gaussian distributions, to increase the probability to satisfy the above constraints. The retrieval phase using LSH has a complexity of $O(\log(N))$, where N is the number of descriptors. Summarizing, performing a similarity search query on an LSH index consists of two steps:

1. apply LSH functions to the query image q;

2. search in the corresponding bucket possible similar images, ranking the candidate objects according to their distances to q.

A different LSH implementation called *multi-probe LSH* [43] was proposed in order to reduce the number of hash tables needed for the application of the LSH method. The idea of this algorithm is to check, extending the LSH process, also the buckets near the query bucket. This approach has several advantages: the first one is the reduction of the hash tables used for the projection phase and the second one is the improvement of the accuracy in the retrieval thanks to the greater number of elements checked during query time than the LSH approach.

4.2 Bag of Indexes

I proposed a multi-probe LSH technique called *Bag of Indexes* (BoI) [45], that is an algorithm for ANN task, similar in the spirit to BoW approach, but more efficient. Figure 4.1 reports a silly example of BoI working. In the example, 3 hash tables are used to index 7 elements. The elements are already stored in different hash tables (it means that the hash functions used in the three hash tables are different). For the retrieval of the query, the BoI structure is introduced. In the end, from the figure, it is possible to infer that the most similar images to the query are third and the fifth images contained in the database set due to the high weight obtained by the image indexes 3 and 5 in the graph.

More specifically, BoI creates a vector of n weights (one for every image), which represent the number of times that the image is found in a bucket. The weights will be used later in the retrieval phase because greater will be the weight and more similar will be the couple of images (query and database image). So, this process allows to make a coarse-grain evaluation of the similarity between the query image and the other images. Then, a fine-grain evaluation is executed with the target of increasing the retrieval performance. Only the ε elements of the vector with the highest weights are re-ranked according to their Euclidean distance from the query. In the end, the

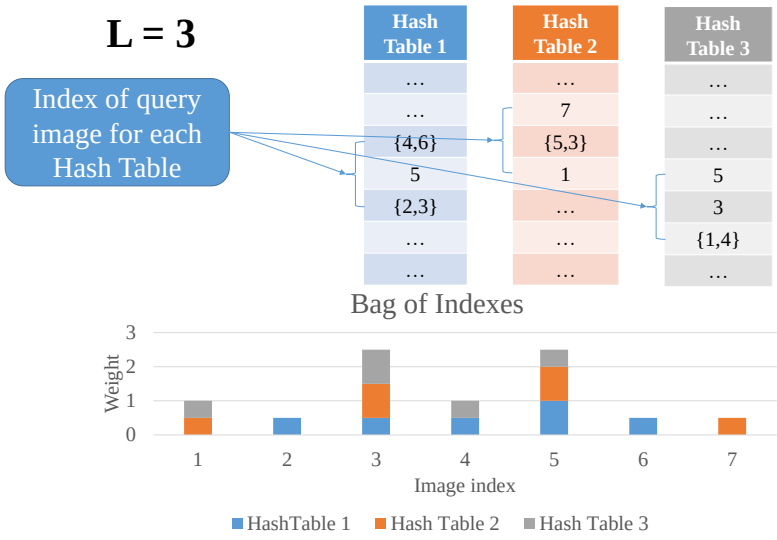

Figure 4.1: Overview figure of the retrieval through BoI multi-probe LSH.

nearest neighbour is searched only in this short and sorted re-ranked list. The main advantage of using a small list for the nearest neighbuor problem is that the Euclidean distance is calculated less times than in standard LSH approach, allowing to greatly reduce the resources and the computational time. Then, another advantage of this approach, unlike LSH, is that it does not require to maintain a ranking list without duplicates for all the L hash tables.

As, other indexing techniques, also BoI can be used in combination with other approaches, different from hashing functions. However, the best trade-off performance-retrieval time, that is common target, can be achieved when it is used with multi-probe LSH, because considering the l-neighbouring buckets the retrieval time increases, but also the accuracy reaches excellent result.

As before reported, the multi-probe LSH approach improves the final retrieval performance due to greater number of buckets considered during the retrieval and the greater probability of retrieving the correct result than LSH. The advantage of multi-probe LSH is due to the exploit of the main principle of LSH that similar objects

should fall in the same bucket or in the ones that are close to each other. An improvement of the BoI structure is the application of the multi-probe LSH technique at the BoI approach. The BoI multi-probe approach differs from the previous method for the query phase, because the projection phase is the same as BoI. In the query phase, many buckets will checked in order to find the correct results of the query. The buckets that will be checked are not randomly chosen, but they are based on the Hamming distance between to the query bucket. The l-neighbours are the buckets that have a Hamming distance less than or equal to l from the hashed value of the query, which correspond to the query bucket. The weights for the any value of l are chosen as follows:

$$w(i,q,l) = \begin{cases} \frac{1}{2^{H(i,q)}} & \text{if } H(i,q) \leq l \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

where i is a generic bucket, q is the query bucket and $H(i,q)$ is the Hamming distance between i and q.

However, even if we want to account for some uncertainty in the selection of the correct bucket, we also want to weight less as soon as we move farther from the "central" bucket.

Unfortunately, BoI multi-probe approach increases the computational time due to the greater number of buckets checked for the search of neighbouring elements. To reduce this problem, I introduced the *BoI adaptive multi-probe LSH*. It starts with a large number of neighbouring buckets γ_0 (e.g., 10) and slowly reduce γ when the number of hash tables increases. This iterative increase of focus can, on the one hand, reduce the computational time and the possible noise that can be found during the search executed by the multi-probe approach. In fact, at each iteration, the retrieval results are supposed to be more likely correct and the last iterations are meant to just confirm them, so there is no need to search on a large number of buckets.

Two different techniques for the reduction of the number of hash tables are evaluated:

- *linear*: the number of neighbouring buckets γ is reduced by 2 every 40 hash

tables, i.e.:

$$\gamma_i = \gamma_{i-1} - 2, \quad i = \{1, \Delta_1, \ldots, k_1\Delta_1\} \tag{4.5}$$

with $\Delta_1 = 40$ and $k_1 : k_1\Delta_1 \leq L$

- *sublinear*: the number of neighbouring buckets γ is reduced by 2

$$\gamma_i = \begin{cases} \gamma_{i-1} & \text{if } i \leq L/2 \\ \gamma_{i-1} - 2 & \text{if } i = \{L/2, L/2+\Delta_2, \ldots, L/2+k_2\Delta_2\} \end{cases} \tag{4.6}$$

with $\Delta_2 = 25$ and $k_2 : L/2 + k_2\Delta_2 \leq L$

The parameters has been chosen through fine tuning after the execution of many test. A configuration example is reported in Table 4.1:

Table 4.1: Summary of notation.

Symbol	**Definition**	**Chosen value**
n	number of images	-
δ	hash dimension	$2^8 = 256$
L	number of hash tables	100
γ_0	initial gap	10
l	neighbours bucket	1-neighbours
ε	elements in the re-ranking list	250
-	reduction	sublinear

The proposed approach contains several parameters. Their values were chosen after an extensive parameter analysis (out of the scope of this paper) and summary of notation is reported in Table 4.1. L, δ and l should be as low as possible since they directly affect the number of buckets $\mathcal{N}_q^l$ to be checked at each query q (and, therefore, the computational time), as follows:

$$\mathcal{N}_q^l = L\sum_{i=0}^{l}\binom{\gamma_i}{i} = L\sum_{i=0}^{l}\frac{(\gamma_i)!}{i!\,(\gamma_i - i)!} \tag{4.7}$$

where $\gamma_i = \gamma_0 = \log_2 \delta, \forall i$ for standard BoI multi-probe LSH.

Chapter 5

Graph Theory and Diffusion

Veni, vidi, vici!

Giulio Cesare

5.1 Graphs

Graphs are structures composed by a set of nodes $V = \{(v_1, v_2, \ldots, v_n)\}$ and a set of edges $E = \{(e_1, e_2, \ldots, e_n)\}$.

They are used in several computer vision tasks: applying diffusion for image retrieval [46], executing unsupervised fine-tuning [47], propagating labels for semi-supervised learning [48], creating manifold embedding [49] and training a classifier, exploiting the cycles found in the graph [50].

The graphs are usually used because on this structure it is possible to infer extra information useful for own task [51].

Specifically, the nodes are the symbolic representation of the dataset images and the edges represent the connections between the images. The connection between images is weighted and represents the similarity between two images: the higher the weight, the more similar the two images are. The weights of the edges are set with

the the cosine similarity calculated between the embeddings.

For our problem we will use the kNN graph, that is an undirected weighted graph. There are many techniques in literature for the creation of the kNN graph. The first and the simplest one for solving the problem is the brute-force approach which results to be very slow, but it usually obtains the best results. It consists in the connection of all the nodes through the use of an huge quantity of edges. Approximated kNN graphs can be used in order to speed up the process, but maintaining good retrieval accuracy. There are two set of strategies for the construction of the approximate kNN graph: methods following the strategy divide and conquer, and methods using local search strategy (e.g., NN-descent [52]). The divide-and-conquer strategy is usually composed by 2 tasks: subdivision of the dataset in subsamples (divide), and, then, for each subsample it executes the creation of the kNN graph and the merging all the subgraphs in the final kNN graph (conquer).

As foreseeable, the performance and the computational time depend on the number of subdivisions. It is worth to note that the first step is very important because you need to exploit an algorithm that fastly and accurately subdivide the dataset in clusters. One of the most famous clustering method is K-means [53], but it is also very slow, so it is not the perfect solution for this problem. I proposed to use Locality Sensitive Hashing (LSH) projections [41] for creating the approximate kNN graph [54]. Other papers as Chen *et al.*[55] follow the same strategy, but applying recursive Lanczos bisection [55]. They proposed two divide steps: the overlap and the glue method. In the former, the current set is divided in two overlapping subsets, while in the latter, the current set is divided into two disjoint subsets and a third, called gluing set, is used in order to merge the two resulting disjoint subsets. Instead, Wang *et al.*[56] proposed a method for the creation of an approximate kNN graph through the application in several iterations of the divide-and-conquer strategy. One of the novelty of the method is that the subsets of the dataset elements used during the divide phase are randomly chosen. This process needs to be repeated several times in order to theoretically cover the entire dataset. Our approach differs from the previous reported approaches for projections step. Moreover, the strategy followed for creating the graph is different and more efficient.

Regarding the second strategy based on the local search, Dong *et al.*[52] implemented the NN-descent [52] algorithm, based on the fact that “a neighbour of my neighbour is my neighbour". Starting from a random kNN list, that contains all the image descriptors, then the method searches random pairs ($< v, w >$) on the kNN list, after that it calculates the similarity between the elements ($dist(v, w)$) and in the end updates two kNN lists ($< v, dist(v, w) >$ and $< w, dist(v, w) >$). This process is repeated until the number of the updates are less than a threshold. Obviously, the growth of the neighbours contained in the kNN list increases the final performance but it tends to the brute-force kNN approach, therefore the trade-off between the speed and accuracy performance needs to be correctly managed. Park *et al.*[57], Houle *et al.*[58] and Debatty *et al.*[59] proposed variations to the NN-descent, by adapting the basic approach to their specific application domains. Sieranoja *et al.*[60] recently proposed a new method, called Random Pair Division, that exploits both the previous strategy: divide and conquer and NN-descent. The subdivision of the dataset elements is executed through a random procedure: two dataset descriptors are randomly selected, then if the descriptor to be clustered is close to the first one, it will be put in the first set, otherwise in the second one. This process will be repeated for all the image descriptors of the datasets. After that, a condition for stopping the process is tested: if the size of each set is greater than a threshold, the subdivision process continues in the same way only for this large set otherwise the process terminated. The conquer phase is executed through the application of the brute-force approach. In the end, for improving the performance of the method, a one-step neighbour propagation is applied. The last step exploits the principle of NN-descent and then the similar nodes not connected will be connected.

