Deep Image Representations for Instance Search

Eva Mohedano Robles

Bachelor in Electronic Engineering

A Dissertation submitted in fulfilment of the

requirements for the award of

Doctor of Philosophy (Ph.D.)

to the

Dublin City University

School of Electronic Engineering

Supervisor: Dr. Kevin McGuinness and Prof. Noel E. O'Connor

September, 2017

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: ________________ ID No: 12122581

Date: ________________

Acknowledgments

Finally, we did it! And I say "we" because writing this thesis would not have been possible without the support of many people that has been involved at some level in helping me to go thought this long and intense journey.

Firstly, I would like to express my sincere gratitude to my supervisors Noel O'Connor, Kevin McGuinness, and my unofficial supervisor Xavier Giró. Thanks Noel for giving me the opportunity of doing this Ph.D, and for you continuous support and advise. Thanks Kevin for always having "5 minutes" (or 1 hour, or whatever it takes), and guiding me through the deep learning universe. You are a source of infinite knowledge and inspiration. And thank you Xavi for all these years sharing your knowledge and enthusiasm about research, *em sento molt molt afortunada d'haver-te conegut*.

I would like to thank Amaia, my "Ph.D buddy in the distance", for her constant support during this adventure. I promise I'll go to visit you wherever you decide, *de veritat*! Thanks a lot to Graham, for guiding me in the EEG world at the beginning, and sharing your enthusiasm for research. Thanks to the cubic guys: Joseph, Dian, Iveel, and the cubic sponsor Kevin J. Fraser. It was a great fun working with you in the cubic! Thanks a lot also to my colleagues and friends in Insight, specially to Camille, Diego, Eric, and Mark.

Also, I would like to thank the incredible people that Dublin has brought me. Thanks a lot to Melina, Abel, Alex, Zeliha, Giusy, Hari, Marina, and Gaby. And specially thanks to Julio, for your huge support, patience, and love during this last year.

And last but not least, I would like to thank my family because I would never ever have been able to do this without their support. Thanks to my parents José Joaquín and Gloria, my sister Gloria, my brother José, my grandma Gloria, and mi little and lovely nephew and niece Martí and Gala. This is for and thanks to all of you. I love you.

Contents

List of publications

- **Eva Mohedano**, Amaia Salvador, Kevin McGuinness, Xavier Giró-i-Nieto, Noel O'Connor, Ferran Marqués. Bags of Local Convolutional Features for Scalable Instance Search. In *ACM International Conference on Multimedia Retrieval (ICMR).* United States, June 2016. doi: 10.1145/2911996.2912061

- Mark Marsden, **Eva Mohedano**, Kevin McGuinness, Andrea Calafell, Xavier Giro-i-Nieto, Noel O'Connor, Jiang Zhou, Lucas Azevedo, Tobias Daudert, Brian Davis, Manuela Hurlimann, Haithem Afli, Jinhua Du, Debasis Ganguly, Wei Li, Andy Way, Alan Smeaton. Dublin CIty University and Partners' Participation in the INS and VTT Tracks at TRECVid 2016. In *TRECVid 2016 Workshop*, United States, Nov 2016.

- Cristian Reyes, **Eva Mohedano**, Kevin McGuinness, Noel O'Connor, Xavier Giró-i-Nieto. Where is my Phone?: Personal object retrieval from egocentric images. In *ACM International Conference on Multimedia Retrieval (ICMR).* Netherlands, October 2016. doi:10.1145/2983576.2983582

- **Eva Mohedano**, Graham Healy, Kevin McGuinness, Xavier Giró-i-Nieto, Noel O'Connor, Alan Smeaton. Improving Object Segmentation by using EEG signals and Rapid Serial Visual Presentation. In *Multimedia Tools and Applications.* Aug. 2015. doi: 10.1007/s11042-015-2805-0

- **Eva Mohedano**, Amaia Salvador, Sergi Porta, Xavier Giró-i-Nieto, Kevin McGuinness, Graham Healy, Noel O'Connor, Alan Smeaton. Exploring EEG for

Object Detection and Retrieval. In *The Annual ACM International Conference on Multimedia Retrieval (ICMR)*. China, Jul 2015.

- Kevin McGuinness, **Eva Mohedano**, Amaia Salvador, Zhenxing Zhang, Mark Marsden, Peng Wang, Iveel Jargalsaikhan, Joseph Antony, Xavier Giro-i-Nieto, Shinichi Satoh, Noel E. OConnor, Alan Smeaton. Insight dcu at trecvid 2015. In *TRECVid 2015 Workshop*, United States, Nov 2015.
- Kevin McGuinness, **Eva Mohedano**, ZhenXing Zhang, Feiyan Hu, Rami Albatal, Cathal Gurrin, Noel O'Connor, Alan Smeaton, Amaia Salvador, Xavier Gir-i-Nieto, Carles Ventura. Insight Centre for Data Analytics (DCU) at TRECVid 2014: Instance Search and Semantic Indexing Tasks. In *TRECVid 2014 Workshop*, United States, Nov 2014.
- **Eva Mohedano**, Graham Healy, Kevin McGuinness, Xavier Giró-i-Nieto, Noel O'Connor, Alan Smeaton. Object Segmentation in Images using EEG Signals. In *ACM Multimedia 2014 , The 22nd ACM International Conference on Multimedia.* United States, Nov 2014. doi: 10.1145/2647868.2654896.

List of Tables

List of Figures

Abstract

Deep Image Representations for Instance Search

Eva Mohedano

We address the problem of visual instance search, which consists to retrieve all the images within an dataset that contain a particular visual example provided to the system. The traditional approach of processing the image content for this task relied on extracting local low-level information within images that was "manually engineered" to be invariant to different image conditions. One of the most popular approaches uses the Bag of Visual Words (BoW) model on the local features to aggregate the local information into a single representation. Usually, a final re-ranking stage is included in the pipeline to refine the search results. Since the emergence of deep learning as the dominant technique in computer vision in 2012, much research attention has been focused on deriving image representations from Convolutional Neural Networks (CNN) models for the task of instance search as a "data driven" approach to designing image representations. However, one of the main challenges in the instance search task is the lack of annotated datasets to fit CNN models parameters.

This work explores the capabilities of descriptors derived from pre-trained CNN models for image classification to address the task of instance retrieval. First, we conduct an investigation of the traditional bag of visual words encoding on local CNN features to produce a scalable image retrieval framework that generalizes well across different retrieval domains. Second, we propose to improve the capacity of the obtained representations by exploring an unsupervised fine-tuning strategy that allow us to obtain better performing representations at the price of losing the generalization of the representations. Finally, we propose using visual attention models to weight the contribution of the relevant parts of an image to obtain a very powerful image representation for instance retrieval without requiring the construction of a large and suitable training dataset for fine-tuning CNN architectures.

Chapter 1

Introduction

With the proliferation of mobile devices, millions of images and thousands hours of video content are uploaded every day to the internet without explicit annotation of their content. Content based image retrieval (CBIR) systems make use of image representations, which are a numerical representation of the image content, to address this task. The challenge is to design algorithms that discriminatingly represent the image content while keeping a low memory requirement and fast retrieval times. The purpose of this chapter is to provide an introduction to content based image retrieval and the instance search problem. We present motivations for the thesis and the hypotheses and research questions that we address during this work.

1.1 Content based image retrieval (CBIR)

Before the emergence of content based image retrieval (CBIR) [117] as a research field in the early '90s, the task of searching images within an image database consisted of creating text annotations that described the content of the images. The images themselves were not really part of the database; they were only referred to by text strings or pointers. The user could then search for a particular image by entering a text description or tags, to retrieve all images annotated with the same description. Some representative text-based image retrieval systems and surveys can be found

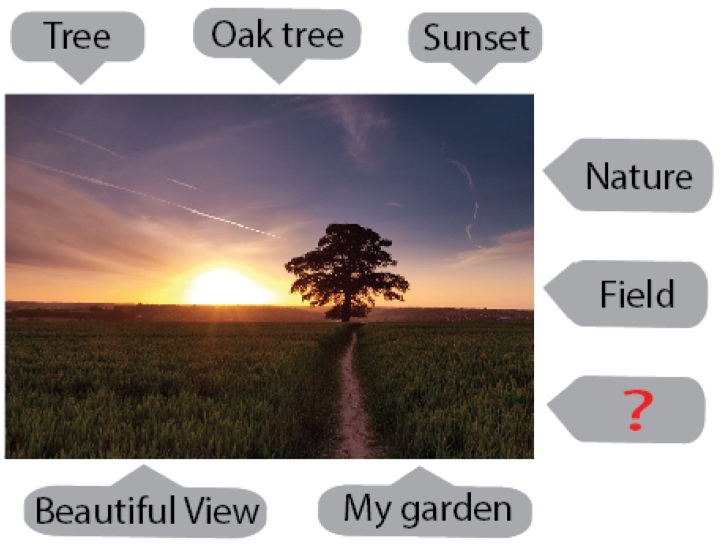


Figure 1.1: Illustration of the subjectivity of the human textual annotation. A variety of textual tags that can be associated to a particular image. The subjectivity on the textual annotations may cause unrecoverable mismatches in later retrieval in text-based image retrieval systems.

in [19, 21, 20, 122].

This approach presented two major issues. The first is that it involves the manual annotation of the image dataset, a labourious and tedious task, which nowadays with the volume of digital media created becomes impractical. Just to give an idea, more than 20,000 photos are shared every second in Snapchat, it would take about 10 years to view all the photos shared in the last hour [1]. 196 million images are uploaded every day to Facebook, 80 million photos are uploaded per day to Instagram and about 220 hours of video are uploaded per hour to Youtube [2]. This is just considering some of the most popular social-media websites: surveillance cameras, pictures or videos capturing by drones, or personal photo collections are not taken into account in those statistics. Given the volume of images generated nowadays it is impractical to manually and exhaustively generate an accurate annotation of the image content.

The second major and even more important issue of a text-based image retrieval system is that it relies solely on a textual annotation of an image. A single image

[1] https://www.omnicoreagency.com/snapchat-statistics/
[2] http://www.smartinsights.com/internet-marketing-statistics/happens-online-60-seconds/

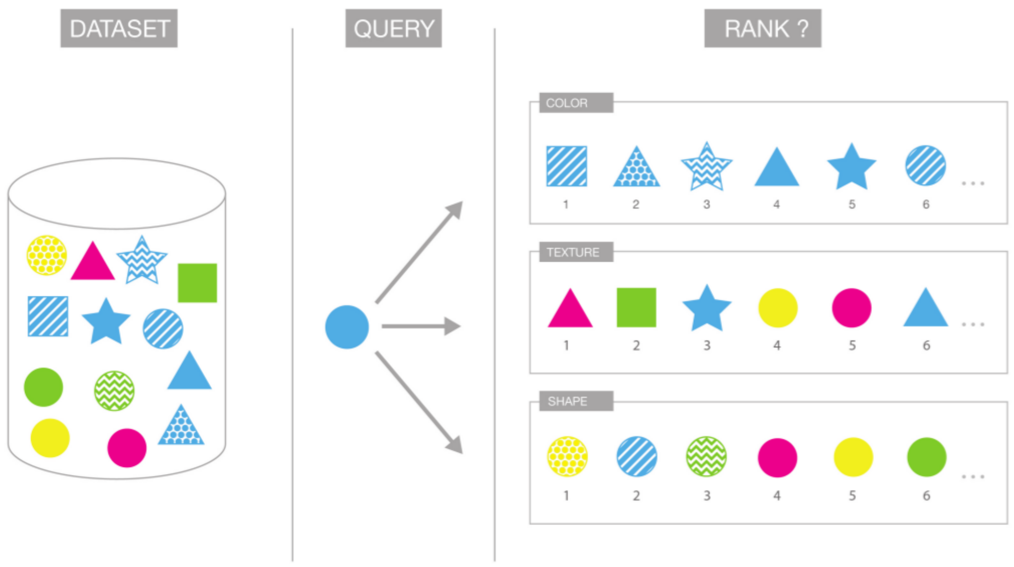


Figure 1.2: Different ways of sorting a dataset items based on the relevance to a particular query. (best view in color)

can be described in multiple ways due to the subjectivity of human perception. For example, the image in the Figure 1.1 can be associated to the tag *tree*, but also we could have provided a more accurate tag of *oak tree*. Instead of focusing on the tree, we could have described the image as a *sunset* or we could have provided a more abstract *peaceful view* or *beautiful view* tag regarding the kind of sentiments that the image invokes. We could have just described it as *field* or *nature* or, if we are lucky and we have a house with those views, we could have associated the tag *my garden* or a combination of some of the mentioned tags. This subjectivity of textual annotations may cause unrecoverable mismatches in later retrieval processes.

While textual annotation might change depending on the person generating the descriptions, the content of the image (pixels) represent a more objective description of the image content. CBIR aims then at organizing and structuring datasets of images based on their content rather than the associated metadata. The images are described using computer vision algorithms that “summarize” the content of the image, capturing information about the colours, shapes and/or textures derived from the pixels in the form of a numerical vector, also called an image representation.

The user's request is given in the form of a query image and the system produces a list of images that are relevant to the query, ordered by a similarity score. The notion of similarity is application dependent. Figure 1.2 illustrates this idea: given a query image (represented with a blue plain circle), the images in the database can be sorted based on the colour, the texture, or the shape. There is no "correct" way of sorting the images in a database, and thus the notion of similarity depends on the final application. One CBIR system can be focused on comparing low-level image characteristics between images such as colours or different textures, while another might focus on retrieving images semantically similar to the query example, taking into account of the meaning of the actions and objects contained in the images [68].

1.2 Instance search

CBIR has been tackled mostly as the problem of instance-level image retrieval [96, 56, 54, 9, 39]. The notion of an "instance" specifically limits the search to "one" instance of a semantic class (i.e to retrieve instances of one specific object, person or location as in [6]). This contrasts with less specific retrieval pipelines in which any instance of any member of a class will satisfy the search. For example an instance search system would focus on retrieving the specific instance of the the fictional TV show character Lassie dog instead of retrieving instances of dogs in general.

One of the main challenges of the instance search task is the lack of annotated data to fit models to recognize a particular query example. Since the query image is unknown, images should be described and indexed in a way such that, for example, it is possible to perform a search of a particular building but also of a particular person, logo, or item of clothing. This is particularly useful in real world scenarios such as image search engines on the web, where the query instance is unknown, or for organization of personal photo collections.

The method of processing the image content for image retrieval applications that dominated during the years 2003 to 2012 [144] relied on extracting local low-level information within the images, specifically designed to be invariant to image scale and rotation as well as to be robust to affine/perpective distortions, viewpoints change, noise, illumination changes, background clutter or occlusions. So, a particular instance could be recognized within an image and retrieved irrespective of all these possible variations. Local descriptors were usually aggregated into a high dimensional and sparse representation, allowing the construction of inverted files structures for fast retrieval. A second step verifying the geometric consistency between matched images was performed to further improve the performance of the results [114, 96, 3, 27, 6].

Since the emergence of deep learning as the dominant technique in computer vision, particularly the use of Convolutional Neural Networks models [64], several works have proposed using representations derived from CNN models pre-trained for classification with a large dataset of images [9, 8, 125]. Many of these works use a pre-trained CNN as a local feature extractor, from which features are typically aggregated by direct sum/max pooling within convolutional layers [8**?** , 125] to generate compact image representations. Some other recent works re-train the CNN models optimizing for a task more related to the instance search, achieving a new state-of-the art in different retrieval benchmarks [39, 100].

However, most of the approaches evaluate their performance in relatively “simple” retrieval scenarios, mostly landmark or scene related, where the diversity in the query domain as well as the relative position and sizes of the query instances are limited.

1.3 Motivation

While several different works have shown the capability of CNN representations for the task of instance retrieval, it is still unclear whether the performance of these systems generalizes well in more realistic and generic scenarios. The purpose of this thesis is to explore CNN models to construct an image representation suitable for the instance search task that is not subject to any particular domain or type of instance, providing an efficient system that is scalable to large and diverse datasets.

1.4 Hypotheses

The hypotheses of this work can be stated as follows:

H1) **When building a global image representation from local CNN image descriptors for retrieval, the aggregation of those descriptors into a high dimensional sparse representations is "better" (in terms of performance and efficiency of the representation) than aggregating them within the original local feature space.**

We expect that the high dimensionality of global representation reduces the chances that local features interfere during the aggregation step (in contrast to performing direct pooling on the original local feature space [8**?**]). Also, the sparsity allows the construction of inverted file structures making the search more efficient. This scheme implemented with bag of visual words encoding of hand-crafted local features has been widely explored and has represented the core of many image retrieval systems since it was first proposed for image retrieval by Sivic and Zisserman [114].

H2) **Re-training a pre-trained CNN for the task of instance retrieval is an appropriate procedure to build specialized but not generic instance search systems.** When a training dataset is available, recent works have shown that fine-tuning CNN models for the retrieval task generates feature representations that outperform the ones extracted from a pre-trained classification model [9, 1, 39, 100]. However, their performance in more realistic and generic databases remains unclear.

H3) **Visual attention models are useful for the task of generic instance retrieval**.

Spatial weighting schemes on local convolutional features have been shown to be effective and beneficial in some retrieval benchmarks [8**?**]. Also, generating a set of region CNN representations has been shown to be beneficial for the instance search task, as a mechanism for detecting the most relevant parts of the image to facilitate the search of a particular instance [110, 125, 39].

Visual attention models [13] mimic the capability of humans to focus cognitive processing onto a subset of the environment. We hypothesize that attention models can be useful and beneficial in the the construction of CNN visual representations for retrieval.

1.4.1 Research questions

To address the first hypothesis, regarding the *aggregation of local CNN features into a high-dimensional sparse representation*, we propose the following research questions:

H1-*Q1* Is it possible to use the traditional BoW encoding on off-the-shelf local representations to generate high dimensional and sparse representations from local CNN representations to address the task of instance search?

H1-*Q2* Is BoW-CNN better (in terms of performance and efficiency) than direct pooling techniques as used in the literature on local CNN descriptors?

H1-*Q3* Is BoW-CNN a solution that generalizes well across different retrieval benchmarks?

To address the second hypothesis, regarding *improving off-the-shelf CNN networks for the task of retrieval*, we propose the following research questions:

H2-*Q4*) Is it possible to perform similarity learning in a particular dataset without requiring additional data and expensive manual annotations?

H2-*Q5*) How does a fine-tuned model trained in a particular domain perform in a different (domain related or not) dataset?

Finally, regarding the third hypothesis of *using visual attention models to improve CNN image representations for retrieval*, we propose the following research question:

H3-*Q6* Are saliency models a good mechanism to weight the contribution of the visual words/features derived from local CNN networks for the task of instance search?

1.5 Thesis outline

The remainder of this thesis is structured as follows. Chapter 2 provides the technical background for the thesis. We explain traditional approaches to generate image representations. We introduce the concept of deep learning, describing convolutional neural network models. We define the metric used for evaluating retrieval performance. Lastly, we introduce commonly used retrieval benchmarks, mostly domain dependent, and two specifically designed for generic instance retrieval as explored in this thesis.

Chapter 3 provides an exhaustive analysis of the existing work using CNN representations for the instance search task. We start by describing the first approaches that use the very top layers from a CNN as a global representation moving on to the most recent approaches exploring the local information contained in convolutional layers. We describe different ways to aggregate, weight and derive region or patch level descriptors from convolutional layers. We also discuss different works that specifically train a CNN architecture for the task of retrieval, and different approaches for constructing the training dataset.

Chapter 4 revisits the traditional bag-of-visual-words (BoW) encoding as a new aggregation method for local CNN representations. This aggregation method benefits from high dimensionality and sparse representations to generate more discriminative features that outperform other methods in challenging retrieval benchmarks. We also apply spatial re-ranking techniques, taking benefit of the compact assignment maps obtained to boost performance of the baseline system.

Since the proposed aggregation step based on BoW is an independent step when building the final image representation, in Chapter 5 we develop a fine-tuning strategy to adapt the features and encoding for the retrieval task. For this, we propose a method to perform fine-tuning within a dataset without requiring any additional labeled images. We investigate the effect of fine-tuning in different domains, finding that fine-tuned CNN models for similarity learning is a good strategy for performing instance search in a particular image domain, at the price of losing the generality of the original off-the-shelf representations.

With the goal of finding a pipeline that generalizes in different instance search domains, Chapter 6 investigates saliency models in combination with off-the-shelf CNN representations to weight the contribution of each local CNN representation. We obtain a system that achieves state-of-the-art performance regardless of the domain of the instances to be retrieved, or specifically when we apply BoW for aggregating the local representations. Finally, Chapter 7 presents the conclusions as well as opportunities for future research.

Chapter 2

Background

2.1 Introduction

The aim of this chapter is to provide the technical background for the research reported in this thesis. In particular, Section 2.2 describes traditional approaches, where we describe scale-invariance image transforms and different aggregation techniques for local image descriptors. Section 2.3 describes deep learning models, in particular the convolutional neural networks used in this thesis. Section 2.4 describes different similarity measurements that can be used to compare image representations to generate a ranked list of images given an query image. Section 2.5 explains how to measure the performance of a retrieval system. Finally, Section 2.6 describes commonly used CBIR datasets. The fundamental concepts and methods described here are used throughout the remainder of this thesis.

2.2 Engineered approaches to image representations

The traditional procedure for extracting relevant information from an image consists of analyzing local patches over several interest keypoints, then aggregating local information (vectors associated with the patches) into a single image representation.