5.2 Ranking with Diffusion

The diffusion process allows to spread the query similarities on a graph of the dataset elements in order to catch all the neighbours elements to the query. This method increases the final retrieval performance in an astounding way, as reported in Fig. 5.1. In this figure two data distribution are showed: red and orange dots. The black point

is the query. In this case, the task is to find the neighbours of the query (black point). The usual scenario is the application of the Euclidean metric, but it does not achieve the best performance. The diffusion application, through the manifold distribution, reaches better retrieval results than the previous one.

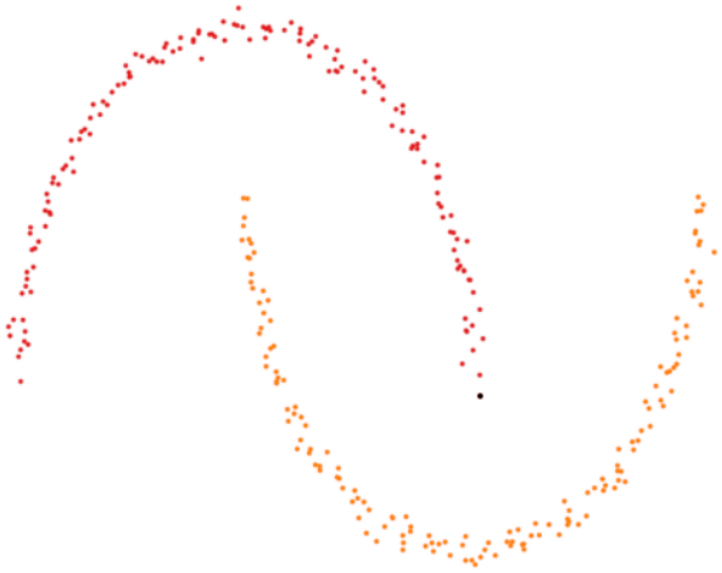

Figure 5.1: Data distributions in which diffusion can help to boost the retrieval results.

However, it is fundamental for the diffusion application the presence of the affinity matrix.

The affinity matrix A is the adjacency matrix of a weighted undirected graph G. This matrix has several properties: symmetric ($A = A^T$), positive ($A > 0$) and with zero self-similarities ($diag(A) = 0$). After that, to complete the diffusion process it is mandatory to calculate the Laplacian of the graph $L = D - A$, where $D = diag(A1_n)$ is the degree of the graph and $A1_n$ is the diagonal matrix with the row-wise sum of A. For improving the performance it is common to normalize the affinity matrix for

obtaining the transition matrix $S = D^{-1/2}AD^{-1/2}$ and the Laplacian $\mathscr{L} = I_n - S$ where I_n indicates the identity matrix, that has size equals to n.

Zhou *et al.*[61] were the firsts to propose to apply diffusion for retrieval purposes. They start the diffusion process from the query points to the database elements. They created a vector $y = (y_i) \in \mathbb{R}^n$ in this way:

$$y_i = \begin{cases} 1 & if \quad x_i \quad is \quad a \quad query \\ 0 & otherwise \end{cases}$$

The diffusion mechanism wants to find the neighbours of the starting point, the query. This it is possible through a ranking function $f = (f_i) \in \mathbb{R}^n$, that allows to generate a vector with the similarity score of each image x_i to the query. In the end, all the score will be sorted and it will be obtained the neighbours elements of the query. More formally, the diffusion can be represented in the following way by the ranking function:

$$f^t = \alpha S f^{t-1} + (1-\alpha)y$$

The parameter $alpha$ used in the ranking function depends to the random walk process executed on the graph because in the first term it indicates the probability to jump on an adjacent vertex according to the distribution S and $(1-\alpha)$ indicates the probability to jump to a query point. This process, repeated several times, allows for each point to spread their ranking score to their neighbours in the graph. Exploiting this principle it is possible to better capture the manifold structure of the dataset than applying the Euclidean distance and finally improving the retrieval performance.

5.3 Efficient kNN Graph Creation

I proposed an approach based on LSH projections to divide the global descriptors of the dataset in many subsets that fastly and efficiently solve the kNN graph creation problem with the objective of diffusion application for image retrieval task. This method is preferrable to the brute force because it is faster and it allows to obtain the same retrieval performance after diffusion application on the graph than the brute force.

Starting from a dataset $\mathscr{S} = \{s_1, \ldots, s_N\}$, composed by N images, and a similarity measure $\theta : \mathscr{S} \times \mathscr{S} \to \mathbb{R}$, the kNN graph for $\mathscr{S}$ is a undirected graph G, that contains edges between the nodes i and j with the value from the similarity measure $\theta(s_i, s_j) = \theta(s_j, s_i)$. The similarity measure depends to the topic and can be calculated in several ways. For our problem, the suggestable metric is the cosine similarity, so it is calculated through the application of the dot product between the image descriptors.

My approach called LSH kNN graph [62] follows the idea to firstly subdivide the entire dataset in many subsets, based on the concept of similarity between the images contained in the dataset. The name of the method comes from LSH because hashing functions are used to create a set of buckets $B = \{B_1, \ldots, B_N\}$ from several hash tables $L = \{L_1, \ldots, L_M\}$. The number of buckets depends on the bits used for the projection (δ) and to the number of the hash tables (M) used in this way: $N = 2^{\delta} \cdot M$. Each of these buckets will contain the projected elements $B_1 = \{b_{11}, \ldots, b_{1n}\}$. The result of this process represents an approximate result because it is not possible, for all the elements in the dataset, to project in the same bucket an element with all of its neighbours. Moreover, it is mandatory to get a trade-off between the number of the buckets for each hash table (2^{δ}), modifying the bits used (δ) for the hashing, and the number of hash tables (M) used for the projection. Consequently, using a small number of buckets allows to project more data in the same bucket. This strategy reduces the computational time for this step, but it increases the entire computation of the proposed approach. Otherwise, with a high number of buckets in each hash table, it increases the computational time of this phase, but it reduces the overall computation of the proposed method.

Then, in the second phase for each bucket, a brute-force kNN graph $G_i = \{\forall(b_{ix}, b_{iy}) \in B_i : \theta(b_{ix}, b_{iy})\}$ is constructed. Experimentally it has been shown that applying a brute-force approach on many subsets is faster than apply one time the brute-force method on the entire dataset. In the end, all the subgraphs need to be merged in the final graph $G = G_1 \cup \cdots \cup G_N$.

The proposed approach does not follow exactly the divide and conquer strategy as instead the method based on LSH [54].

5.3.1 Multi-Probe LSH kNN Graph

In order to reduce the number of hash tables used a multi-probe version is proposed, called multi LSH kNN graph. This method based on multi-probe LSH algorithm [43] differs from the previous one for the projection step because in this case it exploits the main principle of multi-probe LSH, in which the system, during the search step, checks also the neighbours buckets. On this way during the projection phase all the elements are projected in the neighbours buckets, but only in the 1-neighbourhood. This represents the set of buckets differing from a bit to the analysed bucket or that the Hamming distance between the buckets is less or equal than 1 ($H_d \leq 1$). It is worth to remember that the buckets are created with binary codes so the distance in binary space is measured using bits. More formally, the elements obtained with the application of the multi-probe LSH are the followings: $B_{multi-probe} = \{b_{x1}, \ldots, b_{xp}\} : H_d(b_{query}, b_{xj}) \leq 1 \wedge b_{xj} \in B; x \in \{1, \ldots, P\}$.

Increasing the bits used for the projection allows to grow at the number of neighbours of each bucket as following: $\sum_{i=0}^{l} \binom{\log_2 \delta}{i}$. Even though it increases the final retrieval performance, this proposed approach requires more time for the kNN graph creation than the previous one, so in order to get a good trade-off between the computational time required for the similarity measure calculations and the quality of the final graph, only a percentage γ of the elements projected on the 1-neighbour buckets are retained. The best trade-off is reached using $\gamma = 50\%$, that it means that each element is also projected randomly in the half of its 1-neighbour buckets.

Then, during the conquer phase, as in the previous LSH kNN graph method, all the pair of the indexes of the images found in the buckets will be connected through the calculation of the similarity measure.

5.3.2 Complexity Analysis

I briefly analysed the complexity of the proposed methods as follows.

For the first phase (projections phase with LSH), the complexity will be $O(\delta \cdot \Delta \cdot L \cdot N)$, where N are the images contained in the dataset, δ is the number of bits used in each projection, L is the number of different each tables generated and Δ represents

the dimension of the embedding used for the representation of the input image. On the other hand, for multi-probe LSH kNN graph the complexity of the first step will be greater than in the previous case because each image is also projected in different buckets: $O(\delta \cdot L \cdot \Delta \cdot (\frac{N \cdot \gamma \cdot L}{100}) \cdot N)$.

Then, the second phase is the calculation of the similarity measure of the all possible pairs of elements contained in a bucket that has a complexity of $O(n^2 \cdot 2^{\delta} \cdot L)$ where n represents the number of elements found in the bucket. Following a linear distribution on the different buckets, the value of n can be approximated in this way: $n \sim \frac{N}{2^{\delta}}$.

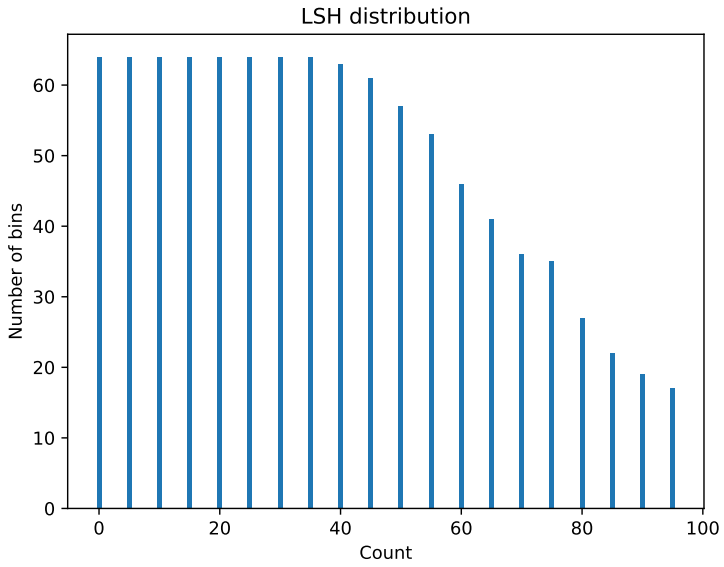

Figure 5.2: Distribution of dataset images projected through LSH on Oxford5k with $\delta = 6$ and $L = 20$.