2.2.1 Scale-invariant feature transform

An image descriptor is a numerical representation of the image content (it can encode information about the colour, the shapes, textures etc.). One of the most popular descriptors in computer vision is the scale-invariant feature transform (SIFT) developed by Lowe in 2004 [79]. It was specifically designed to be invariant to different scales and rotations of the image. The "transform" term is applied because the algorithm transforms the image data into scale-invariant coordinates relative to the local features. It generates a large number of local descriptors that cover the image over the full range of scales and locations, which facilitates recognizing small objects in cluttered backgrounds or identifying partially occluded objects.

The algorithm consists of two main steps: keypoint localization, and keypoint description. The following describes each of these steps in more detail.

2.2.1.1 Keypoint localization

SIFT employs the scale-space theory [75] to find the relevant keypoints using a set of Gaussian kernels to represent the image at multiple scales. The idea is simple: if there is no way to a-priori know what is the relevant scale to analyze an image, then simultaneously consider multiple scales.

A particular scaled version of an image $L(x, y, \sigma)$ is realized by convolving the original image with a Gaussian kernel:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp^{-(x^2+y^2)/2\sigma^2}, \tag{2.1}$$

where $G(x, y, \sigma)$ is the Gaussian kernel characterized by the variance parameter σ^2. The σ represents the scale factor, and it controls the smoothness of the filtered version of the original image. Figure 2.1 shows an example of a scale space for one particular image. In this example, four different filtered versions ($L_n(x, y, \sigma_n)$, $n \in [1, 4]$) of the original image are generated.

Once the image is represented at multiple scales, the difference-of-Gaussian

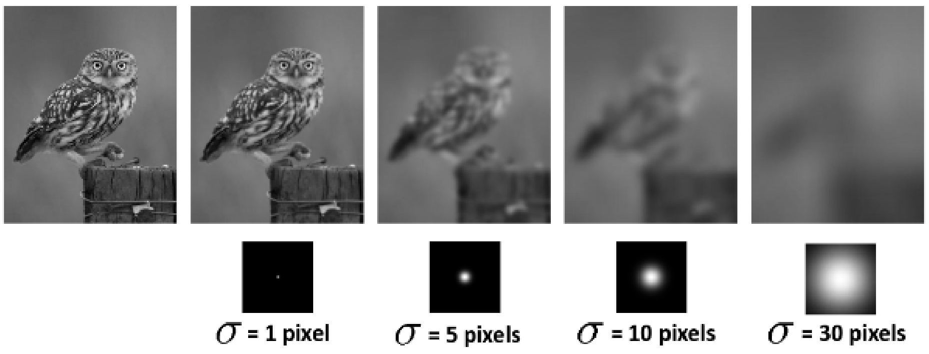


Figure 2.1: Examples of the multi-scale representation. Each filtered image ($L(x, y, \sigma)$) is obtained by convolving the original image with different four Gaussian kernels.

(DOG) operator is used to locate stable keypoints. It is computed from the difference of two nearby scales separated by a constant multiplicative factor k:

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, \sigma) - G(x, y, k\sigma)) \star I(x, y) \\ &= L(x, y, \sigma) - L(x, y, k\sigma). \end{aligned} \tag{2.2}$$

In practice, an image is divided into "octaves". In each octave N, the image is set to a fixed resolution $(H/2^{N-1}, W/2^{N-1})$ that is filtered by a set of Gaussians. The number of octaves and scales are hyperparameters. In particular, Lowe uses 4 octaves and 5 Gaussian kernels. Figure 2.2 shows an example of the space scale generated for each octave, and how DOG are generated.

To detect the relevant keypoints, the local maxima and minima within a $D(x, y, \sigma)$ are identified. If a pixel has the highest or minimum value when compared with its eight neighbours and the nine neighbours in the scale above and below (See Figure 2.3), then the location of the pixel taken as a candidate keypoint.

The obtained keypoints are further filtered to get more accurate results. Points with low contrast are more sensitive to illumination changes. If these points are not filtered, the same picture might have different keypoints under different lighting conditions. To detect these points, the Taylor series expansion of the $D(x, y, \sigma)$ is used to locate the real extrema within a region. If the intensity at this extema is

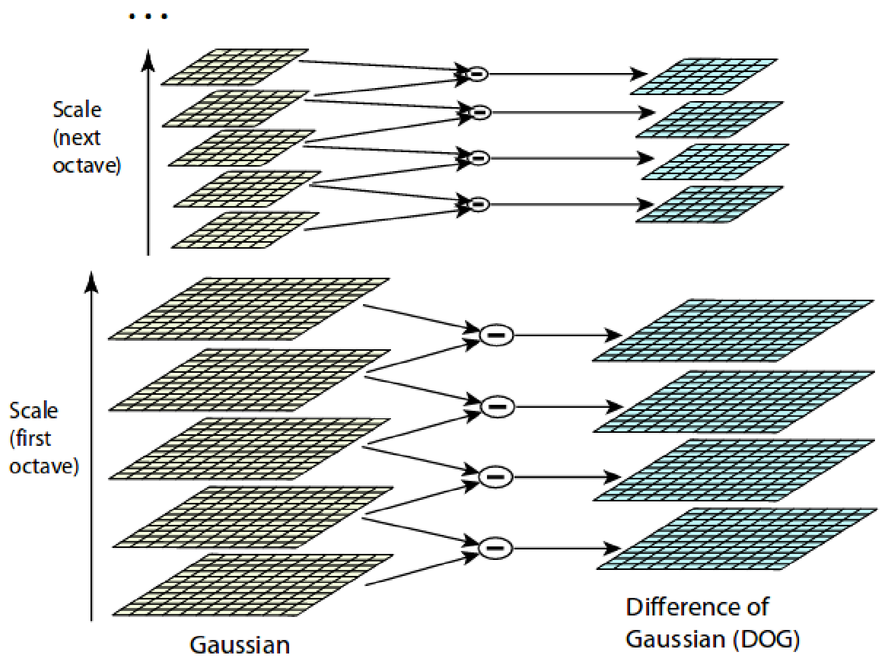


Figure 2.2: The initial image is incrementally convolved with a Gaussian to produce images separated by a constant factor k in scale space (column in the left). Adjacent images are subtracted to produce the difference-of-Gaussian images (column on the right). Once a complete octave has been processed, Gaussian images are resampled to have twice the initial value of σ (reducing the spatial resolution by half). Image source [79]

less than a threshold value (0.003 in the [79]), the keypoint is rejected.

$D(x, y, \sigma)$ have high responses to edges, which are not distinctive locations. Similar to the Harris keypoint detector [42], the trace and the determinant of the local Hessian matrix are used to identify and reject those keypoints associated with edges.

2.2.1.2 Descriptor generation

The first step in feature construction is orientation assignment. Using the scale of the point $(L(x, y, \sigma))$, the gradient magnitude $m(x, y)$ is computing following the next expression:

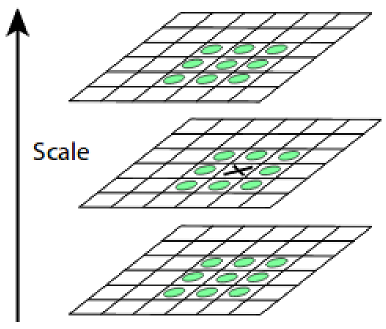


Figure 2.3: Maxima and minima of the difference-of-Gaussian images are detected by comparing a pixel (marked as X) to its 26 neighbours in 3×3 regions at the current and adjacent scales (marked with circles). Image source [79]

$$m(x, y, \sigma) = \sqrt{(L(x+1, y, \sigma) - L(x-1, y, \sigma))^2 + (L(x, y+1, \sigma) - L(x, y-1, \sigma))^2}, \tag{2.3}$$

and the orientation of the image gradient $\theta(x, y)$ is given by

$$\theta(x, y, \sigma) = \tan^{-1}\left(\frac{(L(x, y+1, \sigma) - L(x, y-1, \sigma))}{(L(x+1, y, \sigma) - L(x-1, y, \sigma))}\right). \tag{2.4}$$

A region surrounding a keypoint is considered. The maximum orientation is computed by generating a histogram of 36 bins where the orientation of each pixel is weighted by the magnitude of the gradient and a Gaussian window centered on the keypoint (with a σ parameter set to 1.5 times the scale of the keypoint). Figure 2.4 shows an example of a 8×8 pixel window where each pixel is associated to an arrow vector whose orientation and magnitude represents the orientation and magnitudes of the gradient of $L(x, y, \sigma)$. On the left, an illustration of an 8-bin histogram of orientations is displayed characterizing the whole window. Any point within 80% of the highest orientation peak is used to create a new keypoint with the same main orientation.

Each point is then described by location, scale, magnitude, and orientation (main

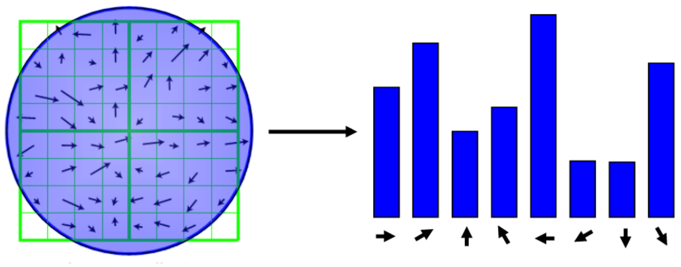


Figure 2.4: 8-bin histogram based in the gradient orientations that characterizes a particular keypoint. Image source [79]

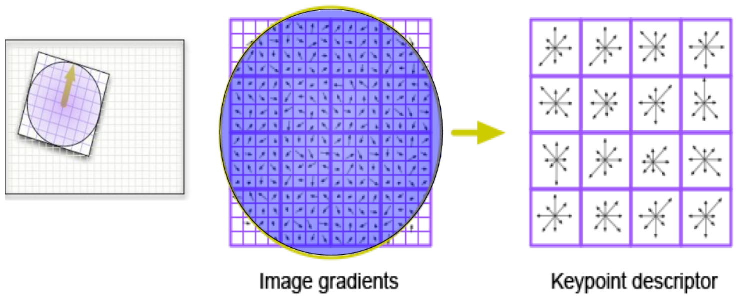


Figure 2.5: SIFT visual: The first step consists of rotating the local coordinates around a keypoint according to the main gradient orientation (left). Then, the 16×16 window is divided into 4×4 blocks, which are weighted by a Gaussian kernel. For each block, an 8 bin histogram of orientation is created, generating a final descriptor of 128 dimensions. Image source [79]

orientation of the region). To generate the final feature descriptor, the 16×16 region is rotated according to the computed main orientation of the keypoint. Then, the gradient region is further divided into 4 blocks (of 4×4 pixels each). A Gaussian weighting around the center is again applied to weight the gradient magnitudes (with σ set to 0.5 times the scale of the keypoint). Finally, each block is described by an 8 bin orientation histogram, where the contribution of the pixel is weighted by the gradient magnitude, generating a representation of $4 \times 4 \times 8 = 128$ dimensions. The whole pipeline to generate the local descriptor of a keypoint is illustrated in Figure 2.5.

Scale invariance is achieved by the Gaussian weighting around the interest point, which determines the area of interest around the keypoint according to its associated scale. **Rotation invariance** is achieved by rotating the local coordinates to the peak gradient orientation.

The final descriptor is further normalized to unit length to achieve **illumination invariance**. In this way, the weighted entries in the histogram will be invariant under local affine transformations of the image intensities around the interest point, which improves the robustness of the image descriptors under illumination variations.

2.2.1.3 Matching of local image descriptors

Given a set of SIFT descriptors extracted for two images, the most straightforward way to compare how similar the images are is to check the number of local matches that they share. Each local descriptor of one image is compared against all the local descriptors of the second image using the Euclidean distance, and the ones with distances below a prescribed threshold are considered as a match. Unfortunately, this strategy generates a large number of false positive matches.

A better criteria is to consider the distance between the closest match and the second nearest match. This approach is known in the literature as the Nearest Neighbour distance ratio (NNDR [79]): For a particular descriptor in the reference image, all local features in the target image are sorted by their Euclidean distance. The nearest neighbour descriptor is taken as a match only if the ratio between its distance and the second nearest neighbour is less than 0.8. In Lowe's experiments, this procedure removed around 90% of false matches while discarding less than 5% of the correct matches.

The SIFT algorithm generates hundreds to thousands of local descriptors per image. Computing the nearest neighbour matching (NN) of all local descriptors in a large dataset is highly computational demanding since it requires computing as many distances as there are local descriptors in the dataset for every local descriptor in the query image. Approximate methods, such as k-d trees [34], are often used to speed-up the processing time over exhaustive search. In particular, the Best-Bin-First (BBF) algorithm [10] is used in Lowe's work [79] to approximate the search of local matches by returning the closest neighbour with high probability.

It is common to apply methods to ensure the geometric consistency among matches, techniques such RANSAC [33, 96] are usually applied to improve retrieval performance. This will be furthered discussed in Section 3.2.

2.2.2 Aggregation techniques

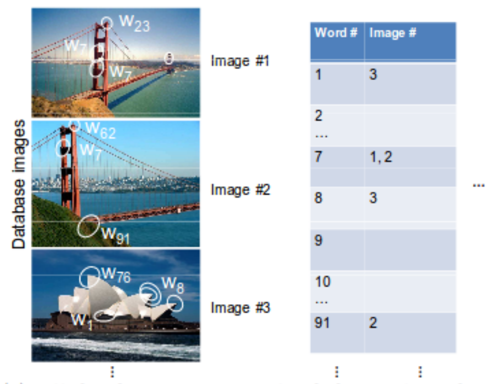
Aggregation methods "summarize" the local representations extracted from an image into a fixed-length representation, reducing both memory storage and query search time, since only a single representation is used per image. The following describes three widely used aggregation methods for image retrieval: bag of visual words (BoW), vector of locally aggregated descriptors (VLAD), and Fisher vectors (FV).

2.2.2.1 Bag of visual words

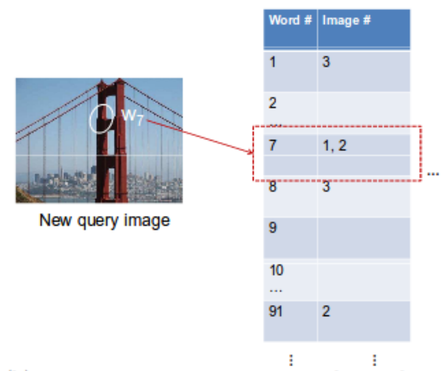
The most widely-used approach for CBIR is the bag of visual words (BoW), where local features are extracted from each image and mapped to discrete visual words [76]. The presence of a certain image feature is treated like the presence of a word in a text document. This approach can benefit from the construction of inverted files that significantly speed up the retrieval time and storage requirements. An inverted index is an index data structure storing a mapping from content (i.e visual words) to their location in a database file (i.e the images in which those words occur). Figure 2.6 illustrates the main idea of an inverted file: an image is considered as relevant if it shares some visual words with the query. This computation can be performed very efficiently by a matrix vector product using sparse matrix representations.

BoW encoding consists of two main steps: the visual codebook generation and the encoding of the local features based on the obtained visual words.

The **visual codebook** (vocabulary) in the BoW model consists of partitioning the local feature space into k non-overlapping regions, called Voronoi cells. The centroids of these cells define the words of the codebook . k-means [77] is typically used in this step. It is an unsupervised clustering algorithm that iteratively refines the centroids, such that the average distance from a point to the center of its



(a) All database images are loaded into the index mapping words to image numbers.



(b) A new query image is mapped to indices of database images that share a word.

Figure 2.6: Main idea of an inverted file index for images represented by visual words. Image source `http://www.cs.utexas.edu/~grauman/courses/fall2009/papers/bag_of_visual_words.pdf`.

cluster is approximately minimized. Its objective can be described with the following expression:

$$\underset{S}{\text{argmin}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} ||\mathbf{x} - \mu_i||_2^2, \tag{2.5}$$

where $S = \{S_1, S_2, ..., S_k\}$ is the set of non-overlapping cluster assignments, S_i is the set of all points in cluster i and $\mu_i \in R^d$ (d is the dimension of the local features, i.e 128 for SIFT) is the centroid of the cluster i (mean of all the points in S_i). The objective function for k-means is non-convex and exact minimization is NP-hard. Lloyd's algorithm [77] finds a locally optimal solution. It proceeds as follows:

Given an initial set of (randomly initialized) centroids $K = \{\mu_i\}, i \in \{1, ..., k\}$, the objective function in Equation (2.5) can be approximately minimized by iteratively following two main steps:

1. assign each descriptor in the dataset $X = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ to the nearest neighbour cluster $a(\mathbf{x}_n) = \underset{i}{\text{argmin}}\, ||\mathbf{x}_n - \mu_i||_2^2$.
2. re-compute the centroids $\mu_i = \frac{1}{|S_i|} \sum_{\mathbf{x} \in S_i} \mathbf{x}$ based on the new assignment set S, where $|S_i|$ denotes the cardinality of S_i.

Convergence is achieved when assignments do not change or the change in the position of the clusters $\{\mu_i\}$ is insignificant, finding a locally optimal partition of the feature space. The method is highly sensitive to the initial placement of the clusters. Typically, the initial centers are chosen randomly from the data points. However, a better initialization of the centroids is given by the *k-means++* algorithm [5]. The main idea of the algorithm consists of selecting centroids that are far away from each other. For this, the algorithm starts by randomly selecting an initial cluster from the data points. Then, the next cluster is selected given the probability of a data point being chosen as a centroid, which is proportional to the square of the distance between a data point and the initial centroid. This strategy leads to a better partitioning of the feature space and faster convergence of the k-means algorithm, since the initial set of centers are closer to the optimal solution than a random initialization.

The main bottleneck of the k-means method appears when dealing with large datasets and vocabularies. At each iteration, k-means needs to compute the Euclidean distance between all the training descriptors and the clusters centers.

In practice, BoW uses vocabularies composed of a large number of visual words (500K - 16M visual words). High dimensional BoW representations obtained from such vocabularies are much more discriminative than those obtained with a reduced number of clusters. To scale-up k-means to high dimensional visual vocabularies, a method such as hierarchical k-means (HKM) [89] or approximate k-means can be used [96]. HKM consists of using a tree structure to apply k-means with a reduced number of clusters on each level of the tree. AKM, on the other hand, replaces the exact nearest neighbour calculation by an approximate nearest neighbour using randomized k-trees [79], which have been shown to work better than HKM in object retrieval scenarios [96].

In the BoW encoding, a set of N d-dimensional local descriptors $\{\mathbf{x}_i\}$ of an image are aggregated into a single representation $\mathbf{h}_{\text{BOW}} = (h_1, ..., h_h)$, where $h_k = \sum_{i=1}^{N} q_{ik}$ represents the count of how many local descriptors are assigned to a particular visual word k of a trained vocabulary. The **hard assignment** of the features associates each local feature of the image $\mathbf{x}_i$ to its nearest neighbour visual word in the vocabulary, and can be expressed as

$$q_{ik} = \begin{cases} 1 & \text{if } k = \underset{m}{\operatorname{argmin}} \|\mathbf{x}_i - \mu_m\|_2^2 \\ 0 & \text{otherwise.} \end{cases} \tag{2.6}$$

As in text retrieval approaches, it is common to apply a weighting to the component of this vector [114]. The standard weighting is know as "term frequency-inverse document frequency," *tf-idf*. Each word in the final histogram, instead of contributing equally to the histogram (Equation 2.6), contributes with a weight defined as:

$$w_k = \frac{n_{ki}}{n_i} \log \frac{N}{n_k}, \tag{2.7}$$

where n_{ki} is the number of occurrences of word k in i-th document (image), n_i is the total number of words in the i-th image, n_k is the number of images containing the word k and N the total number of images in the dataset. This way, words occurring often within an image are up-weighted, while the most frequent words (the most common) across the whole dataset are down-weighted, providing an estimation of how relevant a particular visual word is in an image considering the whole image dataset.

The hard assignment of the features to the closest visual word generates what it is known as "word uncertainty," related to quantization error. Figure 2.7 shows the division of the local feature space into different non-overlapping Voronoi cells. Two local features associated with the same visual word contribute equally to the construction of the histogram regardless of their location with respect the centroid. Moreover, the hard assignment completely ignores nearby clusters and treats them as completely unrelated when they are likely to encode similar patches.

Soft assignment [22, 36] is an improvement over the bag of words representation that considers simultaneously more than one visual word per local descriptor.

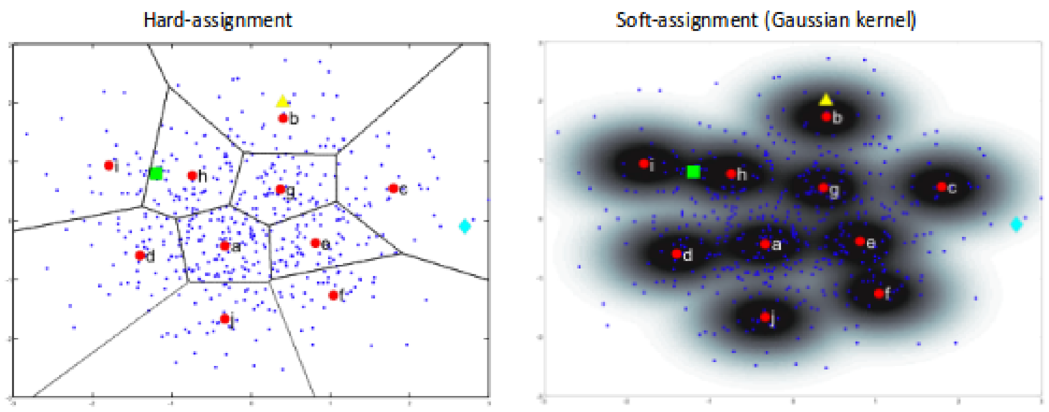


Figure 2.7: Visual representations of the visual vocabulary. On the left, hard assignment is used, which means the BoW representation consists of directly counting the local features falling in a each particular cell. A descriptor far from the cluster center contributes equally to a descriptor located near the cluster (blue point on the left diagram). Also, a descriptor close to the border region is only assigned to one visual word, without considering near-by words (the green point in the left diagram is assigned to cluster h, although it is very close to cluster i too). These effects are mitigated by using soft assignment, where the location within a cell and nearby centroids are taken into account when constructing the final representation. Image source [36] .

Gemert *et al.* [36] proposed to use a kernel codebook scheme to encode each local feature proportionally by measuring their similarities with all visual words as

$$q_{ik} = \frac{K_\sigma(\|\mathbf{x}_i - \mu_k\|)}{\sum_{k=1}^{M} K_\sigma(\|\mathbf{x}_i - \mu_k\|)} \tag{2.8}$$

where M represents the total number of centroids, $K_\sigma(\cdot)$ is the kernel function defined as $K_\sigma(x) = e^{-\frac{(x-\mu)^2}{2\sigma^2}}$, where σ is the corresponding smoothing hyperparameter. With the soft assignment each feature is assigned to multiple words ($\sum_{k=1}^{K} q_{ik} = 1$). Gemert *et al.* Compared with hard assignments, explicitly modeling of visual word ambiguity with soft assignment helps to improve the image representation's power at the price of losing the sparsity of the histograms, and therefore the efficient usage of inverted files for retrieval.

2.2.2.2 Vector of locally aggregated descriptors

The vector of locally aggregated descriptors (VLAD) [54] is another popular technique to aggregate locally invariant descriptors from an image. Similar to BoW, it is based on the construction of a visual vocabulary and uses hard-assignments to associate each local descriptor to the closest visual word. The main difference between VLAD and BoW is that, instead of directly encoding the word frequencies, it accumulates the residual of each descriptor with respect to its assigned cluster, as illustrated in Figure 2.8, adding more information in the encoding and generating a more powerful representation.

The final vector $\mathbf{f}_{\text{VLAD}} = (\mathbf{u}_1^T, ..., \mathbf{u}_K^T)^T$ is the concatenation of residual vectors $\mathbf{u}_k = \sum_{i=1}^{N} q_{ik}(\mathbf{x}_i - \mu_k)$, $\mathbf{u_k} \in R^d$, where q_{ik} denotes the hard assignment (Equation 2.6) of the descriptor $\mathbf{x}_i$ to the centroid μ_k. The final representation is a dense matrix of dimensions $d \times K$.

Several normalization techniques have been proposed for VLAD encoding: direct L^2-normalization [54] and signed square rooting [50] of $\mathbf{f}_{\text{VLAD}}$, both are known

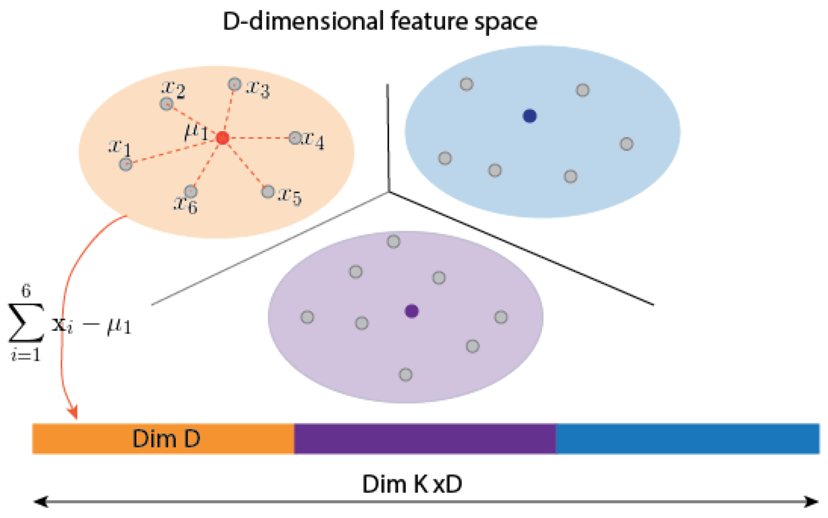

D-dimensional feature space
x_2
x_3
μ_1
x_4
x_1
x_6
x_5
$\sum_{i=1}^{6} \mathbf{x}_i - \mu_1$
Dim D
Dim K xD

Figure 2.8: VLAD vector construction: The relative position of the local descriptors within a Voronoi cell directly encoded in this aggregation. For each cluster, illustrated with different colours, a residual vector is created by computing the distance between $\mathbf{x}_i$ and the centroid μ_k. The final vector is the concatenation of all the residuals.

to assign too much weight to some of the vector components affecting the final similarity measure (Euclidean distance). Intra-normalization [4] alleviates this by individually L^2-normalizing the residual vectors $\mathbf{u_k}$ and then applying signed square root normalization followed by a final L^2-normalization of the final vector.

Although the number of visual words required for this encoding is significantly less than the large vocabularies used in BoW (the typical value for VLAD is 64 to 256 clusters), the vector obtained is very high dimensional and dense. For instance, a codebook of 64 visual words would generate a vector of 8,192 dimensions using SIFT descriptors; 256 clusters generates a 32,768 dimensional vector. It is necessary to apply dimensionality reduction to reduce the memory footprint of the representations to scale the usage of VLAD to large image datasets [54, 50].

The strategy of encoding residuals makes VLAD a more robust representation for retrieval in comparison with BoW. However, this is only true when the vocabulary size of BoW is moderate (from 20k to 200k centroids). For instance, BoW achieves a mean Average Precision (mAP) [1] of 0.35 in the Oxford dataset [2] with a vocabulary of 20k centroids while VLAD achieves 0.38 with a vocabulary of 64 centroids (8,192 dimensional vector) according to [28]. The usage of large vocabularies for BoW allows to represent more diverse visual patterns that compares favorably with the VLAD aggregation, that only utilizes a small number of centroids. In particular, 1M vocabulary for BoW achieves 0.618 mAP in the Oxford [3].

It is unclear whether one representation is better than the other, because the difference in performance is related to the kind of retrieval scenario targeted. In general, on datasets where the instances to retrieve occupy the entire image, such as Holidays dataset or UKB, VLAD tends to outperform BoW using a a more compact representation as shown in [54]. However, in more challenging scenarios, where the instances appear with different sizes or partially occluded (Oxford, Paris, or TRECVID), BoW with large vocabularies are usually the approach used [3, 88, 141, 148, 149].

[1]See Section 2.5 for more details of this metric.
[2]See Section 2.6 for a summary of retrieval datasets.

2.2.2.3 Fisher vectors

The Fisher vector encoding (FV) [95] captures the average first and second order differences between the local image descriptors and the centers of a Gaussian Mixture Model (GMM). Intuitively, FV represents how the parameters of the GMM should be modified to better fit a set of local features that describe a particular image. The main difference between BoW and FV is the construction of a "soft vocabulary" via GMM. With this model, the feature samples are described by the following probability distribution:

$$p(\mathbf{x}|\theta) = \sum_{k=1}^{K} p(\mathbf{x}|\mu_k, \Sigma_k)\pi_k, \tag{2.9}$$

where $p(\mathbf{x}|\mu_k, \Sigma_k)$ represents the probability that a feature vector $\mathbf{x}$ is described by the kth Gaussian distribution, defined by the parameters $\mu_k \in R^d$, $\Sigma_k \in R^{d\times d}$ representing the mean and covariance matrix (which is typically taken to be diagonal to reduce the number of training parameters) respectively. The contribution of each Gaussian in the final probability is weighted by π_k ($\pi_k \geq 0$ and $\sum_{k=1}^{K} \pi_k = 1$). The parameters are calculated using expectation maximization (EM) [82].

Analogous to the codebook kernel [36] used for the soft-assignment in BoW, the GMM defines a soft-assignment data-to-cluster assignment following

$$q_{ik} = \frac{p(\mathbf{x}_i|\mu_k, \Sigma_k)\pi_k}{\sum_{j=1}^{K} p(\mathbf{x}_i|\mu_j, \Sigma_j)\pi_j}. \tag{2.10}$$

The main idea of FV representation is to characterize an image (described by a sample of local features) by its deviation from the GMM distribution. For that, a fixed length representation is constructed by calculating the partial derivatives of the log-likelihood with respect its parameters [49]. The final representation $\mathbf{f}_{\text{fisher}} = \left[\mathbf{u}_1^T, \mathbf{v}_1^T, ..., \mathbf{u}_K^T, \mathbf{v}_K^T\right]^T$, is the concatenation of the partial derivatives obtained for the log-likelihood of the GMM, described by:

$$\mathbf{u}_k = \frac{1}{N\sqrt{\pi_k}} \sum_{i=1}^{N} q_{ik} \left(\Sigma_k^{-1} (\mathbf{x}_i - \mu_k) \right) \tag{2.11}$$

$$\mathbf{v}_k = \frac{1}{N\sqrt{2\pi_k}} \mathrm{diag} \left(\sum_{i=1}^{N} q_{ik} \left[\Sigma_k^{-1/2} (\mathbf{x}_i - \mu_k)(\mathbf{x}_i - \mu_k)^T \Sigma_k^{-1/2} - I \right] \right). \tag{2.12}$$

The descriptor obtained contains $2dK$ dimensions assuming a diagonal covariance matrix (d being the dimension of the local feature space). Similarly to VLAD, the number of clusters (Gaussians for the GMM) is less than in BoW, since the encoding already contains first and second order statistics of the distribution of the data which generates a much richer representation. However, FV are dense (like VLAD) and more computational costly than VLAD and BoW. The final vector is intra-normalized [4] and dimensionality reduction is usually necessary to scale the representation to large datasets [95].

Similarly to VLAD aggregation, FV representations tend to outperform BoW and VLAD in relatively simple image retrieval scenarios such as in the Holidays or the UKB datasets with extremely compact image representations (only 64 or 32 dimensions) [55], making FV especially suitable when it is required a very compact image representation (while compromising the retrieval performance). In practice, as discussed in the previous subsection, BoW with large vocabularies followed by spatial re-ranking and query expansion steps [3] is the dominating approach in the literature for visual instance retrieval.

2.3 Deep learning based image representations

Deep learning is a sub-field of artificial intelligence that aims to model high level abstraction from raw data by using model architectures composed of multiple non-linear transformations of the input data. Architectures such as multi-layer perceptron, convolutional deep neural networks, deep belief networks and recurrent neural networks have been applied in fields like computer vision, speech recognition, natural language processing or bioinformatics‘ [30].

2.3.1 The single neuron model

A single neuron (or perceptron) has a number I of inputs x_i and one output y. The inputs generally correspond to raw features of a single training example. A weight w_i $(i = 1, ..., I)$ is associated to each input. Usually, an additional input x_0 permanently set to 1, is associated to a weight w_0, called the "bias" term. The activation of the neuron is computed by adding its weighted inputs $a = \sum_{i=1}^{I} w_i x_i$. Then, the final output y is set as a function of the activation.

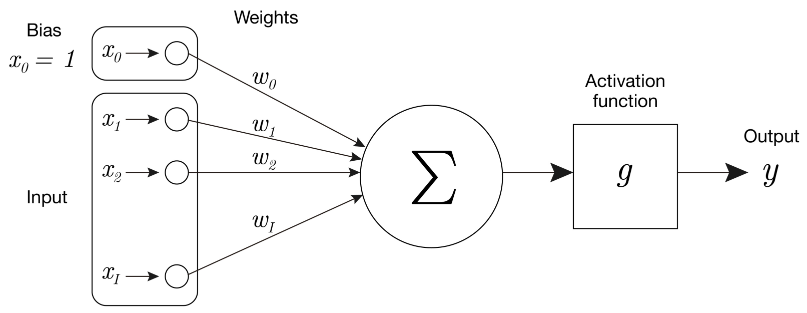


Figure 2.9: Single neuron.

The *activation function* (g) makes the neuron "fire" or not given a certain input and it is often a non-linear function. Figure 2.10 shows the behaviour of three popular activation functions: Sigmoid, Tanh and Rectified Linear Unit (ReLU). A neuron can then be seen as a linear regression or as a linear decision boundary classifier. Given a particular set of training data $(\mathbf{x}_i, y_i)_{i-1}^{N}$ with each $\mathbf{x} \in R^{(I)}$ and $y_i \in \{-1, 1\}$, and a loss function to evaluate the error between the prediction $\hat{y}$ and the real value y; it is possible to tune the parameters of the neuron to fit the data. [3]

Multi-class classification or regression problems can be addressed by considering simultaneously multiple neurons. In the case of classification, it is common to apply the *softmax* as the activation for the neurons . The softmax is defined as:

$$f_i = \frac{e^{a_i}}{\sum_j e^{a_j}} \tag{2.13}$$

[3]In the case of regression $y \in \mathbb{R}$, and for logistic regression $y_1 \in \{0, 1\}$.

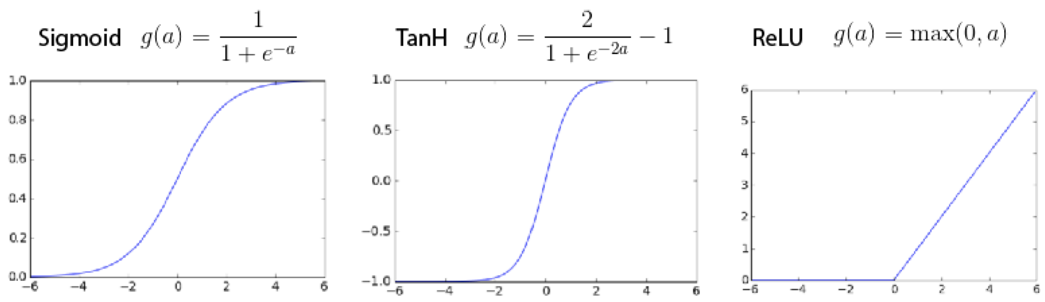


Figure 2.10: Three activation functions commonly used in the neurons.

where a_i is the weighted sum of the inputs for the node i. This function is used to normalize the outputs so that they are a categorical probability distribution over the possible outcomes.

Linear decision boundary classifiers are suitable when the data is likely to be linearly separable, with the advantage that they are fast to compute (mainly a dot product between weights and inputs), and are easy to interpret: the weights directly represent the strength of the influence of each input on the final classification. However, real-world data (images, videos, EEG, a Audio, etc.), without any further pre-processing, contain complex patterns and are usually not linearly separable.

2.3.2 Neural networks

Neural networks are composed of one input layer, one output layer and one or more hidden layers. Each layer is composed of neurons, each one connected to all the inputs of the previous layer (also called multi-layer perceptron or *fully connected* networks).

The neurons within a layer can be seen as a collection of binary classifiers that "fire" when they detect a particular pattern on its inputs. As we add more layers, the network has the ability to represent increasingly more complicated non-linear functions that are more suitable to solve the final task. For instance, in a classification network, the input to the final layer gives a representation of the original data that is more likely to be linearly separable.

The number of hidden layers and neurons per layer are hyperparameters that

$$\mathrm{h}^{(1)} = g(\mathrm{W}^{(1)}\mathrm{x} + \mathrm{b}^{(1)})$$

$$\mathrm{h}^{(2)} = g(\mathrm{W}^{(2)}\mathrm{h}^{(1)} + \mathrm{b}^{(2)})$$

$$\mathrm{y} = g(\mathrm{W}^{(3)}\mathrm{h}^{(2)} + \mathrm{b}^{(3)})$$

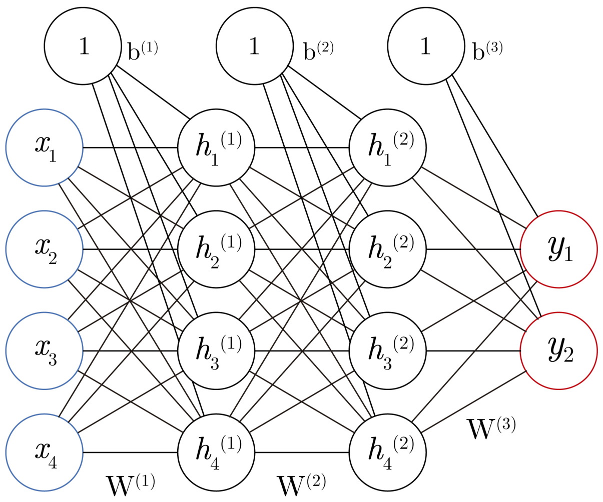


Figure 2.11: A Neural Network with an input layer, an output layer and two hidden layers.

need to be carefully selected. Usually, the more layers added the more capacity the network has to detect complex patterns. However, the amount of parameters to estimate grows very quickly with the number of layers and neurons[4], which may lead to a model that fits the training data extremely well but has very poor generalization on new data (this is termed *overfitting*).

2.3.3 Gradient descent

Gradient descent (GD) is a simple first-order iterative optimization algorithm often used to find a (local) minimum of a function. This is particularly useful when dealing with non-convex optimization problems such as those addressed with CNN models. Given a cost function[5] associated with a model $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$, the algorithm consists of updating the parameters in the direction opposite to the gradient of the cost. If $\hat{\mathbf{y}}$ is

[4]The number of parameters of the ith layer can be computed from the number of neurons at the previous layer (N^i_{input}), the number of neurons within the layer (N^i_{output}) and the bias terms, which are equal to the number of nodes on the layer being $N^i_{total} = N^i_{input} \times N^i_{outputs} + N^i_{outputs}$.

[5]Note that cost and loss functions refer to slightly different concepts. The loss function is a function defined on a data point (hinge loss, square loss, etc.), whereas the cost function is usually a more general object, such as the the sum of loss functions over the full training plus some complexity penalty.

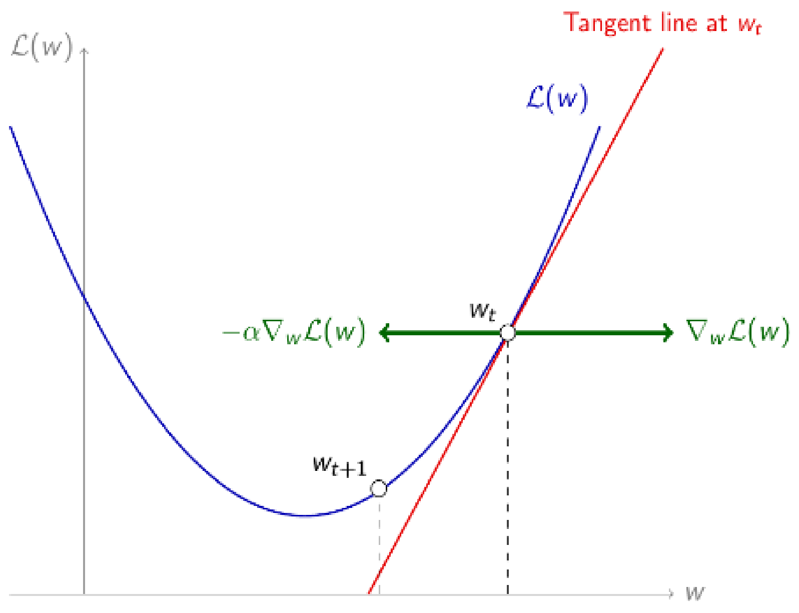


Figure 2.12: Visualization of one step of gradient descent optimization on a one dimensional function.

the output of a NN composed of K hidden layers, the update to the weights on a neuron in layer l can be described as:

$$W^{(l)} \leftarrow W^{(l)} - \alpha\nabla_{W^{(l)}}\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}), \tag{2.14}$$

where α is the *learning rate*, which is an hyperparameter defining the size of the step towards the minimum of the loss. Figure 2.12 illustrates the idea of GD considering one dimensional weights for visualization.

Gradient descent requires calculating the loss and the gradients for all the training set to perform one single parameter update, which usually is very computationally demanding and slow. In practice, *mini-batch Stochastic Gradient Descent* (SGD) is used. For this, the training dataset is shuffled, and a batch of N samples is taken to compute a stochastic estimate of the gradient of the cost and update $W^{(l)}$ parameters.