For supporting the hypothesis of a linear distribution the Figures 5.2 and 5.3 are presented. They show the LSH distributions (for different values of δ) on Oxford5k. The values reported in each graph represent the number of buckets, that at least contain a certain number of items. In the case of $\delta = 6$ and $L = 20$, from the graph it is possible to infer that all the buckets contain at least 35 elements and only 20 on

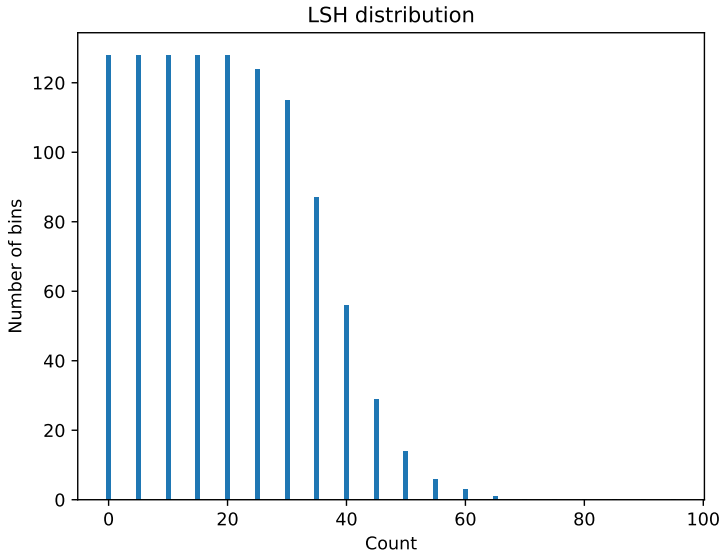

Figure 5.3: Distribution of dataset images projected through LSH on Oxford5k with $\delta = 7$ and $L = 20$.

64 buckets contain 95 elements. Similar results are obtained in the other case when $\delta = 7$ and $L = 20$, but with different values for the count axis.

The last step (combination of all the subgraphs) does not require operations and so the complexity of the method remains unchanged because in my algorithm all the subgrahps are directly appended on the final graph.

The neighbor propagation is not implemented in the proposed algorithm, but it has a complexity equals to $O(N \log_2 N)$ where N are the images contained in the dataset.

In the end, the final complexity of the proposed approach is the sum of the complexity reported for each step. The first equation represents the complexity for the LSH kNN graph and the second one represents the complexity of the multi LSH kNN graph method:

$$O(\delta \cdot \Delta \cdot L \cdot N) + O(n^2 \cdot 2^\delta \cdot L) = O(\delta \cdot \Delta \cdot L \cdot N + n^2 \cdot 2^\delta \cdot L) \tag{5.1}$$

$$O(\delta \cdot L \cdot \Delta \cdot (\frac{N \cdot \gamma \cdot L}{100}) \cdot N) + O(n^2 \cdot 2^{\delta} \cdot L) = O(\delta \cdot L \cdot \Delta \cdot (\frac{N \cdot \gamma \cdot L}{100}) \cdot N + n^2 \cdot 2^{\delta} \cdot L) \quad (5.2)$$

The complexity values of LSH kNN can be simplified in the following way:

$$O(\delta \cdot \Delta \cdot L \cdot N + n^2 \cdot 2^{\delta} \cdot L) = O(L(\delta \cdot \Delta N + (\frac{N^2}{2^{\delta+1}}) \cdot 2^{\delta})) = O(L(\delta \cdot \Delta \cdot N + \frac{N^2}{2})) \quad (5.3)$$

$$O(L(\delta \cdot \Delta \cdot N + \frac{N^2}{2})) = O(L \cdot N(\delta \cdot \Delta + \frac{N}{2})) = O(\frac{L \cdot N^2}{2} + L \cdot N \cdot \delta \cdot \Delta) = O(L \cdot N(\delta \cdot \Delta + \frac{N}{2})) \quad (5.4)$$

$$O(L \cdot N(\delta \cdot \Delta + \frac{N}{2})) = O(\frac{L \cdot N^2}{2} + L \cdot N \cdot \delta \cdot \Delta) = O(\frac{L \cdot N^2}{2} + L \cdot N \cdot \delta \cdot \Delta) \quad (5.5)$$

Only the simplification of LSH kNN method is reported because for multi LSH kNN is pretty much the same. Considering that the first term of the complexity is lower than the second, so the final complexity will be equals to $O(\frac{L \cdot N^2}{2})$ for LSH kNN graph and for multi LSH kNN graph methods.

5.4 Diffusion Parameters Optimization

As previously said, the diffusion process works well for retrieval purposes, but it requires the configuration of many parameters in order to obtain the best performance for each image dataset. For example, some of them are: the number of walks to execute on the graph and the number of neighbours to find and the number of database images to consider for the random walk process. Currently, for the experiments the configuration of these parameters is obtained through an extensive testing of several different configurations. As an alternative, a brute-force approach could be applied but it is infeasible due to the huge time necessary to test all possible combinations of the different parameters.

I proposed to use genetic algorithms to find the best configuration of these parameters in order to improve the retrieval performance for any image dataset.

5.4.1 Diffusion Parameters

Mathematically, the diffusion mechanism consists in the resolution of an equation system: $Ax = b$. The diffusion is similar in the spirit to the Google PageRank algorithm [63] in which a graph is used and the problem is solved by an iteration method. Experimentally it is demonstrated that better performance can be achieved, modifying the affinity matrix A as follows: $A = A - \alpha * A$, where α represents the damping factor used in the Google algorithm to adjust the connections between nodes. In their case, the best value for this parameter is set to 0.85, which is obtained after executing many experiments. In our case, this parameter is a real value in the interval $[0,1]$. Moreover, as previous motivated, the elements of the sparse affinity matrix can be raised to power by a factor β ($a_{ij} = a_{ij}^{\beta}$, where $a_{ij} \in A$) in order to remove useless neighbors, similarly to the power iteration method [64] used for the resolution of the Page Rank problem. The same reasoning can also be applied to the query vector $b_i = b_i^{\gamma}$, where $b_i \in b$.

The other parameters to optimize are:

- k_s, that is the number of steps to execute on the graph during the random walk process;
- k, that is the number of neighbours to find;
- $iterations$, the maximum number of iterations allowed for the algorithm to converge to the equation system solution;
- $trunc$, the number of database elements to be used during the application of the diffusion.

5.4.2 Genetic Algorithms

I proposed to use genetic algorithms to tune the diffusion parameters [65] because they can reach the target faster than the other existing optimization algorithms. Each individual corresponds to a specific setting of diffusion parameters and is represented by a string of seven values, corresponding to the seven parameters. The values have been set in the following ranges:

- $alpha \in \{0,1\}$ $(float)$;
- $\beta \in \{1,10\}$ (int);
- $\gamma \in \{1,10\}$ (int);
- $k_s \in \{20,100\}$ (int);
- $k \in \{5,40\}$ (int);
- $iterations \in \{10,30\}$ (int);
- $trunc \in \{3000, dataset\ size\}$ (int).

The fitness function to be maximized, that is the target of the genetic algorithms, corresponds to the mean Average Precision (mAP) obtained by the diffusion process in the retrieval process.

Generating random individuals, according to the constraints on the parameter ranges, allows to create an initial population, of size *Pop*. After that, the selection operation starts to work and each individual is replaced by the best of three individuals extracted randomly from the current generation (tournament selection). Then, the crossover step is executed and the selected individuals are crossed with a probability *CxPb*, generating new individuals by means of a single-point crossover.The mutation probability of each individual is called *MutPb*, instead the mutation probability of each gene is called *IndPb*. The population is then entirely replaced by the offspring (generational GA). The success of the genetic algorithms is also due to the several number of iterations and so the evolutionary process is iterated for *Gen* generations. Increasing the number of iterations allows to improve the final results, but after a lot of iterations the final performance will not be more improved.

In my case, only the best individuals (those leading to the largest mAP) found during the evolutionary process, and their corresponding fitness (mAP) values, are maintained.

The optimization method has been implemented using a an evolutionary computation framework in python for rapid prototyping and testing called DEAP[1] (Dis-

[1] `https://deap.readthedocs.io/en/master/`

tributed Evolutionary Algorithms in Python) [66].

Chapter 6

Dataset and Evaluation Metrics

And yet it moves.

Galileo Galilei

6.1 Datasets

There are many different image datasets for Content-Based Image Retrieval that are used in order to evaluate the algorithms. The most used are the following:

- **Holidays** [11] is composed of 1491 high-resolution images representing the holidays photos of different locations and objects, subdivided in 500 classes. The database images are 991 and the query images are 500, one for every class. An example of images contained in the dataset is reported in Fig. 6.2;

- **Oxford5k** [67] is composed by 5063 images representing the buildings and the places of Oxford (UK), subdivided in 11 classes. All the images are used as database images and the query images are 55, which are cropped for making more difficult the querying phase. An example of images contained in the dataset is reported in Fig. 6.3;

- **Paris6k** [68] is composed by 6412 images representing the buildings and the places of Paris (France), subdivided in 12 classes. All the images are used as database images and the query images are 55, which are cropped for making more difficult the querying phase. An example of images contained in the dataset is reported in Fig. 6.1;

- **UKB** [69] is composed by 10200 images of diverse categories such as animals, plants, etc.., subdivided in 2550 classes. Every class is composed by 4 images. All the images are used as database images and only one for category is used as a query image;

- **Flickr1M** [70] contains 1 million Flickr images under the Creative Commons license. It is used for large scale evaluation. The images are divided in multiple classes and are not specifically selected for the CBIR task.

Images taken from Flickr1M are used as distractors to make more difficult the retrieval phase. Adding images from Flickr1M to the most common datasets, other datasets have been created: Holidays+Flickr1M, Oxford105k and Paris106k. Finally, the datasets used for the training and fine-tuning of CNN-based architectures are:

- **ImageNet** [32] is composed by millions of images subdivided in many different categories. All the CNN architectures used are pre-trained on this dataset;

- **Landmarks-clean** [30] is composed by 49000 images from 586 classes.

Table 6.1 reports the characteristics of the presented datasets.

6.2 Evaluation Metrics

To evaluate the accuracy in the retrieval phase, mean Average Precision (mAP) is used on all the image datasets used. The mAP is the mean of average precision that identifies how many elements that finds are relevant to the query image. In order to compare a query image with the database, L_2 distance is employed.

Dataset	Images	Query images	Categories
Holidays	1 491	500	holiday photos (locations and objects)
Oxford5k	5 063	55	castles, buildings photos
Paris6k	6 412	55	houses, buildings, places photos
UKB	10 200	2 550	objects pictures
Flickr1M	1 000 000	-	photos of animals, places, objects, ...
Oxford105k	105 063	55	Oxford5k pictures and a subset of Flickr1M
Paris106k	106 412	55	Paris6k photos and a subset of Flickr1M
ImageNet	14 000 000	10 000	images of cars, animals, foods, ...
Landmarks-clean	49 000	586	buildings, castles, churces pictures

Table 6.1: Summary of image dataset characteristcs.

6.2.1 The Importance of Diffusion for Retrieval

The diffusion improves the final retrieval performance on some public image dataset. This is showed in Fig. 6.4. Then, the need to fix up the drawbacks of diffusions.