For optimizing deep neural networks, SGD is usually combined with the back-

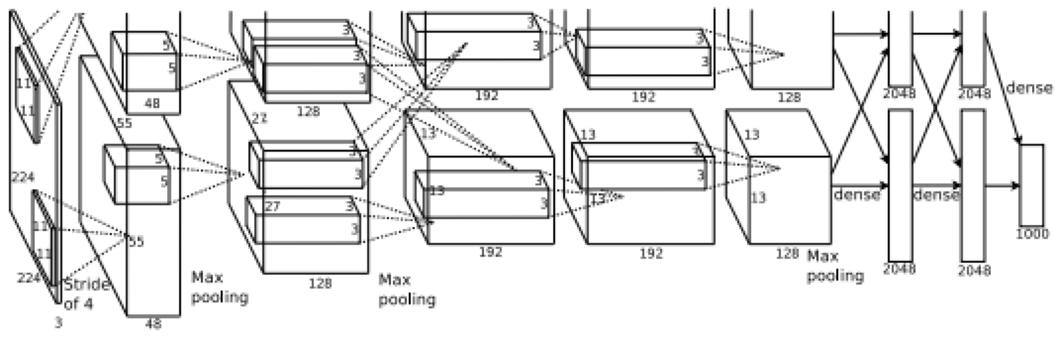


Figure 2.13: AlexNet architecture. This consists of five convolutional layers, each one followed by a max-pooling layer followed by three fully connected layers. ReLUs are used as the activation function in all layers except the last layer, where outputs are normalized with a softmax activation function. Image source [64].

propagation algorithm [105], which combines the calculus chain rule and dynamic programming techniques to keep the computations involving the parameter updates tractable.

2.3.4 Convolutional neural networks

Convolutional Neural Networks (CNN) have been considered as the dominant approach to producing image representations for many computer vision task since the pioneering work carried out by Krizhevsky *et al.* [64] in the 2012 ImageNet challenge. The main difference between CNNs and the fully connected Neural Networks are the *convolutional* and *pooling* layers, which make CNNs particularly suitable for processing spatially structured data such as images.

A convolutional layer outputs a volume of neurons. The depth of the volume is defined by a set of kernels or filters. The layer is characterized by its **local connections**: in contrast to fully connected layers, each neuron in a convolutional layer is connected with a local region of its input. In particular, given an input layer of dimensions $H \times W \times C$, one neuron within a convolutional layer is connected to a window volume of size $C \times K \times K$ of the input, where K is the kernel size. The output of the neuron corresponds to the response of its activation function to the summation of the $C \times K^2$ weighted inputs. Figure 2.14 shows an example of a

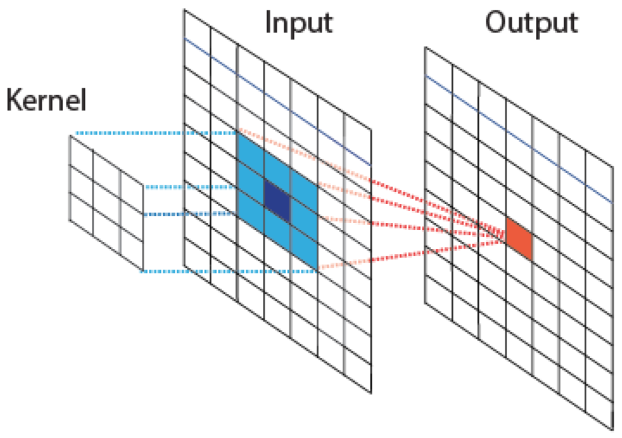


Figure 2.14: Visualization of the local connection of one neuron within a convolutional layer. In the example, the input depth $C = 1$ and the kernel size is $1 \times 3 \times 3$. Usually, C has a larger value. For instance, most CNN models process RGB images, which means that the input layer has to have $C = 3$. The kernel is slid over the spatial dimensions $H \times W$ keeping the same weights across all the different locations. The output is a 2D map, also called the activation map or feature map. Within a convolutional layer, N different kernel filters are considered, each one with independent parameters, generating a volume of neurons composed by N activation maps

$1 \times 3 \times 3$ kernel.

Conceptually, the kernels (filters) are "moved' across the spatial dimensions of the input in a sliding window manner, changing its position by S neurons (stride) each step. Their **weights are shared** or fixed across the spatial positions of the input's layer, with the assumption that if a particular visual pattern is relevant in one position of the image, it will also be relevant in another position.

The pooling layers are used to reduce the spatial dimension of the the convolutional volumes. They operate spatially over a convolution volume. Usually max pooling is applied between consecutive layers to reduce the dimensionality of the representations and achieve some translation invariance. Figure 2.15 illustrates an example of spatial max-pooling using a kernel of 2×2 with stride 2.

The AlexNet proposed by Krizhevsky [64] in 2012 has five convolutional layers and three fully connected, containing 60 milion parameters. The model was trained using SGD with momentum [98] on 1.2 million labeled images from the ImageNet

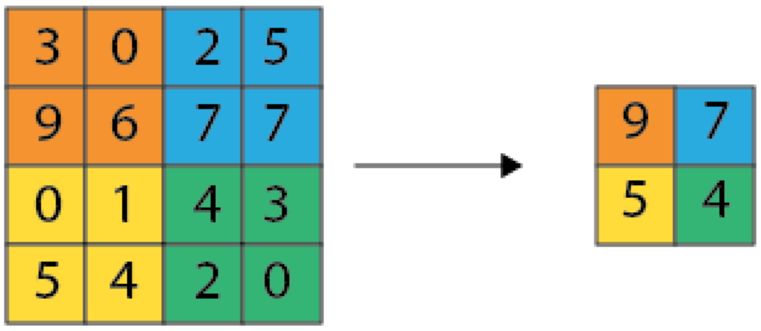


Figure 2.15: Spatial max pooling with 2×2 kernel and stride 2 applied on a single activation map of a convolutional layer.

dataset [29]. Figure 2.13 shows the architecture of the model. This architecture achieved the best result in the ILSVRC-2012 image classification task.

Three main factors that lead to success of this method were having access to a huge number of labeled images, effective use of GPUs for fast computation of convolutional operations, and the usage of non-saturating activation functions (ReLU) to speed-up training and lesser gradient vanishing.

Since then, several works have appeared in the literature including networks trained for object detection [102], semantic segmentation of images [78], visual attention and saliency [7, 67] and image captioning [62, 31]. Representations derived from CNN models have also been considered for image retrieval (see Chapter 3), mostly by using representations from networks pre-trained for classification as a substitute for traditional hand-crafted descriptors such as SIFT.

2.3.4.1 Widely used pre-trained CNN models

Models trained on ImageNet [29] have achieved astounding performance on the image classification task. The architecture proposed by Krizhevsky [64], also known as AlexNet, is composed of five convolutional layers, each one followed by a max-pooling layer, and three fully connected layers, as shown in Figure 2.13.

Recently, other architectures have been proposed in the literature achieving

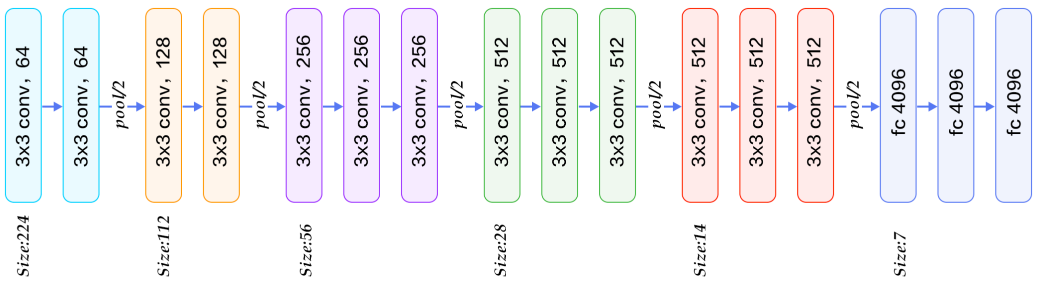


Figure 2.16: VGG16 architecture. The network is divided into 5 convolutional blocks. Each one is composed by a stack of convolutional layers of the same kernel size (2 convolutional layers in blocks 1 and 2; and 3 convolutional layers in blocks 3, 4, and 5). After each block, a max-pooling layer is applied, reducing the spatial dimensions of the convolutional feature maps to a half. The last block is composed by three fully connected layers. Image source `http://book.paddlepaddle.org/03.image_classification/image/vgg16.png`.

better classification performance. Simonyan and Zisserman proposed a very deep convolutional network [113] where the authors proposed increasing the number of convolutional layers and using very small (3×3) convolutional filters, pushing the depth to 16 and 19 layers (VGG16 and VGG19). Figure 2.16 shows the VGG16 architecture. The number 16 refers to the number of convolutional and fully connected layers, without taking into account max-pooling layers. Each convolutional layer have a ReLU as activation function, with exception of the last fully connected layer, that uses a softmax for the final classification prediction. In this thesis we use the VGG16 architecture, extracting feature from the fifth convolutional block. We refer to the layers of this block as conv5_X, being $X = 1, 2, 3$ identifying the first, second, and third convolutional layers within the block. pool5 layer to the output after applying max-pooling on conv5_3.

It was shown that the deeper the network the better results achieved. However, He *et al.* [43] observed that very deep networks had higher training error. For instance, a 56-layer network had higher training error than a 20-layer network. The authors reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. With these skip connections it was possible to train very deep architectures of 50 or 101 layers. In particular the

152-layer architecture proposed in [43] achieved 3.75% top-5 error on the test set of ImageNet and the submission won the ILSVRC 2015 classification challenge. The proposed model was named "ResNet", usually referred as ResNet-N where N refers to the number of layers of the network, typically 50 or 101.

2.3.4.2 Fine-tuning a CNN model

Since it is unlikely to have 1.2 million annotated images (as in the ImageNet dataset [29]) available for an arbitrary computer vision task, it is common practice to re-use an already trained architecture for a different task. The process of *fine-tuning* is a type of transfer learning [110, 137], fine-tuning consists of modifying some of the top layers of an already trained model to optimize the weights for the new task, instead of training from randomly initialized weights. Using the obtained pre-trained weights instead of setting them to random values allows a fast convergence of the models and training of the networks when it is not possible to access to a large collection of annotated images. We exploit this process for image retrieval in Chapter 5.

2.4 Image similarity

Similarity measurement is an important component in a CBIR system. Once the image representations have been generated for the image dataset and a query, it is necessary to compute the similarity between the query image representation and the database images, which can be calculated using different distance metrics. The smaller the distance the more similar the two images.

We introduce some of the common metrics and measures used in CBIR: p-norm induced metrics, cosine similarity (which is the one used in this thesis), and histogram oriented metrics.

2.4.1 p-norm induced metrics

The generalized Minkowski metric is a distance defined in the L_p-norm space:

$$D_p(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{d} |x_i - y_i|^{\,p} \right)^{\frac{1}{p}}, \tag{2.15}$$

where $p \geq 1$ and $\mathbf{x} = (x_1, ..., x_d)^T$, $\mathbf{y} = (y_1, ..., y_d)^T$ represent two image representations in the N dimensional space. Three important cases are:

- $p = 1$, then the distance is known as the L_1*-distance* or *city block* or *Manhattan distance*, and is defined as:

$$D_1(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_1 = \sum_{i=1}^{N} |x_i - y_i|. \tag{2.16}$$

- $p = 2$, then the distance is known as the L_2*-distance* or *Euclidean distance*, and is defined as:

$$D_2(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2 = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}. \tag{2.17}$$

- $p = \infty$, then the distance is known as the L_∞*-distance* or *Chebyshev* distance, and is defined as:

$$D_\infty(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_\infty = \max |x_i - y_i|. \tag{2.18}$$

2.4.2 Cosine similarity

This metric depends on the angle θ between two feature vectors. It is defined as:

$$\cos\theta = \frac{\mathbf{x}^T \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}. \tag{2.19}$$

If descriptors are normalized ($\|\mathbf{x}\| = \|\mathbf{y}\| = 1$), then the cosine distance ($D_{cos}(\mathbf{x}, \mathbf{y}) =$

$-\frac{\mathbf{x}^T\mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|}$) and the rankings produced by the Euclidean distances are equivalent. Considering normalized descriptors, we can expand the Euclidean distance with the following expression:

$$\|\mathbf{x}-\mathbf{y}\|_2^2 = \mathbf{x}^T\mathbf{x} + \mathbf{y}^T\mathbf{y} - 2\mathbf{x}^T\mathbf{y} = 2 - 2\mathbf{x}^T\mathbf{y}, \tag{2.20}$$

which gives the same rankings as $-\mathbf{x}^T\mathbf{y}$. So for L^2-norm vectors it is unnecessary to investigate Euclidean distance and cosine similarities since they are equivalent.

The main advantage of using cosine similarity is that it is fast to compute if the vectors are normalized. The ranking generation can be efficiently done with matrix multiplication. For instance, for a dataset of m images described by vectors in R^d, $X \in \mathbb{R}^{m\times d}$, and query $\mathbf{q} \in \mathbb{R}^d$, all similarities can be computed using $\mathbf{d} = X\mathbf{q}$, which is a computation that is fast in modern CPU/GPUs using optimized linear algebra software (BLAS, LAPACK, cuBLAS, etc.). Often X is sparse (i.e BoW encoding), so that $\mathbf{d} = X\mathbf{q}$ can be evaluated in $O(k)$, with k being the number of non-zero entries in X, using sparse matrix multiplication. This is equivalent to an inverted file lookup.

2.4.3 Histogram-oriented metrics

While the distances described in the previous section can be applied to any vector representation of the image content, one of the most popular image representation for CBIR is the BoW histogram of SIFT features, described in Section 2.2.2. It has been shown in the literature that specifically designed metrics for histogram representations can outperform traditional L_2-*distance*, as shown in [135]. In this thesis we only use the cosine similarity on L^2-norm vectors, due to its simplicity and fast computation. However, in this section we introduce three popular metrics to compare histograms: Histogram intersection, χ^2-distance, and the earth mover's distance; that can potentially be applied on the histogram representations based on CNN local features proposed in Chapter 4 to improve the overall system performance.

2.4.3.1 Histogram intersection

The histogram intersection was first proposed for colour histogram representations [120]. This distance is especially suitable for partial matching between images. In the case of BoW histogram, only the local features in common between two images are taken into account. The distance is robust to occlusions and clutter [23]. Suppose $\mathbf{x}$ and $\mathbf{y}$ are two histogram vectors of d bins describing two different images, the histogram intersection distance is defined as:

$$D_H(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{d} \min\{x_i, y_i\}. \tag{2.21}$$

2.4.3.2 χ^2-squared distance

The χ^2-distance between $\mathbf{x}$ and $\mathbf{y}$ is given by the formula:

$$D_{\chi^2}(\mathbf{x}, \mathbf{y}) = \frac{1}{2} \sum_{i=1}^{d} \frac{(x_i - y_i)^2}{x_i + y_i}. \tag{2.22}$$

This distance takes into account that the difference between large bins may be less important than the difference between small bins by directly reducing the effect of the large bins. It can be derived by taking the Taylor series expansion of the Jensen-Shannon divergence between categorical probability distributions x_i and y_i and approximating using only the first term. Even though it can provide better results than the commonly used Euclidean metrics [63], it is slower to compute and presents issues when $x_i + y_i = 0$.

2.4.3.3 The earth mover's distance

The earth mover's distance (EMD [104]) (aka the Wasserstein metric), in contrast to the other introduced distances, takes into account the cross-bin relationships. It is defined as the minimal cost that must be paid to transform one histogram $\mathbf{x}$ into the other $\mathbf{y}$. Generally, it is too computationally demanding for practical use in CBIR systems.

2.5 Measuring retrieval performance

The quality of a retrieval system can be estimated by inspecting the first retrieved images for a particular query, which is always a good practice to check whether the system is properly functioning. However, a quantitative evaluation is necessary to effectively determine the performance of a system, and to allow comparison with other methods. Usually, measures such as *precision*, *recall*, and *mean Average Precision* are computed to evaluate the performance of a retrieval system on a particular dataset. Precision is defined as the fraction of the documents retrieved that are relevant to a particular query

$$precision = \frac{\#(\text{relevant items retrieved})}{\#(\text{retrieved items})}. \tag{2.23}$$

Recall is defined as the fraction of relevant documents successfully retrieved by the system

$$recall = \frac{\#(\text{relevant items retrieved})}{\#(\text{relevant items})}. \tag{2.24}$$

These measures are computed on a fixed set of relevant/non-relevant values. *Average Precision* (AP) is commonly computed to estimate the quality of an ordered list. This measure computes the average under the *precision-recall curve*, generated by computing a set of precision recall values at different rank positions. Practically, it is computed by the following expression:

$$AP = \sum_{i=1}^{n} p(i)\Delta r(i), \tag{2.25}$$

where $p(i)$ is the precision at i-location of the rank, and $\Delta r(i)$ the increment in the recall between positions i and $i-1$, and n the number of retrieved elements.

An ideal retrieval system would return all relevant images at the top positions

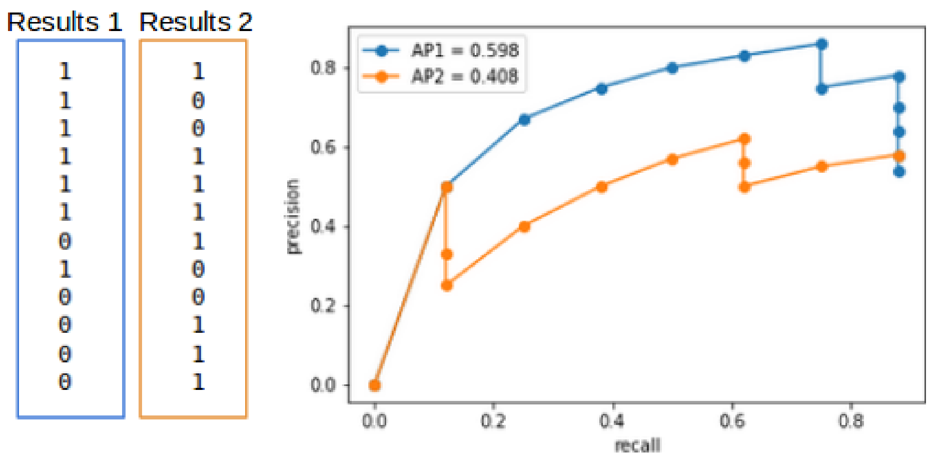


Figure 2.17: *Precision-recall curves* associated to two different ranked list. The list contains a binary label indicating whether a retrieved image is relevant (1) or not (0) for a particular query. For the same amount of relevant information, the first list achieves higher AP than the second list since relevant information is located in the top ranked positions.

having $AP = 1$. However, precision tends to drop with the number of retrieved elements. Figure 2.17 shows how the AP measure favors systems that rank relevant results at the top positions, allowing the comparison of different retrieval methods.

On a particular dataset with N queries, the performance is measured with *mean Average Precision* (mAP), which averages all the AP related to the dataset queries. Metrics such as memory or time required to conduct the search are also important factors to take into account for most real-world problems. A system that achieves $AP = 1$ becomes useless in practice if it requires an infinite amount of time to generate its results. Similarly, a system will not be useful in practice if it requires a large amount of computational resources or memory to store dataset representations.

2.6 Image retrieval benchmarks

Most of the existing work on image retrieval evaluates approaches on publicly available benchmarks to allow development of the field based by objective comparison of techniques. Popular retrieval benchmarks are Oxford Buildings [96], Paris dataset [97],

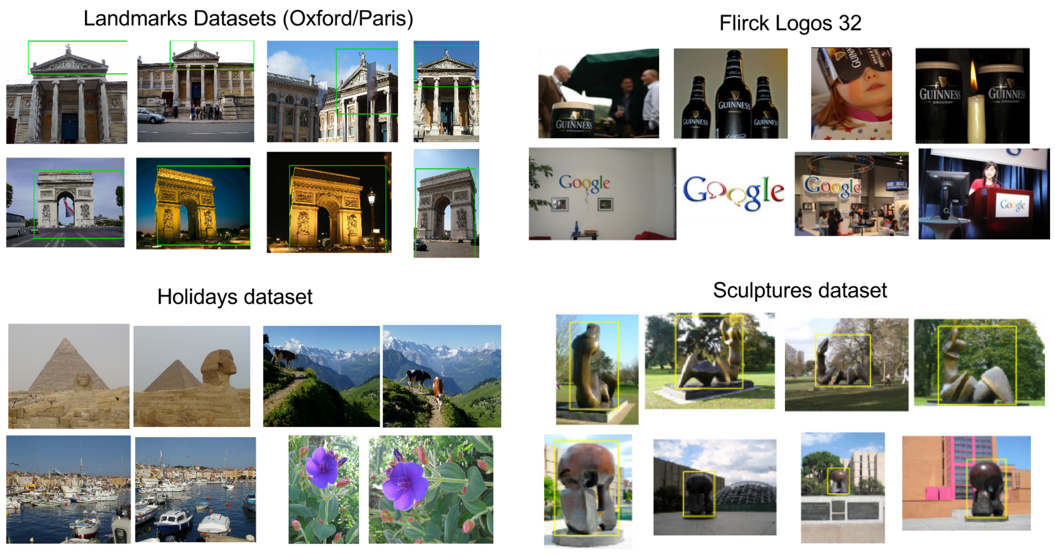


Figure 2.18: Visual examples of different classes in four retrieval benchmarks Oxford, Paris, Holidays, Flickr Logos32 and Sculptures.

Table 2.1: Overview of Retrieval Benchmarks

Name	Number of images	Number of instances	Number of relevant images/instance	Domain
Oxford5k	5063	11	7-220	Buildings
Paris6k	6392	11	87-782	Buildings
Holidays	1491	500	1-12	Landmarks/Objects
Sculptures6k	6340	10		Sculptures
Flickr Logos 32	2240	32	70	Logos

Sculptures [2], INRIA Holidays [51], and Flickr Logos32 [103]. Each dataset is designed such that one specific class defines a particular instance. For example, the "ashmolean" class in the Oxford dataset contains images depicting the Ashmoleam museum under different lighting conditons, camera positions and scales. Figure 2.18 contains some visual examples of different classes for these datasets.

Collecting an adequate dataset for image retrieval is a challenge that usually requires manual annotation to validate the quality of the retrieval system. Due to this fact, the size of the dataset, as well as the amount of accurate annotations tends to be limited. To alleviate the limitation of working with a reduced amount of images, a common practice is to augment the source dataset with a set of distractor images such as Flickr 100K [6] or Flickr 1M [47] that contain images unrelated to the original target queries [96]. This procedure allows researchers to evaluate the scalability of the method as well as the sensitivity to distractors.

We select Oxford and Paris datasets to evaluate and compare our system performance with other approaches:

- **Oxford Buildings [96]** contains 5,063 still images, including 55 query images of 11 different buildings in Oxford. A bounding box surrounding the target object is provided for query images. An additional 100,000 distractor images are also available for the dataset. We refer to the original and extended versions of the dataset as Oxford 5k and Oxford 105k, respectively.

- **Paris Buildings [97]** contains 6,412 still images collected from Flickr including query images of 12 different Paris landmarks with associated bounding box annotations. A set of 100,000 images is added to the original dataset (Paris 6k) to form its extended version (Paris 106k).

2.6.1 Generic instance search benchmarks

Most of the available datasets target a specific domain (buildings, sculptures, logos), which can bias retrieval methods towards a particular visual pattern. A common approach consists of evaluating the same pipeline over different domains, which adds additional complexity to the evaluation because different datasets contain different configurations (ground truth annotations provided in different formats, different image resolutions, different folder organization; which requires extra preprocessing to adapt the data to the main retrieval pipeline). A generic instance search system should be able to find objects of an unknown category that may appear at any position within the images.

In this thesis we evaluate our system in two generic datasets, TRECVID Subset (Chapter 4), and INSTRE dataset (Chapters 5 and 6), Table 2.2 shows a numeric overview of this two datasets.

[6]http://www.robots.ox.ac.uk/ vgg/data/oxbuildings/

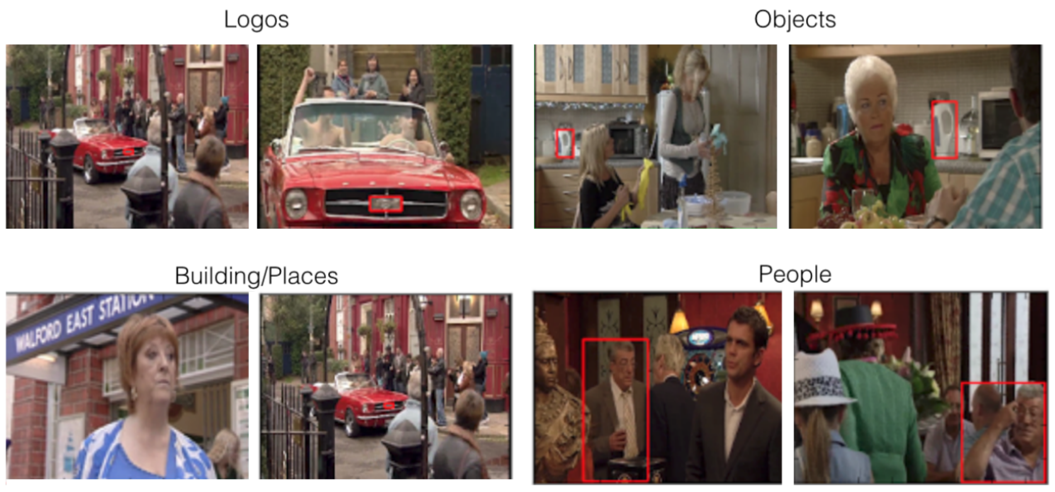


Figure 2.19: Query examples for TRECVID Instance search task. Instance queries (represented by the bounding box) are diverse: they can be logos, objects, buildings or people. The location and scale of the instances is also diverse.

TRECVID [116] is an international benchmarking activity that encourages research in video information retrieval by providing a large data collection and a uniform scoring procedure for evaluation. The Instance Search task at TRECVID consists of finding 30 particular instances within 464 hours of video (a total of 224 video files, 300GB). A common procedure to deal with videos is to perform keyframe extraction, representing the videos a a set of static images to apply on them CBIR techniques. For example, in our 2014 participation [81], the image dataset contained 647,628 image frames (66GB) by extracting 0.25 frames/second of the videos. One of the limitations of this dataset is that it is not exhaustively annotated. For this reason, in our 2014 TRECVID participation [81], as well as in the Chapter 4 of this thesis, we worked with an annotated subset using the ground-truth annotations of TRECVID 2013. We called this dataset "TRECVID Subset" is used in Chapter 4:

TRECVID Subset includes 30 queries and provides 4 still images for each of them, including a binary mask of the object location. Figure 2.19 shows examples of queries for an instance search dataset. We use a subset of TRECVID dataset (keyframes were extracted at 0.25 frames/second), containing only the keyframes that are relevant to the queries. In total it is composed by 23,614 images.

Table 2.2: Overview of Instance Search Benchmarks

Name	Number of images	Number of instances	Number of relevant images/instance	Domain
TRECVID Subset	23,614	30	840 - 3,100	Objects/People/Places
INSTRE	23,070	200	$\sim$ 100	Architectures/Objects

The TRECVID Subset was developed to test the performance of a CBIR system in a more generic and realistic scenario than Oxford or Paris datasets, which are specific to landmarks. However, it is a custom and non-public dataset, which makes it hard to compare our proposed CBIR approach with other existing work. For this reason, in Chapers 5 and 6 we consider the recently proposed INSTRE dataset [133].

INSTRE is a public dataset specifically designed for generic instance retrieval, allowing easy comparison with other recent CBIR systems. The dataset consists of 23,070 manually annotated images, with bounding boxes depicting the location of the instances. It consists of 200 different classes organized into three main categories: architectures (buildings and sculptures), planar objects (designs, paintings and planar surfaces), and different objects (toys and irregularly-shaped instances). Each category contains approximately 100 images, ensuring high intra-class variation within all

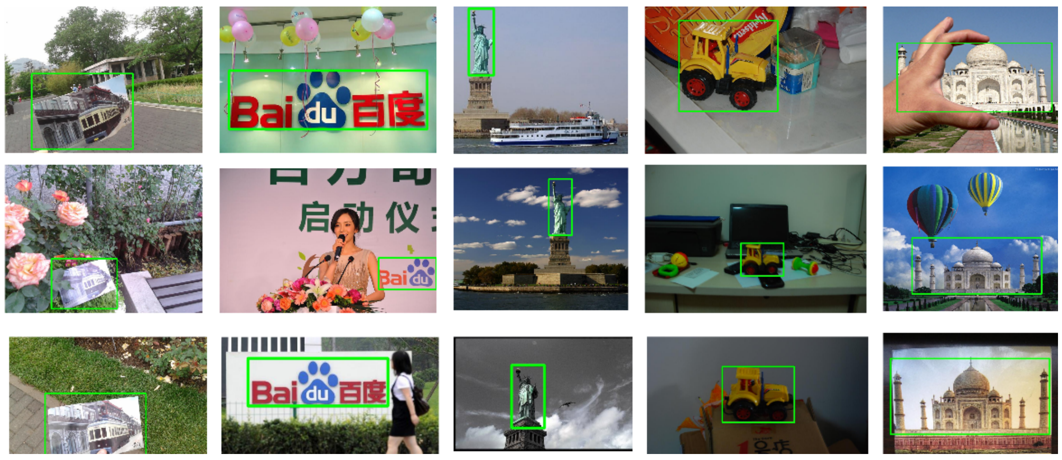


Figure 2.20: 5 different instance-classes in the INSTRE dataset. The dataset provides high intra-class variability in terms of scales and relative position of the objects, as well as backgrounds and occlusions. The dataset domain covers a range of concepts from sculptures, buildings to different kind of objects.

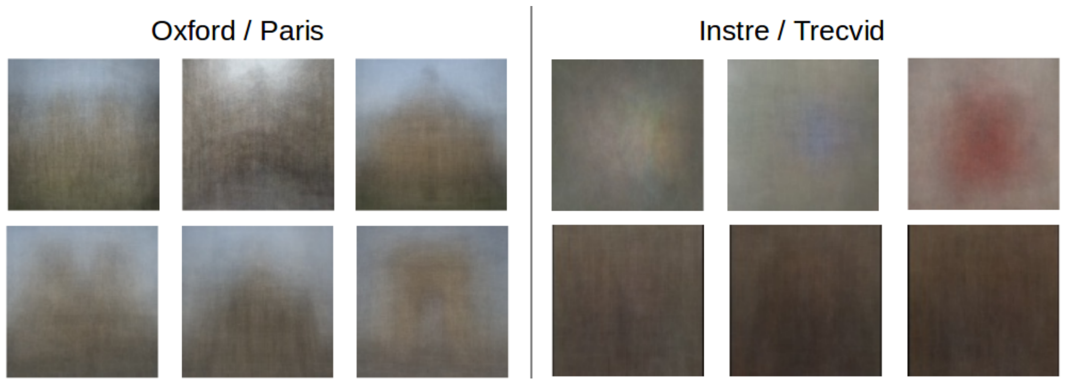


Figure 2.21: Visualization of the intra-class variability. Each image contains an average of up to 200 randomly selected images depicting the same instance. Landmark datasets located in the left (Oxford first row, Paris second row) tend to generate averaged images with relatively clear contours, which denotes limited intra-class variability. General instance dataset, located in the right, generate a more homogeneous averaged images, which shows higher variability of position, background, scales and light conditions of the query instances. (best viewed in colour).

relevant images associated to a query. Figure 2.20 contains some examples of classes, as well as some of their visual variations within a class.

Compared to earlier introduced retrieval benchmarks (e.g. [96, 97, 2, 51, 103]), these two datasets present a more realistic and challenging instance search scenario, where query instances belong to different domains appearing in a wide range of positions, sizes and backgrounds. Figure 2.21 shows the comparison of the intra-class variability for the landmarks datasets and a general instance search dataset. This visualization technique is introduced by Wang and Jiang [133] to show the high instra-class variability of the INSTRE dataset. In the case of INSTRE and TRECVID, the mean image per class is roughly a homogeneous field that cannot be easily recognized by humans. However, in the landmark datasets (Oxford and Paris), it is possible to distinguish relatively clear contours in the averaged images, since the query instance tends to appear in the centre, surrounded by similar patterns.

2.7 Summary

In this chapter we have reviewed “hand crafted” and “deep learning” models to generate image representations. Most of the “hand crafted” traditional approaches are based on SIFT, a local descriptor specifically designed to be robust to scale rotation and lighting variations. While SIFT generates many descriptors associated with interest regions of an image, it is necessary to compress the local information into a global representation to reduce both dataset storage and retrieval time in retrieval scenarios.

The three main approaches were described to encode the local information: BoW, VLAD and FV. All of them are based on generating a visual vocabulary, with the main difference that VLAD and FV encode first and second order statistics, while BoW encodes a histogram with a number of assigned features per cluster, making VLAD and FV a more powerful representation at the price of losing the sparsity obtained in BoW.

We have described the composition of deep learning models, focusing specifically on convolutional neural network (CNN) models, as explored in this thesis. The models are structured in a hierarchical composition of non-linear features that are learned in a supervised fashion to solve a particular task (such as image classification or regression). In these models, the feature learning and feature aggregation steps are simultaneously optimized. This contrasts with the “hand-crafted approach,” where these steps are independent: local feature descriptor generation is an engineered process (such as SIFT) and the aggregation step is an unsupervised method (such as BoW, VLAD or FV).

We have introduced some commonly used similarity metrics/measures to compare image representations. This allows us to generate a ranked list with all the relevant images from the dataset for a particular image query.

Additionally, we have described the main metric used to evaluate a retrieval system (mean Average Precision), highlighting the importance of also considering factors such as query time and memory requirements in the design of a retrieval pipeline.

Lastly, we have introduced common retrieval datasets (Oxford, Paris, Holidays, Sculptures, Flickr Logos) used in the literature, which are oriented over one specific domain, and two more challenging datasets (TRECVID and INSTRE) specifically designed for the task of generic instance retrieval.

2.8 Conclusions

Effectively capturing local structure in images is a key step in the construction of an image representation. SIFT does this by means of an approximation of the Laplacian kernel and finding the local extremas. Then, it encodes statistics regarding the orientations within a particular region and scale. Similarly, the convolutional layers of the CNN models act as dense local feature extractors. In comparison, SIFT can be seen as a shallow convolutional network composed of a single layer of convolutional filters with fixed weights that operates at multiple image scales and a local "pooling" operator that computes a histogram of responses over local patches. However, the filters of a CNN model are trainable so they can automatically adapt to a particular task. In addition, the CNN hierarchical structure of using multiple layers with non-linearities allows it to capture a progressively more semantic representation of the images than only using a shallow one layer structure.

In this thesis we will explore the information from convolutional layers from a CNN to evaluate whether derived local descriptors can benefit from the high dimensional and sparse representation obtained from a BoW encoding, widely used when using SIFT local image descriptors.

Although sophisticated similarity measures have been proposed in the literature, in this thesis we focus in the cosine similarity due to its simplicity and efficient computation and, thus, applicability for large scale image retrieval.

Finally, and as discussed in Chapter 3, whilst CNN methods that have been explored in different retrieval benchmarks achieve remarkable results [39, 100], it is unclear how those methods generalize to more challenging scenarios. In this thesis we focus in developing a general instance retrieval system. Then, we target evaluating its performance in common retrieval benchmarks as well as those specifically designed for instance search task such as the TRECVID or INSTRE datasets.

Chapter 3

Literature Review

3.1 Introduction

Content-based image retrieval (CBIR) has been a long standing research topic in the computer vision community since early 1990. In Chapter 2 we introduced the first approaches for CBIR based on global image representations, generating visual cues based on texture, colour, or shape features [119, 91, 59]. We discussed that almost a decade later, with the emergence of local invariant features such as SIFT [79] and the borrowing of the bag-of-words (BoW) from the text community, the dominant approach consisted of the aggregation of several local features via BoW followed with spatial verification steps to further filter the retrieved results [114, 96]. However, since the introduction of AlexNet for image classification in 2012 [64], deep learning based methods have dominated in many computer vision tasks. Now, the trend is to directly learn the suitable image representations from data for the end task.

In this chapter we provide the reader with a review of different techniques to build image representations from existing CNN architectures for image retrieval. We describe from the first approaches based on using off-the-shelf CNN models trained for classification and extend this to the latest models specifically tuned for retrieval. Figure 3.1 summarizes the performance in the Oxford dataset (described in Section 2.6) of some of the works discussed in this chapter, organized in the four

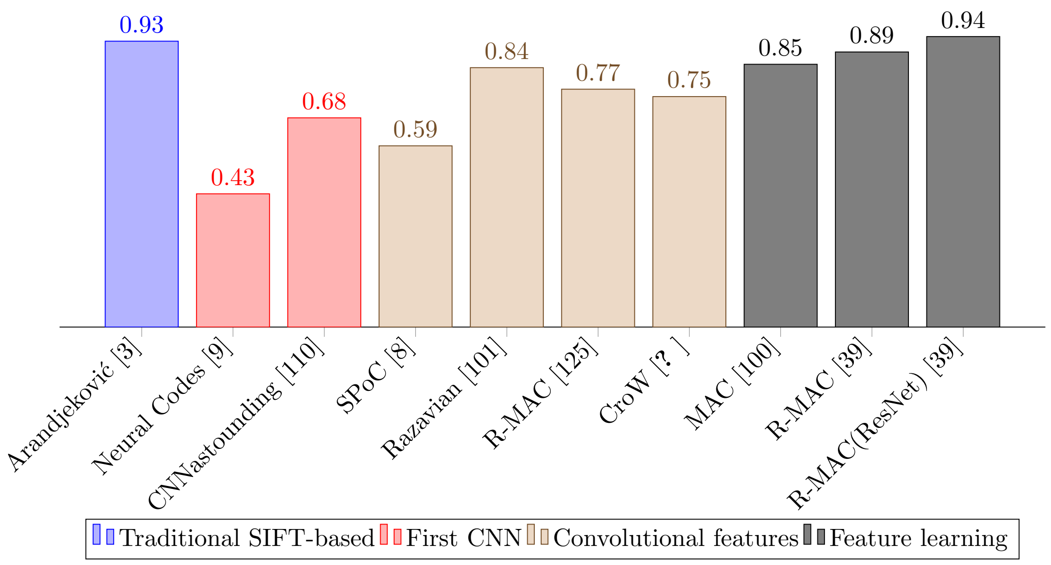


Figure 3.1: Performance comparison (mAP) of different CBIR methods on the Oxford dataset.

main blocs:

- Traditional SIFT-based approaches. In particular, in Figure 3.1 we only show the best performing approach for Oxford. Section 3.2 provides a brief summary of the approaches introduced in Chapter 2.
- The first CNN approaches, discussed Section 3.3, where spatial search greatly contributes in boosting the system performance.
- Works exploiting convolutional features, discussed in Section 3.4, where a variety of aggregation methods, weighting schemes, or region encoding approaches can be applied.
- Lastly, some of the most recent works performing fine-tuning of the CNN model for the end task of retrieval, discussed in Section 3.5.

The remained of the chapter contains the summary in Section 3.6, and conclusions in Section 3.7 drawn from the different work presented and analyzed during this chapter.

3.2 Traditional SIFT-based approaches

In this section we provide a brief summary of different handcrafted methods for the task of CBIR. A more detailed explanation of SIFT, and the different aggregation techniques (BoW, VLAD, and FV) is provided in Chapter 2.

Most retrieval systems are based on the aggregation of local invariant features to produce global image representations. Typically, a keypoint detector such as Hessian Affine detector [85] provides the relevant location within an image from where local descriptors (SIFT [79], RootSIFT [3]) are computed. Local features are aggregated into a fixed-length representation using bag-of-features encoding with large vocabularies [96]. Rankings are generated using term frequency inverse document frequency (tf-idf) scores computed efficiently via an inverted index [114, 90, 96].

Since information might be lost in the quantization process, some works focused on improving the BoW representation. Philib and Chum [97] proposed a soft version of BoW encoding, where local descriptors could be assigned to more than one visual word. Hamming Embedding was proposed in [51] where the visual word associated with each local feature is complemented with a binary code indicating the approximate location of the local descriptor in its Voronoi cell. Other popular aggregation methods such as VLAD [54] or Fisher Vectors [95] have been proposed, where second-order information about the quantization space of local descriptors have been exploited to generate more informative descriptors. However, those methods generate high-dimensional dense representations that are often required to be combined with compression methods [**?** 95, 53] to produce a compact representation at the cost of reduced accuracy.

In practice, state-of-the-art retrieval solutions [27, 3, 26] rely on an *initial stage* that consists of the results returned by a BoW-based system, followed by a second stage of *spatial verification*. This second stage, which is computationally more demanding, is typically used to filter retrieved images whose visual words are not spatially consistent with the words in the query image. The common procedure consists in estimating an affine transformation between the query image and result image via RANSAC scoring [33, 96]. Then, candidate images are re-ranked by the total number of inliers, which are the matching points between images that fit the predicted transformation. This procedure helps to improve the precision of the retrieved results. To improve recall, query expansion methods are used to issue new queries based on the the representations of the top retrieved results [3, 27, 26]. When query expansion is used to obtain a better representation for the image queries, database-side augmentation consists in obtaining better representations for the database images, using features from other images of the same object. Arandjeković and Zisserman [3] achieve a top performance of 0.929 and 0.910 mAP in Oxford and Paris datasets by adding different post-processing steps (re-ranking, QE, and database side augmentation) on top of the initial search. This system achieves the current state-of-the-art in these two datasets as shown in Table 3.4. However, their approach has the downside of being highly a computationally demanding solution (it requires the construction of a similarity graph for the full dataset). It is unclear how the approach can be generalized to non landmark focused datasets.

3.3 First CNN approaches for retrieval

Activations from the last fully connected layers from the AlexNet network proposed by Krizhevsky *et al.* can be used as generic image representations with potential applications for image retrieval. For more details of the AlexNet architecture, see Section 2.3.4.1.

It was observed in [64, 32, 137] that similar images generate similar activation vectors in the Euclidean space. This finding motivated early works in studing the capability of CNN models for retrieval, mostly focused in the analysis of fully connected layers extracted from a pre-trained CNN clasification model AlexNet [9, 110, 38].

3.3.1 Fully connected layers

Fully connected layers are those where each neuron in the layer is connected to all the neurons of the previous layer. The activations of a fully connected layer are given by:

$$x_i^l = g\left(\sum_{j=1}^{N} w_{ji}^{(l)} x_j^{l-1} + b_i^{(l)}\right), \tag{3.1}$$

where x_i^l represents a neuron in the layer l, g is activation function (eg. sigmoid, ReLU, or softmax), and w_{ji} the weight associated to connect the neuron i in the layer l to the neuron j in the layer $l-1$, b_i the associated bias, and N the number of neurons in the previous layer.

Fully connected layers contain the majority of the parameters in the network. For instance, in the AlexNet architecture represented in Figure 3.3.2, the output layer of 1000 dimensions is a fully connected layer connected a layer 7, with 4096 dimensions. This means that each neuron in the output is a linear combination of the 4096 neurons of the previous layer, which then goes though a non-linearity function (softmax). So the total number of parameters can be computed considering the total number of weights and biases: $1000 \times 4096 + 1000 = 4097000$ parameters for the output layer [1].