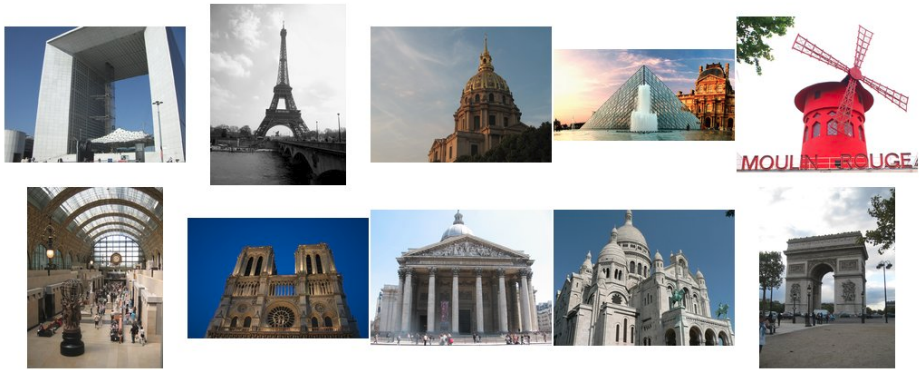

Figure 6.1: Subset of images contained in the Paris6k image dataset.

Figure 6.2: Example of images contained in the Holidays image dataset.

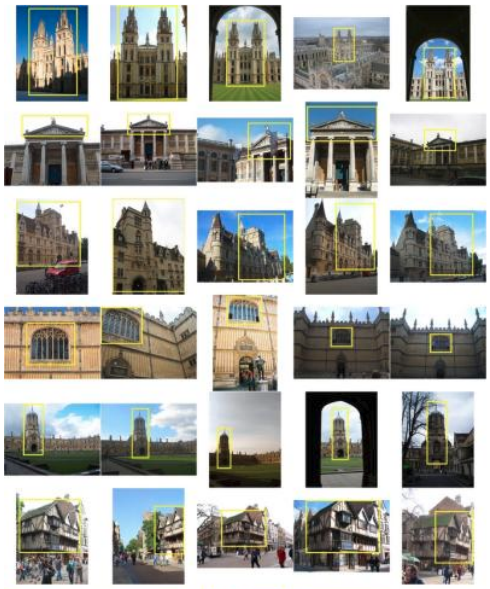

Figure 6.3: Subset of images contained in the Oxford5k image dataset.

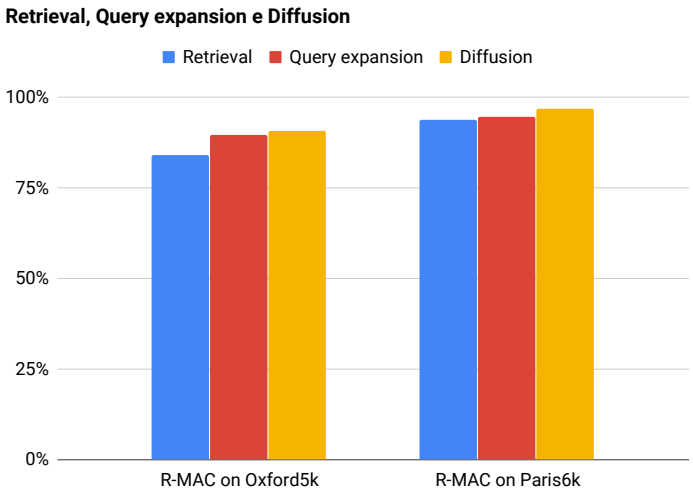

Figure 6.4: Comparison of results obtained using R-MAC descriptors tested with different approaches on Oxford5k and Paris6k.

Chapter 7

Experimental Results

Continuous improvement is better than delayed perfection

Mark Twain

7.1 Dense-Depth Representation

The proposed approach has been extensively tested on public datasets in order to evaluate the accuracy against the state of the art.

7.1.1 Results on Holidays

Table 7.1 reports the results obtained extracting the features with CNN pre-trained on ImageNet [32]. Different CNN architectures have been tested: VGG19 [6] and InceptionV3 [71].

In some experiments, the features are created using a filter that maintains the aspect ratio of the images and not a square one, because it showed to achieve higher performance, as shown in Table 7.1 with an asterisk. Looking at the results summarized in Table 7.1 it is worth noting that the PCA-whitening does not compromise the accuracy (even by a drastic reduction of dimensionality from 12800D to 128D), but it

Network	Layer	Input	DDR	Root square norm.	Encoding	PCA whit.	mAP
VGG19	block4_pool	224x244	✓	✗	VLAD	✗	74.33%
VGG19	block5_pool	550x550	✓	✗	VLAD	✗	77.80%
Inception	mixed_8	450x450	✓	✗	VLAD	✗	81.55%
Inception	mixed_8	450x450	✓	✗	locVLAD	✗	84.55%
Inception	mixed_8	562x562	✓	✗	locVLAD	✗	85.34%
Inception	mixed_8	562x562	✓	✓	locVLAD	✗	85.98%
Inception	mixed_8	562x662*	✓	✓	locVLAD	✗	86.70%
Inception	mixed_8	562x662*	✓	✓	locVLAD	128D	86.99%
Inception	mixed_8	562x662*	✗	✓	locVLAD	128D	85.63%
Inception	mixed_8	562x662*	✓	✓	locVLAD	256D	89.38%
Inception	mixed_8	562x662*	✓	✓	locVLAD	512D	90.21%

Table 7.1: Results on Holidays. In all the experiments the vocabulary is composed by 100 visual words. Different networks, layers and operations are tested. * indicates cases where the image aspect ratio is maintained.

even slightly increases it (from 86.70% to 86.99%). Moreover, DDR has a significant impact on the accuracy (without it, the mAP goes from 86.99% to 85.63%).

Table 7.2 reports the results on Holidays dataset of the state of the art, compared with the best configuration of Table 7.1 of our method at different descriptor sizes (for comparison purposes).

All the implementations of 128D and some of bigger dimensions that reach worse results than our descriptor are reported. The gVLAD [18] is the only technique that is not based on CNN features. Its strenght is the concatenation of various VLAD, one for every principal orientation (usually 4 or 8) of the features extracted from the images. MOP-CNN [73] is more computationally expensive than our approach since it extracts features at different scales and then aggregates them in a final global descriptor. NetVLAD [25] introduces a layer in CNN that allows to create VLAD vectors.

Method	Dimension	Holidays
Spatial pooling [72]	256	74.20%
gVLAD [18]	128	77.90%
MOP-CNN [73]	512	78.40%
Neural codes [33]	128	78.90%
NetVLAD [25]	512	81.70%
Ng *et al.* [26]	128	83.60%
CCS [27]	128	84.30%
OC [28]	128	85.09%
R-MAC [29]	512	85.20%
Gordo *et al.* [30]	512	86.70%
Gordo *et al.* [36]	2048	90.30%
Our method	128	**86.99%**
	256	**89.38%**
	512	**90.21%**

Table 7.2: Comparison of state of the art on Holidays dataset at different descriptor sizes. Red, green and blue are adopted to highlights the results obtained with descriptors of the same size. Moreover bold results indicate the best performance achieved with the relative descriptor dimension.

This method is a precursor of transfer learning. Ng *et al.* [26] were the first authors to extract the features from intermediate layers of CNN for embedding them with image retrieval purposes. Spatial pooling [72] uses also features extracted from CNN, but it proposes some modifications on the pooling function. In order to improve the accuracy, several state-of-the-art methods introduced more sophisticated embedding techniques. For example, CCS strategy [27] combines different CNN codes and SIFT descriptors reaching good results. OC [28] uses a different approach since it does not densely extract features from images, but extract features only in some important

image regions, that are related to detected objects. R-MAC [29] proposes a new embedding technique, similar to VLAD, using object detection, re-ranking and query expansion techniques for improving the final accuracy results.

Neural codes [33] and Gordo *et al.* [30] use a fine-tuning strategy on VGG16 with the objective of increasing the accuracy wrt pure strategies based on pre-trained networks. Also Gordo *et al.* [36] outperformed the state of the art on Holidays, but using a descriptor with an huge dimension, obtained through the fine-tuning of ResNet101. It is rather evident that our method outperform the state of the art at comparable descriptors size. Table 7.2 reports also the results of our method by increasing to 256 and 512 the descriptors size. Though these cases result slower, they also allow us to achieve better performance and to be comparable with other methods using larger descriptors.

7.2 R-MAC+

In order to evaluate the accuracy of the proposed embedding technique with respect to the state of the art, we run experiments on public datasets and employing standard evaluation metrics.

7.2.1 Results on Oxford5k, Paris6k and Holidays

Table 7.3 reports our results obtained on: Oxford5k, Paris6k and Holidays.

In the first two experiments, VGG19 is adopted for the feature extraction step. It allows to extract feature maps of 512D, instead through ResNet50 is possible to extract feature maps of 2048D, reaching more accurate results than the first experiments thanks to the depth of the CNN architecture. The features are extracted from the layer "block5_ pool" on VGG19 and from the layer "activation_43" on ResNet50.

In Table 7.3 are presented the experiments executed without any re-ranking strategies, while in Table 7.4 reports results obtained using query expansion methods in order to improve the final accuracy. M-R RMAC+ with the retrieval based on "db regions" reached the best results on all the 3 public datasets. Moreover, in the cases of Oxford5k and Paris6k, the improvements compared to the previous experiments

Method	CNN	Oxford5k	Paris6k	Holidays
MAC†	VGG19	57.44%	73.15%	76.26%
R-MAC†	VGG19	65.56%	82.80%	87.65%
R-MAC†	ResNet50	71.77%	83.31%	92.55%
M-R R-MAC+	ResNet50	78.88%	88.63%	**94.63%**/95.58%
M-R R-MAC+ with retrieval based on "db regions"	ResNet50	**85.39%**	**91.90%**	94.37%/**95.87%**

Table 7.3: Results of the proposed methods obtained on some public datasets with different network used for features extraction phase and algorithms for global descriptors creation. † indicates that the method is re-implemented. M-R indicates that the multi-resolution approach is adopted. The results after / on Holidays represent the experiments executed on the rotated version of the dataset.

are remarkable. Unfortunately, the average query time on Oxford5k and Paris6k increases from 0.04s and 0.11s to 1.25s and 1.54s because a brute-force approach has been applied and the number of descriptors to be compared have been increased by 45 times.

For the query expansion approach, it has been decided to expand the query with a fixed number of descriptors, as explained in [74]. After some tests, the numbers of top-ranked descriptors used for the query expansion are: 8 for Oxford5k, 6 for Paris6k and 1 for Holidays. This is due to the number of images belonging to the classes: in Holidays, the majority of the classes are composed by a reduced number of elements, instead in Oxford5k and Paris6k, the classes are constituted by many images. Furthermore, the query expansion of "db regions" produces improvements only on Oxford5k. This method uses for the query expansion the "db regions" instead of the R-MAC+ descriptors of the database images.

Method	CNN	Oxford5k	Paris6k	Holidays
M-R R-MAC+ and query expansion	ResNet50	86.45%	92.01%	**94.97%**/95.97%
M-R R-MAC+ with retrieval based on "db regions" and query expansion	ResNet50	87.92%	**93.64%**	94.42%/**96.05%**
M-R R-MAC+ with retrieval based on "db regions" and query expansion of "db regions"	ResNet50	**88.78%**	92.30%	94.28%/95.91%

Table 7.4: Results of the proposed methods with query expansion techniques obtained on some public datasets. † indicates that the method is re-implemented. M-R indicates that the multi-resolution approach is adopted. The results after / on Holidays represent the experiments executed on the rotated version of the dataset.

7.2.2 Comparison with the State Of The Art

In order to have a fair comparison, Table 7.5 reports our results with those of several other state-of-the-art methods on the same datasets.