3.3.2 Neural codes

Babenko *et al.* [9] conducted the first quantitative study of using the outputs of the fully connected layers for retrieval. The authors evaluated the top-3 layers from their own implementation of AlexNet (layers 7, 6, and 5 illustrated in Figure 3.2) as a global image representation, which they named *neural codes.* Euclidean distance on L^2-normalized neural codes was used as a metric function to rank the images. This metric is equivalent to cosine similarity on the L^2-normalized representations, which can benefit from fast and parallel (e.g. GPU) matrix multiplication methods. Two

[1]Even more parameters are required for a layer below this (Layer 7 with $4096^2 + 4096$), and more again for the first fully connected layer 6 connected to a convolutional layer.

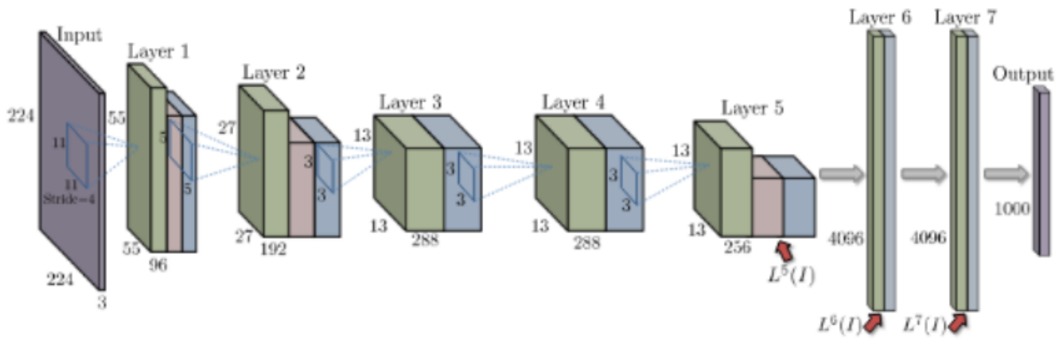


Figure 3.2: Krizhesvsky's architecture model used in Babenko [10]. The model consists of five convolutional layers, each one followed by max-pooling layer and non-linearity layer ReLu; two fully connected layers each one followed by a ReLu and a last Softmax layer with the classification output. In their study they consider the output of the last two fully connected layers L^7, L^6 (both with dimension 4096) and the last convolutional layer L^5, flattened into a 9216-dimensional vector. Input image was fixed to 224×224. Image source [9].

versions of the AlexNet model were considered: the first was pre-trained on ImageNet and the second was a fine-tuned version of the first, using a custom classification dataset that consists of popular landmarks. This second model was considered for learning more suitable neural codes by means of domain adaptation [137], since the landmark dataset images were semantically more similar to some of the target retrieval dataset than the original ImageNet dataset.

The neural codes generated were a high-dimensional dense vector. In particular, 4096 dimensions for layers 7, 6 layers, and 9216 in layer 5. Storing such a high dimensional and dense vector is memory demanding, and represents a limitation in terms of computation and memory requirements at query time when the size of the dataset is large. To alleviate this challenge, the authors investigate two approaches to perform dimensionality reduction: they used principal component analysis (PCA), which is an unsupervised technique that finds a new orthogonal space in which the direction of greatest variance of the dataset lies on the first axis of the new space, the second greatest in the second axis, and so on; and a supervised method from discriminative metric learning for large-margin dimensionality reduction [112]. In this second method, a projection matrix W is discriminatively learned by using triplets of images. As a result, W is learned such that neural codes are projected into a lower dimensional space where distances between matching and non-matching pairs are preserved.

They found experimentally found that the lower fully connected layer 6 was the best performing across different retrieval benchmarks. Contrary to latest findings in the literature [125**?** , 8, 101], descriptors from the lowest layer considered, layer 5 (corresponding to the activations of the last convolutional layer), did not perform as well as the fully connected layer 6. The image representation derived from the convolutional layer 5 was constructed by directly flattening the activations of the tensor volume into a single representation. Since each neuron contains local information of the original image, as described later in Section 3.4.1, this procedure is not well suited to constructing the final representations because the representation is location sensitive.

It was also found that performance of compact CNN codes was almost unaffected until the dimension was reduced to 128-256, while dimensionality reduction methods drastically affect the performance of hand-crafted features. At low dimensions, CNN descriptors outperformed hand-crafted ones. Large-margin dimensionality reduction performed better than the unsupervised PCA, with the drawback of requiring annotated data with matching and non-matching image pairs. Interestingly, they found supervised dimensionality reduction applied to fine-tuned neural codes did not generate any gain. Presumably this is because the network re-training and the discriminative reduction were learned using overlapping training data, making the information learned from the two different methods redundant.

Domain adaptation did generate descriptors more suitable for retrieval in Oxford and its extended version Oxford105k [96] (this last dataset adds 100,000 distractor images from Flirckr to the original Oxford dataset, as described in Section 2.6). However, the gain in performance was not as significant as in the Holidays dataset [51], where the dataset depicts a set of vacation photographs based on scenes and objects. The approach did not improve performance in UKB benchmark [90], where the domain is significantly different to landmarks, containing indoor photographs of different objects. In that case, the ImageNet original training data is more suitable.

If training data relevant to the target image retrieval domain is available, it is possible to learn better image representations by fine-tuning the pre-trained model. However, this strategy assumes two things: first, the existence of such relevant training data and, second, the existence of class image labels for it, which is not usually true and represents a challenge. Discarding the fine-tuning approach (discussed in Section 3.5), another way to improve performance of pre-trained CNN representations is by directly addressing some of their limitations.

3.3.3 Spatial search

Razavian *et al.* [110] explored the usage of *fc* activations for retrieval including a spatial search strategy in their pipeline: a multi-scale sliding window approach to extract multiple descriptors per query and target image. In their approach, each image crop was resized to 221×221 and described by its associated 4096 dimensional vector from the first fully connected layer of the *Overfeat* network [109] pre-trained on ImageNet. Features were post-processed using L^2-normalization, followed by a PCA-whitening to reduce the dimensions to 500 and a second round of L^2-normalization. Cosine similarity was used as a ranking score. The final score per image considered the cross-matching similarities between all regions at different scales.

By processing the images at multiple regions and scales they significantly improve the performance of the original off-the-shelf CNN representations, without any further training or fine-tuning, on multiple retrieval benchmarks: while the neural codes proposed by Babenko [9] achieved 0.435 mAP in Oxford dataset (0.545 mAP after network fine-tuning), the spatial search strategy boosts the performance up to 0.680 mAP. However, generating several descriptors for each image makes dataset storage many times more expensive, as well as compromising query time, making the approach less scalable to large datasets.

Table 3.1: Performance comparison of first CNN approaches in the Oxford5k dataset. Performance of the pre-trained fully connected layers *fc* can be enhanced by fine-tuning the CNN network and/or including spatial search strategy.

Method	Description	Dim	Oxford5k
Neural Codes [9]	pre-trained *fc*	128	0.433
Neural Codes [9]	fine-tuned *fc*	128	0.557
CNNastounding [110]	pre-trained *fc* + spatial search	4-15k	0.680

3.3.4 Aggregation of region features

Still addressing the limitation of the lack of local information of the fully connected layer, a more scalable approach to deal with local information was proposed by Gong *et al.* in [38] where they proposed a Multi-Scale Orderless Pooling (MOP-CNN) of region-based CNN representations. In their approach, images were processed at 3 different levels, extracting a series of regions or patches per level. At each scale, the regions were extracted using a fixed grid of locations. Each patch was then represented with its *fc7* feature (the output of the first fully connected layer from the AlexNet model implemented in *Caffe* [57]), obtained after resizing the patch to 256×256 and forwarding it to the network. For each scale considered,

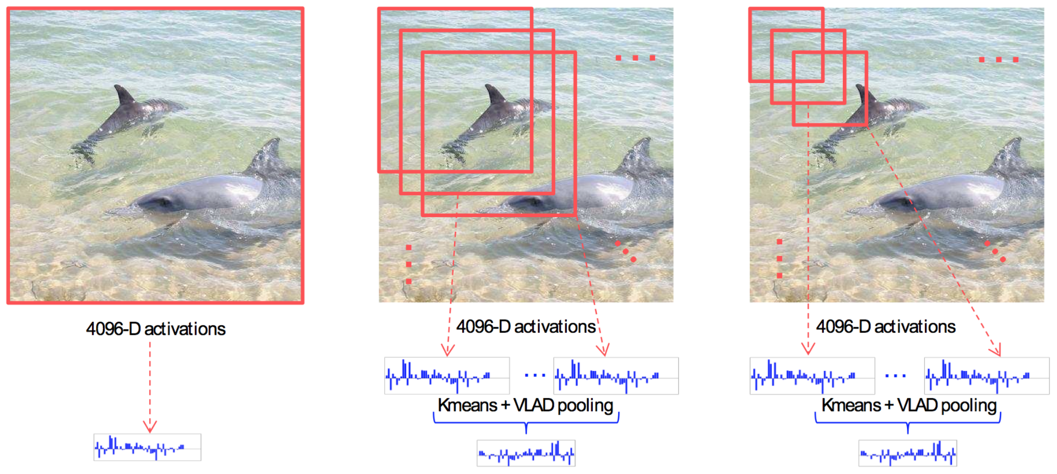


Figure 3.3: Overview of multi-scale orderless pooling for CNN activations (MOP-CNN) [38].

descriptors from different regions were pooled via VLAD encoding [54]. Figure 3.3 shows the multi-scale encoding followed for an image. Without further dimensionality reduction, this method would generate extremely large representations consisting in the concatenation of the three considered scales: the first one of 4096 dimensions, and the next two of 50,000 after applying VLAD encoding, which results in a 54,096 dimensional dense vector. A second round of PCA dimensionality reduction was applied to generate the final representations of 2048 dimensions. Reported results push the performance on the Holidays dataset from 0.701 mAP using 4096 dimensional vector at global scale to 0.802 mAP using the multi-scale global representation of 2048 dimensionality.

3.4 Convolutional features for retrieval

Descriptors from fully connected layers of a pre-trained CNN network in ImageNet achieve competitive performance in comparison with hand-crafted descriptors at low dimensions [9] or directly addressing the lack of invariance of CNN representation by extracting multiple representations in a sliding window manner at multiple resolutions [110, 38]. However, local characteristics of objects at the instance level are not well preserved at those layers, since the information in them is biased towards the final classification task and spatial information is completely lost (each neuron in a fully connected layer is connected to all neurons of the previous layer). Recent works show that spatial max and sum pooling [101, 125, 8**?**] of feature maps output by intermediate convolutional layers is an effective representation, and that higher performance can be achieved compared to using the *fc* layers.

3.4.1 Convolutional volumes as local image descriptors

Each neuron in a convolutional layer responds to a specific region in the original image. The size of this region is known as the *aperture size* or a *receptive field*, and its size depends on the kernel sizes and strides from previous layers. The theoretical receptive field of a neuron in a particular layer $A_l \in \mathbb{R}$ can be computed following the recursive formula:

$$A_l = A_{l-1} + (K_l - 1) \prod_{j=1}^{l} s_j, \tag{3.2}$$

where A_{l-1} is the receptive field of a neuron in the previous layer and s_j is the stride used at the layer l. A_l increases with l, generating for some architectures (VGG16, GoogleNet, ResNet) theoretical receptive fields that can cover the full extend of the image. However, the *practical receptive field* of a neuron in a convolutional layer is much smaller as shown in [80, 145].

Since each neuron in a convolutional layer responds to a particular area of the image, the output of an activation of a convolutional layer of dimension $H \times W \times D$) can be re-interpreted as a collection of N *local* or *region* descriptors of dimension D, where $N = H \times W$ associated to an original image (illustrated in Figure 3.4), and each of the dimensions of the local descriptor encodes a particular visual pattern.

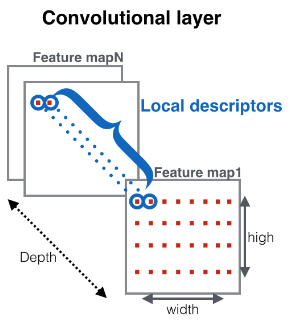


Figure 3.4: Re-interpretation of activation tensor into local descriptors

In this context, *fc* layers can be seen as an aggregation step of the local features contained in a convolutional layer into a fixed length descriptor. The weights of the *fc* layer for off-the-shelf networks are tuned to aggregate this local information to generate representations suitable for its final task (i.e. image classification).

Figure 3.5 represents the taxonomy of the different methods based on convolutional features, where some works explore different aggregation strategies, some others apply a different weighting schemes on the volume of convolutions, and/or

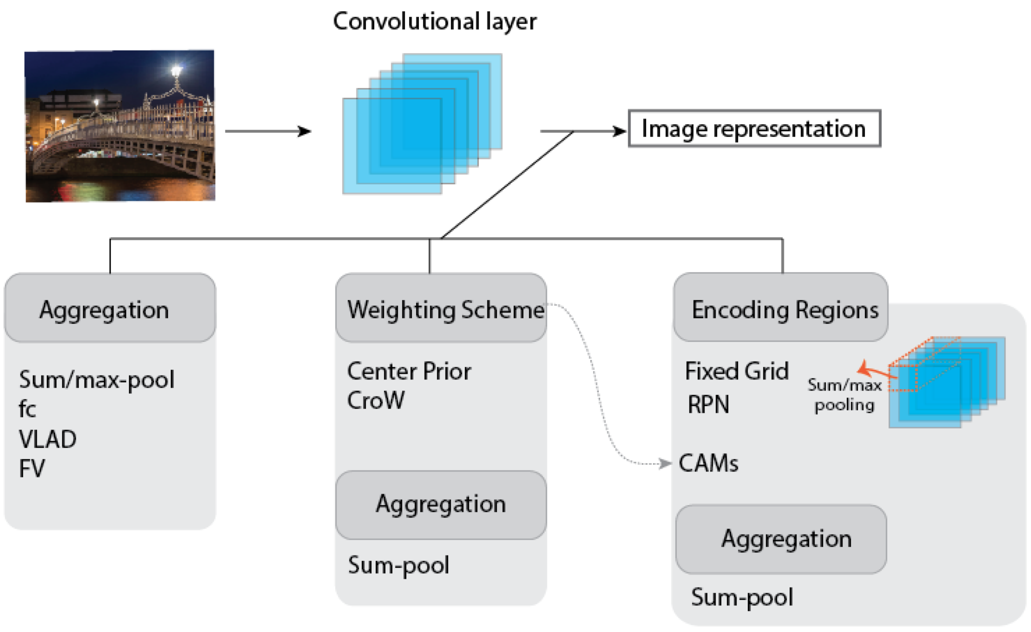


Figure 3.5: An image is represented by a set of local convolution descriptors. Descriptors can be aggregated, they can be weighted prior to aggregation, or region descriptors can be derived and then encoded into a single representation.

some other works perform a region analysis derived from the convolutional layer selected. We describe the most relevant works in the following subsections.

3.4.2 Local feature aggregation

One of the simplest approaches to aggregate the local CNN descriptors of a convolutional layer consists of max or sum pooling all the activations along the feature maps. With this procedure it is possible to aggregate all the information contained in the volume of $H \times W \times D$) into a single representation of dimension D. This approach was first proposed by Babenko [8], which boosted performance in the Oxford dataset from 0.433 mAP (obtained from their earlier proposed neural codes) to 0.589 mAP. In the Holidays dataset, this sum-pooling of convolutional features achieved 0.802 mAP, which is the same result as obtained with MOP [38], consisting of multi-scale processing, VLAD encoding and multiple PCA dimensionality reduction steps; showing the effectiveness as well as simplicity of the method.

In their work they evaluated different strategies to aggregate descriptors from the last convolutional layer from VGG19 (*pool5*). For the aggregation step, as part of the evaluation of direct sum/max pooling, they tested more sophisticated methods such as Fisher vector encoding [95] and T-embedding [56]. They found that the best pooling strategy was sumpooling in combination with a simple post-processing step on the pooled features, consisting of L^2-normalization, applying PCA and whitening, and L^2-normalization again. This simple setting performed surprisingly well and improved performance for compact global descriptors on different standard datasets. They claimed that encoding steps such as FV or T-embedding used in SIFT features were not necessary in the local CNN descriptors due to their high discriminative ability and different distribution properties.

However, this claim is not necessarily true. Yue-Hei [138] applied VLAD encoding to deep local features achieving slightly better results than Babenko's [8] (0.59 mAP and 0.82 mAP in Oxford and Holidays dataset using descriptors of size 128). One of the main differences from [8], apart from the aggregation method used, was to study the performance of several convolutional layers: they found that mAP increased when they extracted descriptors from lower and less semantic layers. They also found that the convolutional filters behave differently according to the input size considered: for a particular layer, larger input sizes capture more textual and fine-grained information. In their approach, instead of extracting descriptors from the last convolutional layer as in [8], they experimentally found that the best performing descriptors for VLAD encoding corresponded to the *conv5_1* layer of VGG16. VLAD was used to encode descriptors of that layer at two different image resolutions. The final representation, compacted to 128D via PCA, obtained state-of-the-art results with even lower dimensionality than sumpooled descriptors [8], at the cost of being a more computational demanding solution (i.e codebook computation, multi-resolution processing of queries and target dataset images).

3.4.3 Region-based descriptors

Razavian *et al.* [101] reported a boost in performance in four retrieval benchmarks by using the last convolutional layer (*pool5* from VGG16) in combination with an spatial search strategy, achieving 0.843, 0.879, and 0.896 mAP in Oxford, Paris, and Holidays datasets respectively. They explore the spatial information contained in a convolutional volume to generate a set of region descriptors per image, similar to the Fast R-CNN network for object detection [37]. A particular region in the original image was mapped to the convolutional feature maps. Direct max-pooling was applied to pool all activations contained within a particular region to generate a pooled vector of the same dimensions as filters contained in the convolutional map (i.e *pool5* from VGG16 generates a descriptor of 512D). As an example, Figure 3.6 illustrates the direct spatial pooling on the full convolutional volume and in four different regions.

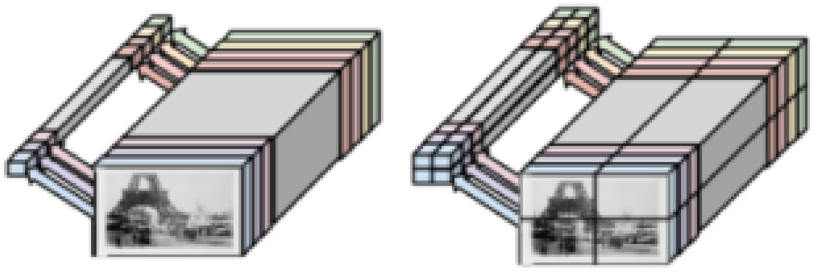


Figure 3.6: Spatial max-pooling. On the left, max-pooling at one global region generate a single 512D representation. On the right, pooling at 4 different regions generate 4 × 512D region-CNN representation. Image source [101].

This pipeline is much more efficient that the one proposed in [110] due to the fact that multiple regions can be generated in a single forward pass of the image through the network. Images are processed at multiple scales, from which multiple region-CNN representations are computed. Region CNN descriptors are not pooled, which allows an exhaustive search of the query at the cost of increasing the memory storage and query time. Since each image is processed at multiple scales (for target and query images), the final distance (computed via cosine similarity) is computed

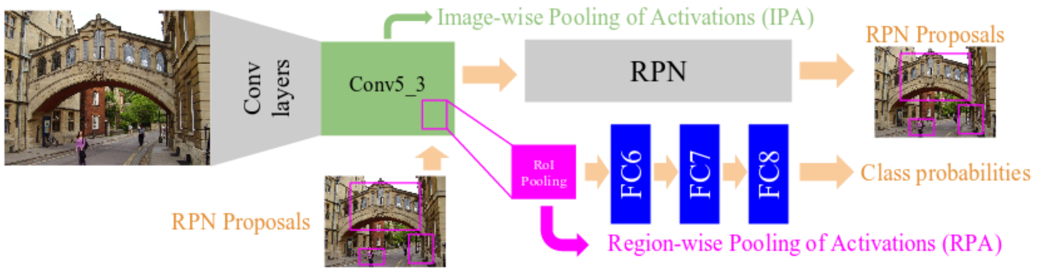


Figure 3.7: Faster R-CNN [102] used for instance retrieval. The base module follows the VGG16 structure where all layers above *conv5_3* have been replaced by a second module. The second module is composed of two branches, one RPN that generates the candidate regions, and a second classification network composed of three fully connected layers that take a descriptor associated with each region of interest (spatial max-pooling of a region in *conv5_3*) and predicts a classification score for different object classes. Image source from [106].

as follows: for each scale, the minimum distance between query sub-patch and target image sub-patch is computed. Then, the final distance is computed by taking the minimum of all scale distances obtained.

Focused on exploring the advantage of processing different regions of the image independently, Salvador *et al.* [106] propose to use a fine-tuned version of an object detection network. In particular, they use the Faster R-CNN [102] which is a network composed of a base module, which is a fully convolutional CNN (i.e VGG16 architecture), and a top module composed of two branches: one branch is a Region Proposal Network that learns a set of window locations, and the second one is a classifier (composed of three fully connected layers) that learns to label each window as one of the classes in the training set.

A baseline ranking is produced by performing spatial sum-pooling on the last convolutional layer of the base model (identified as *image-wise Pooling of Activations* in Figure 3.7). Then, for the top-100 images, the tuned Fast R-CNN network generates a set of candidate regions per image with a probability score associated for all queries in a dataset. The re-ranking is done for a particular query by taking the region with maximal probability associated with it and re-sorting the list.

With this approach they achieved very competitive performance of 0.710 and

0.798 mAP for the Oxford and Paris datasets (0.786 and 0.842 mAP when doing query expansion [27, 3]). However, the main limitation of this approach is that it assumes that all queries are known and fixed for a particular dataset, which is not a realistic solution for practical applications.

3.4.4 Regional maximum activation of convolutions (R-MAC)

Tolias *et al.* [125] extended Razavian's pipeline [101] by sum-pooling the L^2-normalized region descriptors, called *regional maximum activations of convolutions* (R-MAC) into a single representation. Their baseline system was further improved by re-ranking [96] and query expansion [27, 3], which provided a boost in performance and also allowed them to localize the target instance within the images. Region descriptors were computed at different scales in a fixed grid of locations. Similar to [8, 101], descriptors were post-processed via L^2-normalization, PCA-whitening and L^2-normalization, and then pooled via sum-pooling into a compact representation. The authors claim that cosine similarity on R-MAC descriptors can be seen as a simple kernel that cross matches all possible regions, including across different scales.

With this approach, the performance obtained on the Oxford and Paris datasets was 0.669 and 0.83 mAP (0.773 and 0.865 mAP after re-ranking and query expansion). The overall performance in these two datasets is very similar to that achieved in Salvador's work [106], with the advantage of being more flexible to different types of queries. Reported results for R-MAC, however, are slightly worse than the performance achieved by Razavian [101], although the approach is significantly more efficient: one unique representation (based on pooling individual regions) is required per image; the explicit comparison of an individual region is only considered in the re-ranking step.

3.4.5 Spatial weighting

For image retrieval, and especially for instance search, not all parts of the image are equally important. For instance, in the Oxford and Paris datasets the target

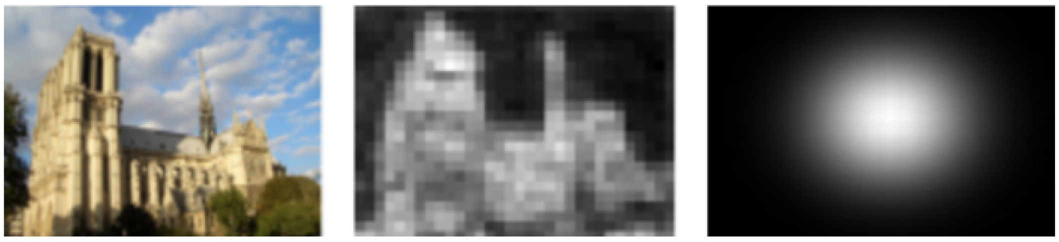


Figure 3.8: CroW spatial weights (center) and center prior (right).

instances are buildings that are usually located in the center of the image. Babenko [8] proposed a simple center prior weighting to provide more importance to the features located in the center of the image. They equally weight each of the dimensions of a particular local convolutional feature using a scalar provided by a Gaussian weighting (see image in the right in Figure 3.8). This simple approach, increased mAP performance from 0.589 to 0.657 mAP in the Oxford dataset, however it did not perform as well in the Holidays dataset where the unweighted sum-pooling achieved 0.802 mAP in contrast to 0.784 mAP after applying the Gaussian weighting. The fact that the weights were fixed regardless of the image content, made the strategy unlikely to generalize well to datasets where the target instances are not located in the center of the image [134].

To provide a more customized weighting, Kalantidis *et al* [**?**] proposed a parameter free weighting scheme based on the activations of the last convolutional layer (CroW). The weighting was composed based on two factors: one based on the "strength" of each local feature, measured by the L^2-norm of each feature; and a second one derived from the channel sparsity. They showed that the sparsity of the feature maps in a convolutional layer contained discriminative information beneficial for retrieval. The weighting derived for the sparsity was applied following the same concept of inverse document frequency as traditional BoW [96] to boost the contribution of rare features. Sum-pooling with the proposed weighting (represented in the central image of Figure 3.8) improved performance with respect to their unweighted features (uCroW), especially on PCA reduced descriptors, achieving 0.682, and 0.797 mAPs in the Oxford and Paris datasets (0.718 and 0.815 mAP after

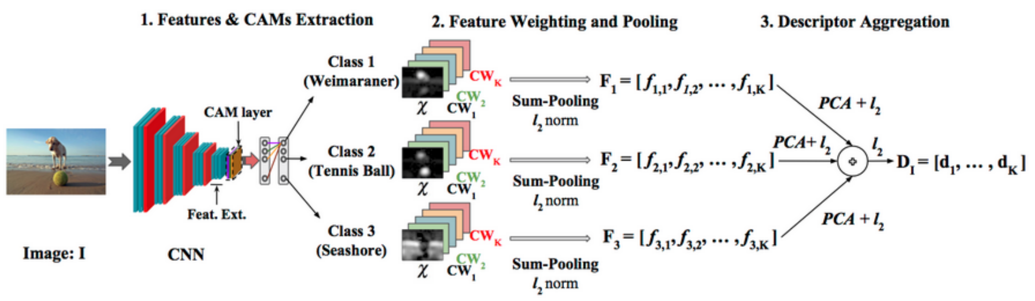

1. Features & CAMs Extraction
2. Feature Weighting and Pooling
3. Descriptor Aggregation
Image: I
CNN
CAM layer
Feat. Ext.
Class 1 (Weimaraner)
Class 2 (Tennis Ball)
Class 3 (Seashore)
Sum-Pooling l_2 norm
$F_1 = [f_{1,1}, f_{1,2}, \cdots, f_{1,K}]$
$F_2 = [f_{2,1}, f_{2,2}, \cdots, f_{2,K}]$
$F_3 = [f_{3,1}, f_{3,2}, \cdots, f_{3,K}]$
$PCA + l_2$
$D_1 = [d_1, \cdots, d_K]$

Figure 3.9: Pipeline for generating CAM-weighted representations. Region descriptors used in R-MAC are substituted by a class descriptors, each one built by spatial weighting the activations of a convolutional layer by the corresponding weighting schemes obtained for one particular class. The class-vectors are L^2-normalized, PCA-whitened, and L^2-normalized again. The final vector is constructed by sum-pooling all the class-vectors into a single representation. Image source [60].

query expansion).

Recently, Jimenez *et al.* [60] proposed a technique that can be seen as a combination of the ideas introduced in CroW [**?**] and R-MAC [125]. They propose using Class Activation Maps (CAMs) [146], which is a technique that can be applied to most of the the state-of-the-art CNN networks for classification to create a spatial map highlighting the contribution of the areas within an image that are more relevant for the network to classify an image as one particular class. This way, several weighting schemes can be generated for each of the classes for which the original pre-trained network was trained (typically the 1000 classes of ImageNet [29]). Each of the weighting schemes can be used in the same way as in CroW to generate different vectors per class. All the obtained class-vectors are then sum-pooled to get a final compact representation, just as in the R-MAC approach. Figure 3.9 shows the schematic for the construction of the final compact vector. With this method, it is possible to achieve very competitive performance of 0.736 and 0.855 mAP on the Oxford and Paris datasets (0.811 and 0.874 after re-reranking of the top 1000 images and query expansion).

Table 3.2: Performance of works exploiting convolutional features. In the case where the method generates multiple region vectors per image, "Global Aggegation" refers to the method applied to the aggregation of the region vectors (with the exception of [85], where it applies directly on the convolutional features). The performance of the methods including re-ranking and query expansion stages are included in parenthesis.

Method	Multiple Regions	Region Aggregation	Weighting scheme	Global Aggregation	Dim	Oxford5k	Paris6k
SPoC [8]	No	-	Yes	sum-pool	256	0.589	-
Ng *et al.* [138]	No	-	No	VLAD	128	0.593	0.590
Razavian [101]	Yes	max-pool	No	No	32k	**0.843**	**0.879**
R-MAC [125]	Yes	max-pool	No	sum-pool	512	0.669(0.773)	0.830(0.865)
CAMs [60]	Yes*	sum-pool	Yes	sum-pool	512	0.712(0.801)	0.805(0.855)
CroW [**?**]†	No	-	Yes	sum-pool	512	0.708(0.749)	0.797(0.848)
Faster-RCNN [106]**	Yes	max-pool	No	sum-pool	512	0.710(0.786)	0.798(0.842)

* One weighting scheme per class (the top N most probable per image). Each region vector is generated by weighting the original convolutional features with N different spatial weights.
† The method does not include re-ranking, only query expansion in the post-processing step.
** Network is fine-tuned using the query images.

3.4.6 Discussion

Table 3.2 summarizes the performance and the main features of the discussed works exploring the applications of convolutional layers of pre-trained CNN models for retrieval. The best performing approach is based on generating multiple region vectors per image without including any final aggregation [101], which means the pipeline is not scalable for large-scale datasets. However, it is possible to leverage the benefit of region-based representations by aggregating them into a compact descriptor via sum-pooling [125, 60] and then including a spatial search on the top-N results (performing a re-ranking and query expansion) to significantly improve performance.

It is notable that CroW [**?**] does not use a re-ranking stage (the method does not generate multiple region-representations) and still achieves nearly the same performance as the methods based on multiple regions vectors. This provides evidence for the importance of exploring better weighting schemes over the convolutional features to better exploit the spatial information encoded in the convolutional layers.

Region-based methods such [101, 125] use a fixed grid of location to generate the region representations. Instead, using an object proposal algorithm would generate more precise regions. In this spirit, Faster-RCNN is explored in [106]. However, the obtained regions from the RPN are only used as the re-ranking step, and they are not aggregated into a global representation, which seems to enhance the performance of the global representation with respect to directly aggregating the original local CNN representations [125]. It is also notable that the region-based methods do not include any weighting scheme (with the exception of [60], where every region *is* a weighted sum-pooled vector of the original convolutional features).

Lastly we note that, where VLAD encoding has been explored to aggregate local representations derived from a convolutional layer [138], it would be interesting to see whether more sophisticated aggregation methods (BoW, VLAD, FV) can be applied to the region-based CNN descriptions instead of applying direct pooling.

3.5 Feature learning for retrieval

Deep learning is a proven mechanism for successfully learning useful semantic representations from data. However, most of the work discussed uses off-the-shelf CNN representations for the task of retrieval, where representations have been implicitly learned as part of a classification task on the ImageNet dataset.

These approaches contain two main drawbacks. The first is the **source dataset** ImageNet from where features are learned. While ImageNet is a large-scale dataset for classification, covering a diverse set of 1000 classes (from airplanes, landmarks to general objects), and allowing models to learn good generic features, it has been explicitly designed to contain high intra-class invariance which is usually not a desirable property for retrieval. The second is in the **loss function** used. Categorical cross entropy evaluates the classification prediction without explicitly taking into account the representational similarity between different instances, which may be desirable in several retrieval scenarios.

3.5.1 Fine-tuning with a classification loss

One simple but effective solution to improve the effectiveness of the CNN features consists of learning representations that are more suitable to the test retrieval dataset by *fine-tuning* the CNN network to perform classification in a new domain.

Babenko [9] improved the performance of off-the-shelf CNN representations by using a pre-trained CNN network on a dataset semantically closer to the Holidays and Oxford retrieval benchmarks. For that, they generated the Landmark dataset[2], by crawling images related to a predefined image queries (top-10,000 most visited landmarks in Wikipedia) followed by manual annotation. The final dataset contains 672 classes and a total of 213,678 images.

The last softmax layer from a pre-trained AlexNet model was replaced by a new softmax layer which mapped the last fully connected layer to a 672 dimensional space where each dimension represented the different class probabilities. The categorical cross entropy loss was used as the loss function and SGD was used to update the weights of the model.

Despite improving performance, the final metric and the layers were different to the ones actually optimized during training. This suggests that the improvements obtained are due to the domain adaptation of the network. Directly optimizing the final image representations for similarity learning leads to better performing descriptors for image retrieval as will be explained in the following sections.

3.5.2 Similarity learning

Following the informal definition provided in Kulis *et al.* [66], similarity learning (also known as metric learning) can be stated as: given an input distance function $D(x, y)$ between two objects $\mathbf{x}$ and $\mathbf{y}$ (eg. the Euclidean distance), along with supervised information regarding an ideal distance, the objective is to construct a new distance function $\hat{D}(x, y) = d(f(x), f(y))$ which is "better" than the original distance function. In the image retrieval context, $\mathbf{x}$ and $\mathbf{y}$ are the raw images. The mapping function f can be formulated as a CNN model. The whole fine-tuning process of the CNN can be cast as a metric learning problem, which is more suitable for retrieval than using a classification loss.

[2]http://sites.skoltech.ru/compvision/projects/neuralcodes/

3.5.2.1 Siamese networks

First introduced in 1994 for signature verification [15], siamese networks have been applied for dimensionality reduction [41], learning image descriptors [18, 111, 139], and face verification [24, 121].

Siamese networks [24, 41, 121] are architectures composed of two branches (each one based on convolutional, ReLU, and max-pooling layers) that share exactly the same weights across each layer. They are trained on paired data consisting of an image pair $(\mathbf{x_i}, \mathbf{x_j})$, where $\mathbf{x}$ represents the representation obtained from a CNN, and $Y_{i,j} \in \{0, 1\}$ represents the binary label indicating if the images belong to the same category or not. The network optimizes the contrastive loss function defined for each pair as:

$$\mathcal{L}(\mathbf{x_i}, \mathbf{x_j}) = \frac{1}{2}\left(Y_{i,j} D(\mathbf{x_i}, \mathbf{x_j}) + (1 - Y_{i,j}) \max(0, \alpha - D(\mathbf{x_i}, \mathbf{x_j}))\right), \tag{3.3}$$

where $D(\mathbf{x_i}, \mathbf{x_j}) = \|\mathbf{x_i} - \mathbf{x_j}\|_2$.

Figure 3.10 shows the behaviour of the loss. Given a particular *anchor* or reference image, a positive example is pushed closer to the anchor in the feature space. For a negative example, if its distance to the anchor is less than a certain margin, then the negative is pushed so that the distance is larger than that margin. We display multiple examples as illustration of the loss. However, only one pair of images at a time is considered during training.

3.5.2.2 Triplet networks

Triplet networks are an extension of the siamese networks where the loss function minimizes relative similarities (illustrated in Figure 3.11). Each training triplet is composed of an *anchor* or image of reference, a positive example of the same class as the anchor, and a negative example of a different class to the anchor. The loss function can be defined as:

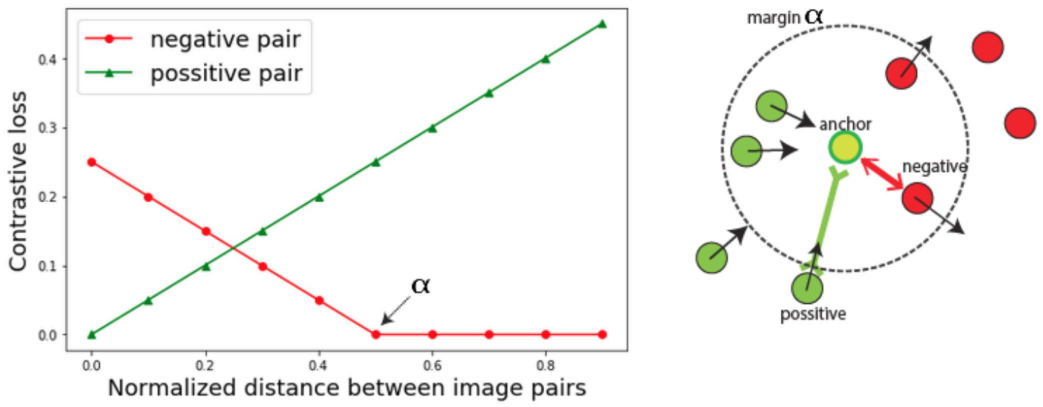


Figure 3.10: Contrastive loss (Equation 3.3). The loss is an equally weighted combination of the effect of the negative and positive pairs. The negative pairs contribute to the final loss with the hinge loss: if the distance between negative pairs is less than certain margin α (in the plot $\alpha = 0.5$) then, the pair contributes with the value of their distance, otherwise the effect to the loss is 0 since the negative pair already satisfies the margin condition. The positive pairs directly contribute to the loss with the value of their distance.

$$\mathcal{L}(\mathbf{x_a}, \mathbf{x_p}, \mathbf{x_n}) = \frac{1}{2} \max(0, D(\mathbf{x_a}, \mathbf{x_p}) - D(\mathbf{x_a}, \mathbf{x_n}) + \alpha), \tag{3.4}$$

where $D(\mathbf{x_a}, \mathbf{x_p})$ is the Euclidean distance between the anchor and a positive example, $D(\mathbf{x_a}, \mathbf{x_n})$ is the Euclidean distance between the anchor and a negative example, and α is a margin hyperparameter.

Figure 3.10 illustrates the loss behaviour, which ensures that given an anchor image, the distance between the anchor and a negative image is larger than the distance between the anchor and a positive image by a certain margin α.

The main difference between siamese and triplet architectures is that the first optimized separately positive and negative pairs, whereas the second optimized relative distances with a reference image or anchor. This usually results in better performance, as shown in [44, 108].

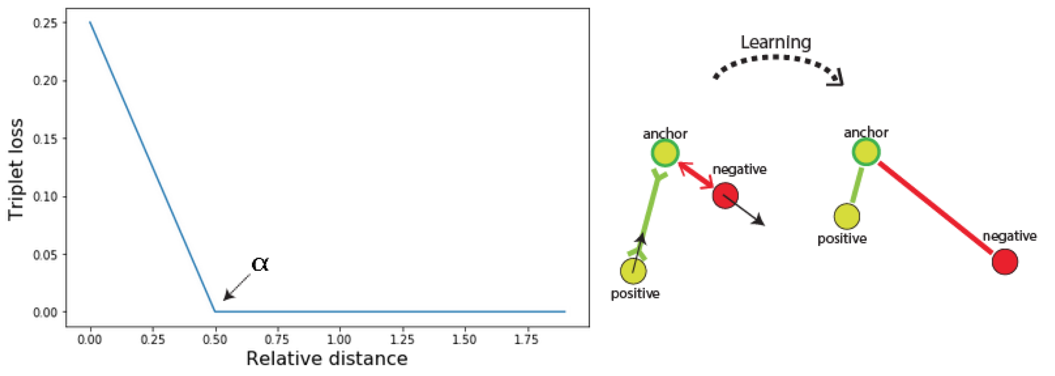


Figure 3.11: Visualization of the triplet loss. Relative distance refers to $D(\mathbf{x_a}, \mathbf{x_n}) - D(\mathbf{x_a}, \mathbf{x_n})$ On the left are the values of the loss (Equation 5.1) based on the relative distance between positive and negative pairs. On the right, the visualization before and after the the optimization of the triplet loss: In the example, the distance between the anchor and a positive image $D(\mathbf{x_a}, \mathbf{x_p})$, and between anchor and negative $D(\mathbf{x_a}, \mathbf{x_n})$, has been reduced and enlarged simultaneously such that $D(\mathbf{x_a}, \mathbf{x_p}) + \alpha < D(\mathbf{x_a}, \mathbf{x_n})$.

3.5.3 Training data for similarity learning

Generating training data for similarity learning is not a trivial task. The procedure for collecting a new image dataset is usually divided into two steps:

- **Web crawling**: Given a list of pre-defined text queries depicting different categories, query for them using a popular image search engine (Google Image Search, Bing, Flick) to download a noisy set of labeled images.
- **Data cleaning**: Images retrieved by available search engines usually contain noisy results such as near-duplicates or unrelated images, high intra-class image variations such as interior or exterior images of a particular building (Figure 3.12), or high diversity in the image resolution. Two approaches are followed after web crawling:
  - Manual data cleaning: exhaustively inspecting all images for dataset of moderate or small sizes [8, 96, 97, 51] or by making use of crowd sourcing services such as Amazon Mechanical Turk for large scale datasets [29, 147, 11].

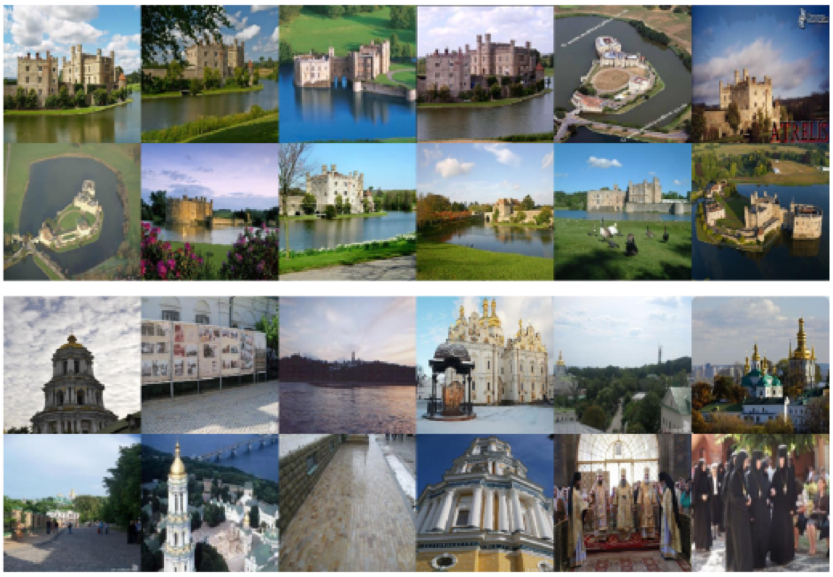


Figure 3.12: Example of the intra-class diversity between two classes of Landmarks dataset “Leeds Castle” and “Kiev Pechersk Lavra” . The two first rows correspond to one class, where the same building is visible in all the instances. The third and fourth rows correspond to the second class, where interior and exterior images are mixed and spatial geometry is not well preserved. Image source [8].