As reported in the introduction and in related works, the methods that implemented classical embedding as VLAD based on CNNs [25, 76] obtain better results than the ones based on hand-crafted methods for the feature extraction phase [75, 18].

The methods based on sum-pooling [77, 78] obtained worse results than the methods based on max-pooling [29].

Moreover, Gordo *et al.*, in [30, 36], with their recent improvements due to fine-tuning raised excellent result in the image retrieval task. Laskar *et al.* [82] improved the R-MAC pipeline through a saliency weighting of the regions extracted.

The proposed method R-MAC+ with retrieval based on "db regions" outperforms the state of the art on Holidays and reached good results on Oxford5k and Paris6k,

Method	Dimension	Oxford5k	Paris6k	Holidays
VLAD [75]	4096	37.80%	38.60%	55.60%
gVLAD [18]	128	60.00%	-	77.90%
NetVLAD [25]	4096	71.60%	79.70%	81.70%
Ng *et al.* [76]	128	55.80%*	58.30%*	83.60%
Neural codes [33]	128	55.70%*	-	78.90%
Babenko *et al.* [77]	256	65.70%	-	78.40%
Kalantidis *et al.* [78]	512	68.20%	79.70%	83.10%
Yan *et al.* [27]	128	-	76.76%	84.13%
Mopuri *et al.* [79]	128	-	70.39%	85.09%
BLCF [80]	25k	73.80%	82.00%	-
BLCF-SalGAN [81]	336	74.60%	81.20%	-
R-MAC [29]	512	66.90%	83.00%	85.20%
Gordo *et al.* 2016 [30]	512	83.10%	87.10%	86.70%
Gordo *et al.* 2017 [36]	2048	86.10%	94.50%	90.30%/94.48%
Laskar *et al.* [82]	2048	**90.20%**	**95.80%**	-
Seddati *et al.* [31]	variable	72.27%	87.10%	94.00%
R-MAC+	2048	85.39%	91.90%	**94.37%/95.87**

Table 7.5: Comparison of state-of-the-art methods on different datasets. * indicates that the method is applied on the full-size query images. The results after / on Holidays represent the experiments executed on the rotated version of the dataset.

overcame only by methods based on fine-tuning strategies [36, 30, 82]. The main drawback of fine-tuning is that, if more data to retrieve than the actuals are added then a new fine-tuning procedure needs to be issued, requiring more time as well as additional labelled data. With the addition of query expansion technique, the R-MAC of Gordo *et al.* [30, 36] also outperformed our approach, as reported in Table 7.6

Method	Dimension	Oxford5k	Paris6k	Holidays
R-MAC [29]	512	77.30%	86.50%	-
Kalantidis *et al.* [78]	512	72.20%	85.50%	-
Azizpour *et al.* [83]	4096	79.00%	85.10%	90.00%
Gordo *et al.* 2016 [30]	2048	89.00%	93.80%	-
Gordo *et al.* 2017 [36]	2048	**90.60%**	**96.00%**	-
R-MAC+	2048	87.92%	93.64%	**94.42%/96.05**

Table 7.6: Comparison of state-of-the-art methods with query expansion techniques on different datasets. The results after / on Holidays represent the experiments executed on the rotated version of the dataset.

because they obtained better results than ours in the first stage of the retrieval problem. Besides, the performance gap between our approach and the R-MAC of Gordo is maintained similar before and after the application of query expansion techniques. However, the proposed approach still shows state-of-the-art mAP on the Holidays dataset and on Oxford5k, Paris6k without the application of the fine-tuning strategy.

7.3 Bag of Indexing

The proposed approach has been extensively tested on public datasets in order to evaluate the accuracy against the state of the art.

7.3.1 Results on Holidays

In the case of Holidays+Flickr1M all the results are fairly comparable because the same locVLAD [24] descriptors are used for every experiment. Also, the same value of top results considered in each test is chosen, equals to 250. Choosing a small value for this parameters allows to obtain a reduced retrieval time, but usually with a loss in retrieval accuracy. FLANN [85] represents the state of the art of indexing techniques and reached 83.97%, instead LOPQ [84] obtained only 36.37%. More

Method	Holidays+Flickr1M	
	mAP	**avg retrieval time**
LSH	86.03%	3103 msec
Multi-probe LSH (L = 50)	86.10%	16706 msec
PP-index [40]	82.70%	2844 msec
LOPQ [84]	36.37%	4 msec
FLANN [85]	83.97%	995 msec
BoI adaptive multi-probe LSH [45]	85.35%	8 msec

Table 7.7: Results in terms of mAP and average retrieval time in msec on Holidays+Flickr1M.

specifically, even though LOPQ results to be faster than the others methods (only 4 msec), its retrieval accuracy is very low. Instead, FLANN achieved good retrieval accuracy, but with an average retrieval time of 995 msec. LSH and Multi-probe LSH obtained excellent results, but the average query time is very high (3103 msec for LSH and 16706 msec for Multi-probe LSH). PP-index [40] reached good results in terms of retrieval accuracy, but the query time is worse than the one of FLANN. Finally, the proposed BoI adaptive multi-probe LSH achieved an excellent trade-off between retrieval accuracy and time, with a mAP of 85.35% in only 8 msec for a query. This excellent performance is due to the reduced dimension of hash tables (where only 256 buckets are considered), the small numbers of hash tables (only 100), the effectiveness of LSH projection and, finally, the application of multi-probe LSH strategy during the retrieval phase, combined with the reduction of focus, proposed in the adaptive version of the BoI multi-probe LSH approach.

7.3.2 Results on Holidays+Flickr1M datasets

This section reports the results of our approach, by adding to the Holidays dataset a different number of distractors, obtained from the Flickr1M dataset. All the experiments have been conducted several times and a mean has been computed in order to

eliminate the randomness of the Gaussian distribution used in the hashing function. The embeddings used are locVLAD descriptors [24], while the features are extracted from the layer mixed8 of Inception V3 network [71] that is a CNN pre-trained on the ImageNet [32] dataset. The vocabulary used for the creation of locVLAD descriptors is calculated on Paris6k.

Method	ε	**Holidays+Flickr1M**	
		mAP	**avg retrieval time**
LSH*	250	86.03%	3 103 msec
Multi-probe LSH* (L = 50)	250	86.10%	16 706 msec
PP-index* [40]	250	82.70%	2 844 msec
LOPQ [84]	250	36.37%	4 msec
FLANN [85]	250	83.97%	995 msec
BoI LSH	250	78.10%	5 msec
BoI multi-probe LSH	250	85.16%	12 msec
BoI adaptive multi-probe LSH	250	85.35%	8 msec
PP-index* [40]	10k	85.51%	15 640 msec
LOPQ [84]	10k	67.22%	72 msec
FLANN [85]	10k	85.66%	1 004 msec
BoI adaptive multi-probe LSH	10k	**86.09%**	**16 msec**

Table 7.8: Results in terms of mAP and average retrieval time in msec on Holidays+Flickr1M. ε represents the number of elements adopted in the re-ranking phase. * indicates our re-implementation.

Table 7.8 summarizes the results on Holidays+Flickr1M dataset in terms of mAP and average retrieval time (msec). The first experiments evaluated only the top $\varepsilon =$ 250 nearest neighbours.

LSH and multi-probe LSH achieve excellent results, but with an huge retrieval time. Also PP-index [40] needs more than 3 seconds for a query to retrieve the results.

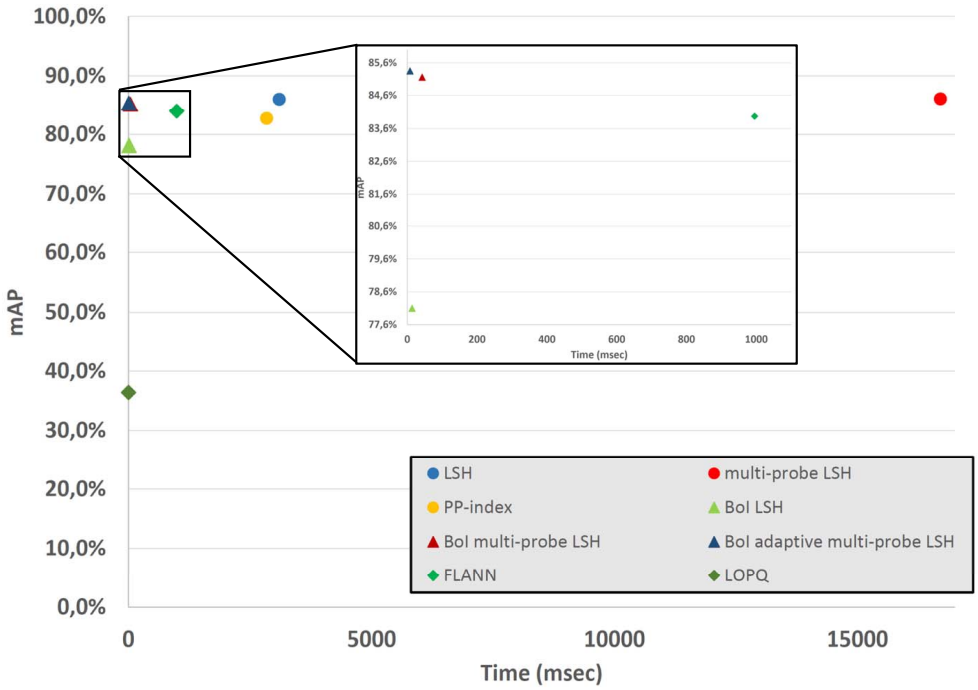

Figure 7.1: Relationship between time and accuracy on Holidays+Flickr1M with different approaches.

LOPQ[84] reaches poor results on large-scale retrieval with an accuracy equals to 36.37%, while FLANN [85] achieved a better result of 83.97%. However, while query time for LOPQ is pretty low compared to the other test cases, FLANN is not able to keep the query time low. It is worth saying that both LOPQ and FLANN has been tested using the available codes from authors and reported results correspond to the best found configuration of parameters. Given the significantly low (especially for LOPQ) performance in accuracy, further experiments have been conducted for LOPQ, FLANN, as well as PP-index and our method by increasing ε from 250 to 10k. As foreseeable, all the accuracy results improved with respect to $\varepsilon = 250$ (LOPQ increases from 36.37% to 67.22%), but the proposed BoI adaptive multi-probe LSH method still outperforms all the others. Moreover, our method still results to be faster

than the others (LOPQ is fast like ours, but with lower accuracy, while PP-index and FLANN are slightly lower in accuracy, but much slower).

Overall speaking, our proposal outperforms all the compared methods in the trade-off between accuracy and efficiency. To better highlight this, Fig. 7.1 shows jointly the mAP (on y-axis) and the average query time (on x-axis). The best trade-off has to be found in the upper left corner of this graph, i.e. corresponding to high accuracy and low query time. All the BoI-based methods clearly outperform the other methods.

Regarding the memory footprint of the algorithm for 1M images with 1M descriptors of 128D (float = 4 bytes), brute-force approach requires 0.5Gb (1M x 128 x 4). LSH needs only 100 Mb: 1M indexes for each of the L=100 hash tables, because each indexes is represented by a byte (8 bit) and so 1M indexes x 100 hash tables x 1 byte = 100Mb. The proposed BoI only requires additional 4 Mb to store 1M weights.

7.3.3 Results on Oxford105k and Paris106k datasets

Since our goal is to execute large-scale retrieval for landmark recognition, we have also used the Oxford105k and Paris106k datasets. In this case, all the methods are tested using R-MAC descriptors, fine-tuned by Gordo *et al.* [36], since VLAD descriptors are demonstrated to be not suited for these datasets [1].

Table 7.9 show the mAP and the average retrieval time. Using $\varepsilon = 2500$, the proposed approach obtained slightly worse results than PP-index, but resulted one order of magnitude faster in both datasets. When more top-ranked images are used ($\varepsilon = 10k$), BoI adaptive multi-probe LSH obtained the best results and with lower query time. Furthermore, LOPQ [84] works better on Paris106k than Oxford105k, while FLANN [85] performs poorly on both datasets.

7.4 KNN Graph Creation

Previous works have evaluated the methods for creating approximate kNN graphs by checking the number of common edges between the approximate and the exact kNN graph. In our case the kNN graph pipelines are evaluated after the diffusion and

Method	ε	Oxford105k		Paris106k	
		mAP	avg ret. time	mAP	avg retr. time
LSH*	2500	80.83%	610 msec	86.50%	607 msec
PP-index* [40]	2500	81.89%	240 msec	88.14%	140 msec
LOPQ [84]	2500	71.90%	346 msec	87.47%	295 msec
FLANN [85]	2500	70.33%	2118 msec	68.93%	2132 msec
BoI adaptive multi-probe LSH	2500	81.44%	12 msec	87.90%	13 msec
PP-index* [40]	10k	82.82%	250 msec	89.04%	164 msec
LOPQ [84]	10k	69.94%	1153 msec	88.00%	841 msec
FLANN [85]	10k	69.37%	2135 msec	70.73%	2156 msec
BoI adaptive multi-probe LSH	10k	**84.38%**	**25 msec**	**92.31%**	**27 msec**

Table 7.9: Results in terms of mAP and average retrieval time (msec) on Oxford105k and Paris106k. * indicates our re-implementation of the method.

retrieval modules in order to evaluate how effective (and efficient) are our proposals for the task in terms of retrieval accuracy when diffusion is applied. The diffusion approach and the R-MAC descriptors adopted are the same of the work of Iscen et al. [46].

7.4.1 Results on Oxford5k

Table 7.10 reports the retrieval results after diffusion application of different kNN graph techniques. Note that changing the values of LSH (δ and L) produces different results. The best configuration is $\delta = 6$ and $L = 2$ applying the multi LSH kNN graph approach. the best trade-off between the computational time for kNN graph creation and the final retrieval performance is the LSH kNN graph with $\delta = 6$ and $L = 20$. NN-descent produces good results, but it needs a lot of time for the graph creation (55 seconds). Furthermore, it does not obtain results comparable to the other methods.

RP-div [60] is very fast but collecting random elements from the dataset for the divide task does not give good results in retrieval after diffusion.

Method	LSH proj.	kNN graph creation	mAP
NN-descent [52]*	-	55 s	83.81%
RP-div [60] (size = 50)*	-	1.16 s	82.68%
Wang *et al.*[56]*	-	1.5 s	90.60%
Brute-force	-	1.33 s	90.79%
LSH kNN graph ($\delta = 6, L = 20$)	0.45 s	**0.52 s**	90.95%
LSH kNN graph ($\delta = 8, L = 10$)	0.4 s	0.95 s	88.98%
Multi LSH kNN graph ($\delta = 6, L = 2$)	0.29 s	1.54 s	**91.13%**

Table 7.10: Comparison of different approaches of kNN graph creation tested on Oxford5k. * indicates that the method is a C++ re-implementation.

The method implemented by Wang *et al.*[56] obtains a different result each execution, so the reported performance is the average of ten experiments. The approach is very fast, but did not achieve the best mAP. Note also that the brute-force method is executed on GPU, instead all the other methods are executed on CPU.

Figure 7.2: Query results obtained with R-MAC descriptor.

Figure 7.3: Query results after diffusion application.

Figure 7.2 and Figure 7.3 report the results obtained from the same query image. As you can see, the results obtained in the second case are better due to the diffusion application.

We also perform some experiments with regional descriptors (Table 7.11). The use of regional descriptors demonstrates an improvement on the final performance due to high number of descriptors for each image (usually 21). In this case the total number of descriptors used for the creation of the kNN graph are approximately 100K. Note we omit testing RP-div and NN-descent here due to poor previous accuracy/computation performance.

Method	**LSH proj.**	**kNN graph creation**	**mAP**
Wang *et al.*[56]*	-	148 s	91.69%
Brute-force	-	15816 s	93.80%
LSH kNN graph (δ=6, L=20)	9 s	100 s	**94.67%**
LSH kNN graph (δ=8, L=10)	6 s	**45 s**	93.68%
Multi LSH kNN graph (δ=6, L=2)	6 s	350 s	93.96%

Table 7.11: Comparison of different approaches of kNN graph creation tested on Oxford5k using regional R-MAC descriptors. * indicates that the method is a C++ re-implementation.

7.4.2 Results on Paris6k

Table 7.12 shows the results on the Paris6k dataset, which are similar to those obtained to Oxford5k.

However, in this case, LSH kNN graph is the fastest approach and also it obtains the best retrieval performance after the application of diffusion. Multi-LSH kNN method obtains a good result, but in more time than the brute-force approach.

Method	LSH proj.	kNN graph creation	mAP
NN-descent [52] (neighbours = 50)*	-	60.10 s	94.24%
RP-div [60] (size = 50)*	-	3.63 s	96.25%
Wang *et al.*[56]*	-	1.95 s	96.75%
Brute-force	-	1.81 s	96.83%
LSH kNN graph ($\delta = 6, L = 20$)	1 s	0.80 s	**97.01%**
LSH kNN graph ($\delta = 8, L = 10$)	0.78 s	**0.28 s**	95.93%
Multi LSH kNN graph ($\delta = 6, L = 2$)	0.35 s	2.28 s	96.81%

Table 7.12: Comparison of different approaches of kNN graph creation tested on Paris6k. * indicates that the method is a C++ re-implementation.

7.4.3 Results on Oxford105k

Table 7.13 reports results for the experiments executed on Oxford105k. again RP-div and NN-descent are not tested due to poor trade-off previously obtained. Increasing

Method	LSH proj.	kNN graph creation	mAP
Wang *et al.*[56]*	-	150 s	91.00%
Brute-force	-	4733 s	91.45%
LSH kNN graph ($\delta = 6, L = 20$)	23 s	**77 s**	92.50%
LSH kNN graph ($\delta = 8, L = 10$)	15 s	145 s	90.79%
Multi LSH kNN graph ($\delta = 6, L = 4$)	5 s	420 s	**92.85%**

Table 7.13: Comparison of different approaches of kNN graph creation tested on Oxford105k. * indicates that the method is a C++ re-implementation.

the dimension of the dataset illustrates the difference in accuracy and computational time between the proposed approach and brute-force. The proposed approaches ob-

tain better results and trade-offs than other methods. In particular, LSH kNN graph ($\delta = 6$ and $L = 20$) achieves 92.50% in only 77s for the graph creation process. The multi LSH kNN graph needs more time than the previous approach, but it reaches the best mAP on this dataset equals of 92.85%.

7.5 Genetic Algorithms for Diffusion

In this section we illustrate the experimental results we have obtained on three public datasets: Oxford5k, Paris6k and Oxford105k.

The results of the GA optimization are compared to the results obtained by other commonly used techniques for parameter tuning.

7.5.1 Results on Oxford5k

Different experiments have been executed on the Oxford5k dataset. In order to find the best configuration of the diffusion parameters, several combination of genetic algorithm parameters have been tested.

Gen	Pop	CxPb	MutPb	IndPb	mAP
10	50	0.5	0.2	0.1	94.31%
20	50	0.5	0.2	0.1	94.31%
50	50	0.5	0.2	0.1	**94.40%**
100	50	0.5	0.2	0.1	**94.40%**

Table 7.14: Results on Oxford5k varying the values of number of generations (*Gen*).

Tables 7.14-7.18 report the results obtained on Oxford5k by varying one parameter of the genetic algorithm at a time.

Starting from a standard configuration of the GA ($CxPb = 0.5$, $MutPb = 0.2$ and $IndPb = 0.1$), the number of generations and the population size have been varied from 10 to 100, considering a maximum budget of 5000 fitness computations. The

Gen	Pop	CxPb	MutPb	IndPb	mAP
50	10	0.5	0.2	0.1	94.32%
50	20	0.5	0.2	0.1	94.16%
50	50	0.5	0.2	0.1	**94.40%**
50	100	0.5	0.2	0.1	94.36%

Table 7.15: Results on Oxford5k varying the values of population size (*Pop*).

Gen	Pop	CxPb	MutPb	IndPb	mAP
50	50	0.1	0.2	0.1	93.73%
50	50	0.3	0.2	0.1	**94.41%**
50	50	0.5	0.2	0.1	94.40%
50	50	0.8	0.2	0.1	94.36%
50	50	1.0	0.2	0.1	94.34%

Table 7.16: Results on Oxford5k varying the values of crossover probability (*CxPb*).

Gen	Pop	CxPb	MutPb	IndPb	mAP
50	50	0.3	0.1	0.1	93.73%
50	50	0.3	0.2	0.1	**94.41%**
50	50	0.3	0.3	0.1	**94.41%**
50	50	0.3	0.4	0.1	94.32%
50	50	0.3	0.5	0.1	94.31%

Table 7.17: Results on Oxford5k varying the values of mutation probability (*MutPb*).

best configurations, as shown in Table 7.14 and 7.15, correspond to the largest numbers of fitness computations ($Gen = 50$, $Pop = 50$ and $Gen = 100$, $Pop = 50$). Since these configurations lead to the same mAP (94.40%), the remaining parameters of the GA have been varied starting from the configuration which is fastest to compute

Gen	Pop	CxPb	MutPb	IndPb	mAP
50	50	0.3	0.2	0.1	**94.41%**
50	50	0.3	0.2	0.3	94.40%
50	50	0.3	0.2	0.5	94.27%
50	50	0.3	0.2	0.8	94.23%
50	50	0.3	0.2	1.0	94.22%

Table 7.18: Results on Oxford5k varying the values of mutation probability for each gene (*IndPb*).

($Gen = 50$, $Pop = 50$).

Table 7.16 shows that the precision reaches its highest value for a crossover probability (*CxPb*) of 0.3 (94.41%). Regarding the mutation probability (*MutPb*), the best results have been achieved with values 0.2 (94.41%) and 0.3 (94.41%), as shown in Table 7.17. Considering the mutation probability for each gene (*IndPb*), the highest precision has been achieved with value 0.1 (94.41%).

Therefore, as shown in Table 7.18, the best set of parameters for the genetic algorithm thus obtained is: *Gen*= 50, *Pop*= 50, *CxPb*= 0.3, *MutPb*= 0.2, *IndPb*= 0.1. The corresponding configuration obtained for the diffusion parameters is: $\alpha = 0.97$, $\beta = 3$, $\gamma = 2$, $k_s = 53$, $k = 9$, $iterations = 10$, $trunc = 4136$.

After this preliminary analysis, another set of experiments has been performed. The number of generations has been increased in order to check the convergence status of the GA, obtaining a further improvement in the mAP. It is to be noticed that this set of experiments is less structured than the previous one, due to the longer computation time. The best set of GA parameters thus obtained ($mAP = 94.44\%$) is: 100 generations, population size equal to 50, crossover probability set to 0.3, mutation probability to 0.2 and mutation probability for each gene to 0.1. The corresponding configuration obtained for the diffusion parameters is: $\alpha = 0.97$, $\beta = 3$, $\gamma = 1$, $k_s = 95$, $k = 7$, $iterations = 10$, $trunc = 3046$. Given the stochastic nature of the GA, five independent runs of the algorithm have been executed to assess how repeatable

the results are (avg = 94.39%, stdev = 0.038, max = 94.44%, min = 94.34%).

Method	Fitness comp.	Time	mAP
Manual configuration [62]	1	2 s	90.95%
Random search [86]	5000	29045 s	93.67%
Random search [86]	10000	41610 s	93.81%
Grid search	5000	28680 s	92.73%
Grid search	10000	42513 s	94.43%
PSO [87]	5000	27767 s	94.30%
Genetic algorithms	**5000**	**17695 s**	**94.44%**

Table 7.19: Comparison of different approaches to the optimization of the diffusion parameters on Oxford5k in terms of mAP, time and number of fitness computations.

Table 7.19 reports the results of different optimization techniques applied on the diffusion process. For each technique the table shows the result of the best configuration found. The results have been compared in terms of mAP, running time and number of fitness computations.
The random search [86] has sampled, in this case, 5k and 10k configurations using uniform distribution for all the parameters to test.
The Particle Swarm Optimization [87] has been executed using the same number of fitness computation of the GA (population of 50 particles, 100 iterations). Moreover, the minimum speed is set to -0.50 and the maximum speed to 0.50.
The grid search has been performed over 5k and 10k different parameter setting. Given the large number of fitness computations it can be seen as a brute-force strategy.
“Manual configuration" means that the configuration of the parameters of the diffusion mechanism was taken from the literature.

The “manual configuration" technique obviously requires less time than the other methods, but it obtains the worst final results. The genetic algorithms achieve an excellent result in much shorter time than the others. It is to be noticed that, in all the

previous experiments, the GA has performed better than manual configuration and random search. Thus, only the manual configuration and the GAs have been tested on the other datasets. The results of PSO are comparable, but the computation time required to perform the same number of fitness computations as the GA is longer.

7.5.2 Results on Paris6k

Method	Fitness comp.	Time	mAP
Manual configuration [62]	1	4 s	97.01%
Random search [86]	5000	28916 s	97.29%
Random search [86]	10000	58708 s	97.25%
Grid search	5000	33067 s	97.26%
Grid search	10000	59777 s	97.26%
PSO [87]	5000	67071 s	97.31%
Genetic algorithms	**5000**	**18787 s**	**97.32%**

Table 7.20: Comparison of different approaches for the optimization of the diffusion parameters on Paris6k.

Table 7.20 reports the results of different optimization methods applied on Paris6k. The best result (97.32%) has been obtained with the following GA configuration: $Gen = 100$, $Pop = 50$, $CxPb = 0.5$, $MutPb = 0.2$, $IndPb = 0.1$. The final configuration of the diffusion parameters is: $\alpha = 0.87$, $\beta = 1$, $\gamma = 2$, $k_s = 40$, $k = 11$, $iterations = 10$, $trunc = 3761$.

As in the previous dataset, the GAs need more computation time than the “manual configuration", but they improve the final performance of the diffusion process for retrieval.

7.5.3 Results on Oxford105k

Given the large dimension of Oxford105k dataset, the ranges of parameters k_s and k have been extended to $\{20,250\}$ (int) and $\{5,100\}$ (int), respectively.

Table 7.21 reports the results of different optimization methods applied on Oxford105k.

Method	Fitness comp.	Time	mAP
Manual configuration [62]	1	13 s	92.50%
Random search [86]	5000	90780 s	93.65%
Random search [86]	10000	197501 s	93.70%
Grid search	5000	91243 s	93.85%
Grid search	10000	201546 s	94.10%
PSO [87]	5000	108178 s	94.20%
Genetic algorithms	**5000**	**63911** s	**94.20%**

Table 7.21: Comparison of different approaches for the optimization of the diffusion parameters on Oxford105k.

The best result (94.20%) is obtained with the following GA configuration: $Gen = 100$, $Pop = 50$, $CxPb = 0.5$, $MutPb = 0.2$, $IndPb = 0.1$. The final configuration of the diffusion parameters is: $\alpha = 0.97$, $\beta = 2$, $\gamma = 1$, $k_s = 68$, $k = 7$, $iterations = 10$, $trunc = 18353$.

The “manual configuration" is faster than the GAs, but the final performance is very different: the GAs obtain 94.20% while the "manual configuration" achieves only 92.50%.

Chapter 8

Conclusions

I've missed more than 9000 shots in my career. I've lost almost 300 games. 26 times, I've been trusted to take the game winning shot and missed. I've failed over and over and over again in my life. And that is why I succeed.

Michael Jordan

This thesis presents the approaches and the methods applied for the resolution of the Content-Based Image Retrieval problem.

Firstly, the focus is on the classical approaches for solving this problem: the most famous are Bag of Words and VLAD descriptors.

Secondly, it is proved that the deep learning techniques outperform the results obtained by the hand-crafted methods like SIFT and SURF. On this way, I proposed a new Dense-Depth Representation (DDR) that allows, combined with locVLAD descriptors, to outperform the state of the art related to VLAD descriptors on several public image datasets. The combination of the dense representation (DDR) and the focus on the most important part of the images of locVLAD allowed to obtain a better representation of each image and, therefore, to improve the retrieval accuracy without the need to augment the dimension of the descriptor that still remains $512D$.

I also take advantages of new strategies as the R-MAC embedding and the fine-tuning of the networks. On this way, I implemented several improvements on R-MAC

descriptors with the objective to make the retrieval very accurate. The R-MAC+ pipeline exploits a new multi-resolution approach that improves the performance thanks to the use of bigger feature maps. The R-MAC+ region detector uses adaptable grids allowing to detect more local maxima, that are the relevant patches for the MAC descriptors. The last improvement in the R-MAC+ pipeline is the new retrieval method based on "db regions" that highly boosts the performance on Oxford5k and Paris6k. The proposed method outperforms the state of the art on Holidays, both on the original and rotated version. Also it outperforms the state-of-the-art results on some other public benchmarks without the fine-tuning application. The proposed strategies (both for the descriptors and the retrieval methods) are general enough to be applicable to the image retrieval tasks.

Thirdly, the thesis talks about large-scale retrieval. This is now the biggest challenge in CBIR and future works should focus on achieving good accuracy in datasets that will get bigger, while maintaining small retrieval time and memory occupancy. I proposed a novel multi-index hashing methods called Bag of Indexes (BoI) for approximate nearest neighbour search problem. The method demonstrates an overall better trade-off between accuracy and speed with reference to the state-of-the-art methods on several large-scale CBIR datasets. Also, it works well with different embedding types (VLAD, R-MAC and R-MAC+).

Fourthly, I presented a method called LSH kNN graph for the creation of an approximate kNN graph exploiting LSH projections. This method allows to solve one of two problems due to the diffusion application. Diffusion outperforms the retrieval performance on several public benchmarks, but the creation of the kNN graph requires a lot of time and computational resources. The proposed approach can create a kNN graph faster than other state-of-the-art approaches, obtaining the same or better accuracy results after diffusion application. Furthermore, an optimized version of the previous algorithm called multi LSH kNN graph is proposed, which exploits the main principle of multi-probe LSH, improving the quality of the final graph due to the greater number of elements found in the buckets of the hash tables.

Fiftly, I proposed to use genetic algorithms in order to solve the second major problem of the diffusion application. I opted to genetic algorithms for the diffusion

parameters optimization, that usually requires an huge number of experiments because the parameters are several and the value range of them is large. The application of genetic algorithms allows to find a better set of parameters, the results will be more precise. Compared to my method, the other techniques: random search, grid search and PSO are slower and they obtain the same or worst retrieval results. Unfortunately, it is not possible to find a set of parameters that works well with different image datasets.

In the end, the image retrieval task is far being solved. Problems as change of illuminations, point of views, presence of distractors, scale and rotation variations have not completely solved in image retrieval. It is worth to note that the current used images are challenging, but the images that can be taken in the real world at runtime can introduce more complexity. The CBIR pipelines are not yet tested on real life scenarios.

The use of deep learning for feature detection and description highly improved the performance. Modern algorithms for global descriptors creation executed in deep learning architectures allow to boost the final retrieval results. In the end, re-ranking techniques as diffusion applied on graphs achieve very outstanding results.

On the other hand, the current hardware available on smartphones and mobile devices do not allow to implement and execute all the mentioned CBIR algorithms. The future target of CBIR will be the application and execution on mobile devices without any loss in performance because the CBIR pipelines will be very useful to adopt in several fields: from automatic landmark location to tourist guides.

Bibliography

[1] Ying Liu, Dengsheng Zhang, Guojun Lu, and Wei-Ying Ma. A survey of content-based image retrieval with high-level semantics. *Pattern Recognition*, 40(1):262–282, 2007.

[2] Jonathon S Hare, Paul H Lewis, Peter GB Enser, and Christine J Sandom. Mind the gap: Another look at the problem of the semantic gap in image retrieval. In *Multimedia Content Analysis, Management, and Retrieval 2006*, volume 6073, page 607309. International Society for Optics and Photonics, 2006.

[3] Paisarn Muneesawang, Ning Zhang, and Ling Guan. Mobile landmark recognition. In *Multimedia Database Retrieval*, pages 131–145. Springer, 2014.

[4] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[5] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[6] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ILSVRC 2015*, 2014.

[7] Liang Zheng, Yi Yang, and Qi Tian. Sift meets cnn: A decade survey of instance retrieval. *IEEE transactions on pattern analysis and machine intelligence*, 2017.

[8] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al. Going deeper with convolutions. IEEE Conference on Computer Vision and Pattern Recognition, 2015.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[10] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. *International Conference on Computer Vision*, 2:1470–1477, 2003.

[11] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *European conference on computer vision*, pages 304–317. Springer, 2008.

[12] Jégou Hervé, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3304–3311, 2010.

[13] Florent Perronnin, Yan Liu, Jorge Sánchez, and Hervé Poirier. Large-scale image retrieval with compressed fisher vectors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3384–3391. IEEE, 2010.

[14] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. On the burstiness of visual elements. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1169–1176. IEEE, 2009.

[15] Giorgos Tolias, Yannis Avrithis, and Hervé Jégou. To aggregate or not to aggregate: Selective match kernels for image search. In *IEEE International Conference on Computer Vision*, pages 1401–1408, 2013.

[16] Hervé Jégou and Ondřej Chum. Negative evidences and co-occurences in image retrieval: The benefit of PCA and whitening. *European Conference on Computer Vision*, pages 774–787, 2012.

[17] Relja Arandjelovic and Andrew Zisserman. All about VLAD. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 1578–1585, 2013.

[18] Zixuan Wang, Wei Di, Anurag Bhardwaj, Vignesh Jagadeesh, and Robinson Piramuthu. Geometric VLAD for large scale image search. *International Conference on Machine Learning*, 2014.

[19] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. *European Conference on Computer Vision*, pages 143–156, 2010.

[20] Wan-Lei Zhao, Hervé Jégou, and Guillaume Gravier. Oriented pooling for dense and non-dense rotation-invariant features. *British Machine Vision Conference*, 2013.

[21] Qiuzhan Zhou, Cheng Wang, Pingping Liu, Qingliang Li, Yeran Wang, and Shuozhang Chen. Distribution entropy boosted VLAD for Image Retrieval. *Entropy*, 2016.

[22] Christian Eggert, Stefan Romberg, and Rainer Lienhart. Improving VLAD: hierarchical coding and a refined local coordinate system. *International Conference on Image Processing*, 2014.

[23] Ziqiong Liu, Shengjin Wang, and Qi Tian. Fine-residual VLAD for image retrieval. *Neurocomputing*, 173:1183–1191, 2016.

[24] Federico Magliani, Navid Mahmoudian Bidgoli, and Andrea Prati. A location-aware embedding technique for accurate landmark recognition. *International Conference on Distributed Smart Cameras*, 2017.

[25] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5297–5307, 2016.

[26] Joe Yue-Hei Ng, Fan Yang, and Larry S Davis. Exploiting local features from deep networks for image retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 53–61, 2015.

[27] Ke Yan, Yaowei Wang, Dawei Liang, Tiejun Huang, and Yonghong Tian. Cnn vs. sift for image retrieval: alternative or complementary? In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 407–411. ACM, 2016.

[28] Konda Reddy Mopuri and R Venkatesh Babu. Object level deep feature pooling for compact image representation. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 62–70, 2015.

[29] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. Particular object retrieval with integral max-pooling of CNN activations. *International Conference on Learning Representation*, 2015.

[30] Albert Gordo, Jon Almazán, Jerome Revaud, and Diane Larlus. Deep image retrieval: Learning global representations for image search. In *European Conference on Computer Vision*, pages 241–257. Springer, 2016.

[31] Omar Seddati, Stéphane Dupont, Saıd Mahmoudi, Mahnaz Parian, and Bd Dolez. Towards good practices for image retrieval based on cnn features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1246–1255, 2017.

[32] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[33] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. Neural codes for image retrieval. In *European conference on computer vision*, pages 584–599. Springer, 2014.

[34] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *International Conference on Computational Statistics*, pages 177–186. Springer, 2010.

[35] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.

[36] Albert Gordo, Jon Almazan, Jerome Revaud, and Diane Larlus. End-to-end learning of deep visual representations for image retrieval. *International Journal of Computer Vision*, 124(2):237–254, 2017.

[37] Federico Magliani and Andrea Prati. An accurate retrieval through R-MAC+ descriptors for landmark recognition. In *Proceedings of the 12th International Conference on Distributed Smart Cameras*, page 6. ACM, 2018.

[38] Relja Arandjelović and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2911–2918. IEEE, 2012.

[39] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.

[40] Andrea Esuli. Use of permutation prefixes for efficient and scalable approximate similarity search. *Information Processing & Management*, 48(5):889–902, 2012.

[41] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

[42] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Symposium on Computational Geometry*, pages 253–262. ACM, 2004.

[43] Qin Lv, William Josephson, Zhe Wang, Moses Charikar, and Kai Li. Multi-probe LSH: efficient indexing for high-dimensional similarity search. In *Proceedings of the 33rd international conference on Very Large Data Bases*, pages 950–961. VLDB Endowment, 2007.

[44] Jingdong Wang, Heng Tao Shen, Jingkuan Song, and Jianqiu Ji. Hashing for similarity search: A survey. *arXiv preprint arXiv:1408.2927*, 2014.

[45] Federico Magliani, Tomaso Fontanini, and Andrea Prati. Efficient nearest neighbors search for large-scale landmark recognition. In *International Symposium on Visual Computing*, pages 541–551. Springer, 2018.

[46] Ahmet Iscen, Giorgos Tolias, Yannis S Avrithis, Teddy Furon, and Ondrej Chum. Efficient diffusion on region manifolds: Recovering small objects with compact CNN representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, page 3, 2017.

[47] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Mining on manifolds: Metric learning without labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7642–7651, 2018.

[48] Matthijs Douze, Arthur Szlam, Bharath Hariharan, and Hervé Jégou. Low-shot learning with large-scale diffusion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3349–3358, 2018.

[49] Jian Xu, Chunheng Wang, Chengzuo Qi, Cunzhao Shi, and Baihua Xiao. Iterative manifold embedding layer learned by incomplete data for large-scale image retrieval. *IEEE Transactions on Multimedia*, 2018.

[50] Dong Li, Wei-Chih Hung, Jia-Bin Huang, Shengjin Wang, Narendra Ahuja, and Ming-Hsuan Yang. Unsupervised visual representation learning by graph-based consistent constraints. In *European Conference on Computer Vision*, pages 678–694. Springer, 2016.

[51] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[52] Wei Dong, Charikar Moses, and Kai Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th International Conference on World Wide Web*, pages 577–586. ACM, 2011.

[53] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the 18th annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[54] Yan-Ming Zhang, Kaizhu Huang, Guanggang Geng, and Cheng-Lin Liu. Fast kNN graph construction with locality sensitive hashing. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 660–674. Springer, 2013.

[55] Jie Chen, Haw-ren Fang, and Yousef Saad. Fast approximate kNN graph construction for high dimensional data via recursive lanczos bisection. *Journal of Machine Learning Research*, 10(Sep):1989–2012, 2009.

[56] Jing Wang, Jingdong Wang, Gang Zeng, Zhuowen Tu, Rui Gan, and Shipeng Li. Scalable k-nn graph construction for visual descriptors. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1106–1113. IEEE, 2012.

[57] Youngki Park, Sungchan Park, Sang-goo Lee, and Woosung Jung. Scalable k-nearest neighbor graph construction based on greedy filtering. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 227–228. ACM, 2013.

[58] Michael E Houle, Xiguo Ma, Vincent Oria, and Jichao Sun. Improving the quality of K-NN graphs for image databases through vector sparsification. In *Proceedings of International Conference on Multimedia Retrieval*, page 89. ACM, 2014.

[59] Thibault Debatty, Pietro Michiardi, Olivier Thonnard, and Wim Mees. Building k-nn graphs from large text data. In *IEEE International Conference on Big Data*, pages 573–578. IEEE, 2014.

[60] Sami Sieranoja and Pasi Fränti. Fast random pair divisive construction of knn graph using generic distance measures. In *Proceedings of the 2018 International Conference on Big Data and Computing*, pages 95–98. ACM, 2018.

[61] Denny Zhou, Jason Weston, Arthur Gretton, Olivier Bousquet, and Bernhard Schölkopf. Ranking on data manifolds. In *Advances in Neural Information Processing Systems*, pages 169–176, 2004.

[62] Federico Magliani, Kevin McGuiness, Eva Mohedano, and Andrea Prati. An efficient approximate kNN graph method for diffusion on image retrieval. *International Conference on Image Analysis and Processing*, 2019.

[63] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[64] RV Mises and Hilda Pollaczek-Geiringer. Praktische verfahren der gleichungsauflösung. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 9(2):152–164, 1929.

[65] Federico Magliani, Laura Sani, Stefano Cagnoni, and Andrea Prati. Genetic algorithms for the optimization of diffusion parameters in content-based image retrieval. *ACM Proceedings of the 13th International Conference on Distributed Smart Cameras*, 2019.

[66] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. Deap: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13(Jul):2171–2175, 2012.

[67] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[68] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[69] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 2161–2168. Ieee, 2006.

[70] Mark J Huiskes and Michael S Lew. The MIR flickr retrieval evaluation. In *Proceedings of the 1st ACM international conference on Multimedia Information Retrieval*, pages 39–43. ACM, 2008.

[71] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[72] Ali S Razavian, Josephine Sullivan, Stefan Carlsson, and Atsuto Maki. Visual instance retrieval with deep convolutional networks. *ITE Transactions on Media Technology and Applications*, 4(3):251–258, 2016.

[73] Yunchao Gong, Liwei Wang, Ruiqi Guo, and Svetlana Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *European conference on computer vision*, pages 392–407. Springer, 2014.

[74] Ondrej Chum, James Philbin, Josef Sivic, Michael Isard, and Andrew Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[75] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3304–3311. IEEE, 2010.

[76] Joe Yue-Hei Ng, Fan Yang, and Larry S Davis. Exploiting local features from deep networks for image retrieval. pages 53–61, 2015.

[77] Artem Babenko and Victor Lempitsky. Aggregating local deep features for image retrieval. In *Proceedings of the IEEE international conference on computer vision*, pages 1269–1277, 2015.

[78] Yannis Kalantidis, Clayton Mellina, and Simon Osindero. Cross-dimensional weighting for aggregated deep convolutional features. In *European Conference on Computer Vision*, pages 685–701. Springer, 2016.

[79] Konda Reddy Mopuri and R Venkatesh Babu. Object level deep feature pooling for compact image representation. In *CVPR Workshops*, pages 62–70, 2015.

[80] Eva Mohedano, Kevin McGuinness, Noel E O'Connor, Amaia Salvador, Ferran Marques, and Xavier Giro-i Nieto. Bags of local convolutional features for scalable instance search. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, pages 327–331. ACM, 2016.

[81] Eva Mohedano, Kevin McGuinness, Xavier Giro-i Nieto, and Noel E O'Connor. Saliency weighted convolutional features for instance search. pages 1–6, 2018.

[82] Zakaria Laskar and Juho Kannala. Context aware query image representation for particular object retrieval. In *Scandinavian Conference on Image Analysis*, pages 88–99. Springer, 2017.

[83] Hossein Azizpour, Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. Factors of transferability for a generic convnet representation. *IEEE transactions on pattern analysis and machine intelligence*, 38(9):1790–1802, 2016.

[84] Yannis Kalantidis and Yannis Avrithis. Locally optimized product quantization for approximate nearest neighbor search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2321–2328, 2014.

[85] Marius Muja and David G Lowe. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2227–2240, 2014.

[86] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[87] Riccardo Poli, James Kennedy, and Tim Blackwell. Particle swarm optimization. *Swarm intelligence*, 1(1):33–57, 2007.

Acknowledgments

First and foremost I want to thank my advisor, the Full Professor Andrea Prati. It has been an honor to be one of his Ph.D. student. I appreciate all his contributions of time, ideas, and funding to make my Ph.D. experience productive and stimulating. The joy and enthusiasm he has for his research was contagious and motivational for me, even during tough times in the Ph.D. pursuit. I am also thankful for the excellent example he has provided as a successful engineer and professor. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D. study.

The members of the IMP lab group have contributed immensely to my personal and professional time at University of Parma. The group has been a source of friendships as well as good advice and collaboration. I am especially grateful to Tomaso Fontanini who stuck it out in grad school with me. I would like to acknowledge honorary group member Luca Donati who teaches me how to improve my coding skills in C++. We worked together on several projects, and I very much appreciated his enthusiasm, intensity and cleverness. In the end, a special thanks to the other IMP lab group members: Eleonora Iotti, Leonardo Rossi and Akbar Karimi.

My sincere thanks also goes to Assistant Professor Kevin McGuinness, Dr. Eva Mohedano, and Prof. Noel O'Connor, who provided me an opportunity to join their team as intern in the Insight Centre for Data Analytics, as a lab of DCU in Ireland, and who gave access to the laboratory and research facilities. Without they precious support it would not be possible to conduct this research and improve my soft skills.

In the end, it is worth to note that this is work is partially funded by Regione

Emilia Romagna under the “Piano triennale alte competenze per la ricerca, il trasferimento tecnologico e l’imprenditorialità".