- Automatic data cleaning: in this approach, metadata associated with the images is exploited as an additional filtering step such as geo-tagged datasets [130, 1] and/or the usage of image representations to estimate similarity and geometric consistency between images, a process that usually relies on hand-crafted local invariant features [39, 100, 132].

Depending on the source data used for optimizing the CNN model for similarity learning, we can divide the different retrieval works into three main categories: works using existing annotated datasets [131], those exploring GPS-annotated datasets [1], and those that construct their own image training set [132, 11, 39, 100] .

3.5.3.1 Existing annotated datasets

The most direct solution for generating training data to optimize a model for image similarity is using the annotations provided in retrieval benchmarks. This approach was followed by Wan *et al.* [131], where they used the Oxford, Paris, and the Holiday dataset ground-truth labels for fine-tuning. *fc7* from the AlexNet model pre-trained on ImageNet was used as the image representation. The model was fine-tuned by optimizing the triplet hinge loss of Equation 5.1. Training triplets were sampled based on the relevant query annotations: a positive pair (anchor and relevant images) was randomly sampled from the same landmark query, and the associated negative image was randomly sampled from all the non-relevant images to the anchor landmark. The obtained models significantly outperform representations derived from off-the-shelf classification models. However, the query images are unknown in real-world retrieval scenarios and it is unrealistic to assume that annotations relative to a query will be available. Also, it is unclear how the fine-tuned model would generalize when retrieving different instances in different domains.

While the method significantly improved performance of off-the-shelf CNN representations, using ground truth annotations in a real-world image retrieval problem is not a realistic solution: first, it is unlikely to have ground-truth annotations; and second, since the amount of images (as well as diversity on the kind of instances of the considered dataset) are limited, it is unclear how the fine-tuned model would generalize in other datasets and/or image domains.

3.5.3.2 GPS-annotated dataset

Tasks such as image geo-localization [130], landmark classification [72], and place recognition [1] can benefit from GPS-annotated datasets. In particular, Arandjelović *et al* [1] explored using the Google Street View Time Machine to query multiple street-level panoramic images taken at different times at close-by spatial locations.

In their approach, a triplet CNN network based on the pre-trained VGG16 network was fine-tuned. Representations from the last convolutional layer were aggregated via the VLAD encoding, similar to the approach followed in [138]. One of their main contributions was reformulating VLAD using soft assignments, so the whole encoding operation was differentiable. This way, feature learning and descriptor encoding could be optimized at the same time using a similarity objective function.

Triplets were sampled so that given an anchor image a, a set of potential positive $\{p_i\}$ (those images within 10 metres difference with respect the anchor) and definite negatives $\{n_j\}$ (random images separated more than 20 metres) are considered. The triplet loss is modified to a *weakly supervised ranking loss*:

$$\mathcal{L}(\mathbf{x_a}, \mathbf{x_p}, \mathbf{x_n}) = \sum_{\mathbf{x}^- \in N_a} \left(\max(0, \min_{\mathbf{x}^+ \in P_a} (D(\mathbf{x_a}, \mathbf{x}^+) - D(\mathbf{x_a}, \mathbf{x}^-) + \alpha) \right). \tag{3.5}$$

For a particular anchor a, the loss takes into account its most similar image (the one with smaller Euclidean distance D) and a set of negative candidates according to the associated GPS information. While the triplet loss (Equation 5.1) only compares the anchor with one positive and negative example at the time, the proposed loss adds multiple negative examples at every step. This is known to lead to faster convergence and to learning more discriminative representations of the data than the one obtained with the original triplet loss [118].

3.5.3.3 Manually cleaned datasets

Bell *et al.* [11] recursively browsed pages on Houzz.com downloading each photo and its metadata. Duplicates and near duplicates were detected using *fc7* descriptors from pre-trained AlexNet. A crowd sourcing mechanism was used for further filtering and annotation of the spatial extent of the object of interest. AlexNet and GoogleNet architectures were fine-tuned on the new data, removing the last softmax layer and using the last fully connected layer as image representation as in [9, 101].

A similar procedure was followed by Wan *et al.* [132]. The authors fine-tuned AlexNet for the task of fine-grained image recognition optimizing a triplet loss. In this case, Google Image Search was used to query 100,000 predefined text queries. The top 140 retrieved images were kept per query. After that, a semi-automatic cleaning approach was employed to generate the relevance dataset, where a *golden* *feature* was used to compute the relevance score between image pairs $r_{i,j}$ within the same category. Images from different queries were set $r_{i,j} = 0$. The golden feature consisted of a weighted linear combination of several hand-crafted image features including features learned through human annotated data. The relevance score was used to sample the training triplets, where positive pairs shared the same class and high relevance score, and negatives were uniformly sampled from different random categories and those sharing low relevance scores from the same class.

Although these methods use image representations to clean and improve the quality of matching image pairs, both approaches [11, 132] rely heavily on human-annotated data. Generalization performance across different datasets is unclear since retrieval performance is evaluated only on the generated datasets.

3.5.3.4 Automatically cleaned datasets

Recent approaches [39, 100] have used automated methods to generate the training image set and to evaluate the performance of the fine-tuned models on different retrieval benchmarks. Both methods rely on hand-crafted invariant local features.

Gordo *et al.* [39] used the Landmark dataset [9] to produce a set of clean training data. For this, they run a strong image matching baseline within images of each landmark classes. Hessian-Affine keypoints detectors and SIFT were used to extract local discriminative features. For each image pair within a class, keypoints were matched using first-to-second neighbor ratio rule, and further verified with an affine transformation model [96]. An image graph was built for each class where each node represented the images and edges the pairwise matches, as illustrated in Figure 3.13. Low scored edges were removed and only the largest connected components were kept. Predicted affine transformations were further used to estimate bounding boxes.

Similarly, Radenović *et al.* [100] used a large unlabeled image collection downloaded from Flickr (using keywords of famous landmarks, cities, countries, and architectural sites). Images are initially described using a SIFT-BoW representation [96], and then clustered via min-hash and spatial verification [25]. A 3D model

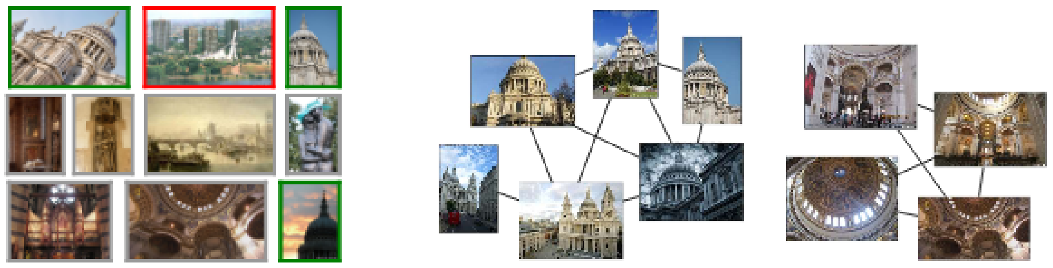


Figure 3.13: Random images “St Paul's Cathedral’ from the uncleaned Landmark dataset [9]. Green, gray and red border resp. denote prototypical, non-prototypical, and incorrect images. Right: two largest connected component of the pairwise matching graph. Image source [39].

based on Structure-from-Motion [99] is used to build a similarity graph for each cluster.

Both methods estimate similarity graphs for different landmarks (related to initial labeled data [39] or automatically discovered data [100]). Triplets can be sampled by selecting positive matching pairs from the same image cluster and negative image pairs from different clusters.

3.5.4 Hard negative mining

During learning, networks are typically optimized via mini-batch stochastic gradient descent (SGD). Sampling pairs or triplets at random is an inefficient strategy because many of them may already satisfy the margin criteria of Equations 3.3 and 5.1. That means that no error is generated and no gradients are backpropagated, so the weights of the CNN model are not updated and no learning occurs.

To sample positive pairs, a common procedure consists of sampling images that belong to the same class [39], 3D point or cluster [111, 100]. Some approaches select positive pairs with minimal distance within the initial embedding space [54]. In order to avoid sampling very similar images, Radenović *et al.* [100] make use of the strong matching pipeline with 3D reconstruction to select only positives that share the minimum amount of local matches, so matching images depict the same object but also ensure variability of viewpoints.

For the negative pairs a common procedure consists of iterating over non-matching images that are "hard" negatives, those being close in the descriptor space and that incur a high loss. For that, the loss is computed over a set of negative pairs and only a subset with higher loss is selected for training. This procedure is repeated every N iterations of SGD so "hard" examples are picked throughout the learning process [111, 39]. Variability in the sampling is ensured in [100] by selecting the negative pairs from different clusters or 3D points. Selecting negative pairs based on the loss generally results in multiple and very similar instances of the same object.

3.5.5 Architectures for retrieval

Recent end-to-end networks proposed for retrieval are based on state-of-the-art architectures for image classification. Final retrieval representations are build from convolutional layers. Proposed architectures start from a pre-trained CNN model (i.e AlexNet [64], VGG16 [113], ResNet101 [43]), which differ mainly in the top layers designed to aggregate the local convolutional features as illustrated in Figure 3.14.

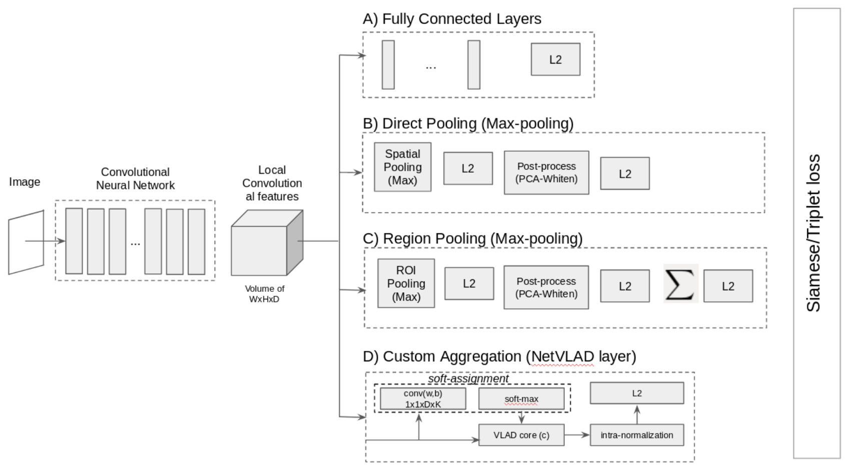


Image
Convolutional Neural Network
...
Local Convolution al features
Volume of WxHxD
A) Fully Connected Layers
...
L2
B) Direct Pooling (Max-pooling)
Spatial Pooling (Max)
L2
Post-process (PCA-Whiten)
L2
C) Region Pooling (Max-pooling)
ROI Pooling (Max)
L2
Post-process (PCA-Whiten)
L2
∑
L2
D) Custom Aggregation (NetVLAD layer)
soft-assignment
conv(w,b) 1x1xDxK
soft-max
L2
VLAD core (c)
intra-normalization
Siamese/Triplet loss

Figure 3.14: End-to-end CNN architectures. The final representations are always L^2-normalized and siamese or triplet loss is used as loss functions. The approaches mainly differ in how they build the final image representation from the convolution layer: They can be aggregated with fully connected layers (A); they can be aggregated via direct spatial pooling and adding a post processing layer (PCA linear transformation) (B); local descriptors can be locally pooled to generate a set of region vectors. For this, an additional layer for region pooling is added followed by a region-post-processing and sum-pooling (C); the method can use a custom aggregation strategy such as the one proposed in NetVLAD (D).

Some of the approaches directly fine-tune the original classification network [131, 11, 132]. For instance, the full AlexNet architecture is used in [11] (identified as line A) in Figure 3.14), where the authors explore multi-task learning by optimizing the model for similarity learning along with a classification loss for product identification and search. To switch between classification and similarity comparison, the softmax operation at the end of the network is replaced with an inner product layer with a L^2-normalized output. The full architecture of AlexNet is also used in [132]. In this case two additional channels containing a shallow CNN are used to process two low-resolution versions of the original image. The full AlexNet is also fine-tuned in [131], where performance is evaluated on Oxford and Paris showing a substantial improvement over pre-trained fully connected layers (see Table 3.3).

Recent works base their architectures on VGG16 exploiting the capabilities of convolutional layers [100, 39, 1]. Gordo *et al.* [39] proposed a fine-tuned version of R-MAC (introduced in Section 3.4.4), where a region proposal network [102] is learned

Table 3.3: End-to-end retrieval approaches reporting results in Oxford and Paris datasets. In parenthesis, the performance of the method when applying re-ranking and/or query expansion. The two best-performing systems are highlighted.

		Pre-trained ImageNet		Fine-tuned	
	Dim	Oxford5k	Paris6k	Oxford5k	Paris6k
Neural Codes [9]	128	0.433		0.557	
Wan [131]	4096	0.417	0.580	0.783	**0.947**
Faster-RCNN [106][*]	512	0.588(0.647)	0.657(0.732)	0.798(0.786)	0.798(0.842)
NetVLAD [1]	512	0.590	0.702	0.676	0.749
MAC [100]	512	0.583	0.726	0.800(0.854)	0.829(0.870)
R-MAC [39]	512	**0.654**	**0.813**	**0.834(0.894)**	0.871(0.912)
R-MAC (ResNet101) [39]	2048	**0.729**	**0.866**	**0.852(0.944)**	**0.940(0.966)**

[*] Model was pre-trained for object detection in Microsoft COCO [74].

as a replacement of the fixed grid originally proposed in [125]. Coordinates predicted by the RPN are mapped on the convolutional layer and activations are max-pooled to generate a fixed length vector that is L^2-normalized. Region descriptors are post-processed with PCA whitening, which is implemented with a shifting and a fully connected layer. Region vectors are then L^2-normalized again and pooled into the final global descriptor via sum-pooling (line C) in Figure 3.14). This approach achieves the current state-of-the-art in Oxford and Paris using CNN representations. When using a deeper network architecture (ResNet101 [43]) combined with database side augmention, [3] achieves 0.947 and 0.966 mAP in Oxford and Paris dataset, which is the best performance reported for those benchmarks to date.

Randenović [100] follow the same architecture as in [39] but directly pooling all descriptors generated by the convolutional layer (MAC), without learning a RPN (line B) in Figure 3.14). A discriminative dimensionality reduction transformation is learned via linear discriminant projections proposed by Mikolajczyk and Matas [84] using the annotated training data. Arangjelović [1] propose a more sophisticated encoding by implementing a VLAD layer on top of the convolutional features using soft-assignments to be able to tune the parameters via backpropagation (line D) in Figure 3.14). Similarly, a Fisher-Vector layer has been proposed in [92].

Table 3.4: Performance of discussed CNN approaches in different image retrieval benchmarks. Performance of approaches with post-processing steps such as re-ranking and/or query expansion are given in parenthesis.

	Method	Dim	Oxford5k	Oxford105k	Paris6k	Paris106k	Holidays
Off-the-shelf Fully connected	Neural Codes [9]	128	0.433	0.386			
	CNNastounding [110]	4-15k	0.68		0.79		
	MOP [38]	2048					0.802
Off-the-shelf Convolutional	SPoC [8]	256	0.589	0.578			0.802
	Ng *et al* [138]	128	0.593		0.590		0.816
	Razavian [101]	32k	**0.843**		**0.879**		**0.896**
	R-MAC [125]	512	0.669(0.773)	0.616(0.732)	0.830(0.865)	0.757(0.798)	
	CAM [60]	512	0.712(0.801)	0.672(0.769)	0.805(0.855)	0.733(0.800)	
	CroW [**?**]	512	0.708(0.749)	0.653(0.706)	0.797(0.848)	0.722(0.794)	0.851
End-to-End Training	Neural Codes [9]	128	0.557	0.523			
	Wan [131]	4096	0.783		0.947		
	Faster-RCNN [106]	512	0.710(0.786)		0.798(0.842)		
	NetVLAD [1]	256	0.635		0.735		0.843
	MAC [100]	512	0.800(0.854)	0.751(0.823)	0.829(0.870)	0.753(0.796)	0.795
	R-MAC [39]	512	0.831(0.894)	0.786(0.873)	0.871(0.912)	0.797(0.868)	0.891
	R-MAC (ResNet101) [39]	2048	**0.845(0.890)**	**0.816(0.878)**	**0.912(0.938)**	**0.863(0.906)**	**0.960**
Hand-crafted methods	BoW(1M)+QE [26]		0.827	0.767	0.805	0.710	
	Arandjelović [3]		**0.929**	**0.891**	**0.910**		

3.6 Summary

Early works (Section 3.3) focused on the use of fully connected layers as a global image representation. Descriptors were ranked according to their L^2-normalized Euclidean distance. It was shown that PCA dimensionality reduction and whitening are common post-processing steps that help to improve the performance of off-the-shelf CNN representations. However, CNN descriptors only outperform traditional approaches at low dimensions. Some works propose to apply spatial search to generate a set of multiple region descriptors per image, along with data augmentation, to obtain a significant boost in performance at the price of memory and computational requirements. To overcome that limitation, it was shown that traditional aggregation methods (such as VLAD) could potentially be used on CNN region descriptors to generate a richer global image representation.

A second generation of works explored convolutional layers (Section 3.4). Descriptors derived from convolutional layers have the advantage of containing more general information, preserving the spatial layout of the image and, also, they allow processing of images at full resolution, while keeping the original aspect ratios.

Different aggregation approaches have been applied on local descriptors derived from convolutional layers, direct max-pooling being one of the most popular due to its simplicity. Constructing region descriptors from these layers is also more efficient than using fully connected layers. Moreover, different spatial weighting schemes can be applied to weight the contribution of different areas within an image.

It was shown that fine-tuning the network with images more semantically related to the final task does help to create better image representations. However, using similarity learning objectives to directly optimize the descriptor is a more suitable strategy for retrieval than using a classification loss. In Section 3.5 we discussed a series of works where authors perform end-to-end learning of a CNN for the task of image retrieval. For that, siamese and triplet architectures are commonly used. One of the challenges that still remains is generating suitable training data to optimize such models, including the dataset annotation (manual or automatic) and negative mining of the pairs or triplets.

Table 3.4 shows the performance of the discussed works in well-known retrieval benchmarks reported in the original papers. End-to-end approaches based on direct pooling (MAC) and pooling region vectors (R-MAC) are currently the state-of-the art CNN-based representations. Performance of the method has improved further by query expansion, achieving similar results to SIFT-BoW systems with spatial verification and query expansion stages [3, 26].

3.7 Conclusions

End-to-end CNN models achieve a new state of the art performance in some popular retrieval benchmarks. However, the generalization of those models in other retrieval domains remains unclear and needs to be furthered explored. Additionally, the best performing approaches involve the generation of a large manually and/or automatically processed dataset of images for network training; which have been selected allows the model to recognize instances of a particular domain.

With the purpose of developing a general instance search system, in Chapter 4 we propose to use bag of words encoding (BoW) as a direct aggregation of pre-trained convolutional features. While BoW has been one of the most popular approaches in the traditional pipelines based in hand-crafted features such SIFT, it has yet not being explored as an aggregation method for local convolutional representations. The BoW representation has the advantage of generating high-dimensional and sparse representations, that can benefit from inverted files structures to efficiently store and retrieve information.

Additionally, mapping each CNN local feature into one visual word of a high dimensional vocabulary allows the inclusion of weighting schemes in our pipeline (weighting the contribution of each word prior to aggregation) and region encoding (by aggregating only the words of a particular image region).

As highlighted in this chapter, one of the main challenges to train a CNN for similarity learning is the generation of a suitable training set. For this reason in Chapter 5 we investigate an unsupervised approach to fine-tune a CNN based in combining a SIFT-based and a CNN-based BoW systems. SIFT and CNN local features contain complementary information allowing the generation of highly accurate ranked lists. Training images can be sampled from the obtained ranks to train a simpler and more efficient retrieval system.

Lastly, some of the discussed works have applied weighting schemes on convolutional features to weight the contribution of different local regions of images. Some other works, alternatively, perform region analyis by extracting a set of region vectors per image, typically in a fixed pre-defined grid of locations. In Chapter 6 we provide a re-interpretation of the region analysis as a kind of weighting (when regions are sumpooled into a compact representation), and we propose to use attention models as a more sophisticated weighting scheme.

We observe in Chapter 4 that BoW can be successfully applied to off-the-shelf, and to fine-tuned CNN models in Chapter 5, obtaining an efficient retrieval system that can greatly improve performance by applying an attention mechanism to weight the contribution of the most relevant areas within the images, as we do in Chapter 6.

Chapter 4

Bags of Local Convolutional Features

4.1 Introduction

Representations based on convolutional neural networks (CNNs) have been demonstrated to outperform the state-of-the-art in many computer vision tasks. CNNs trained on large amounts of labeled data produce global representations that effectively capture the semantics in images. Features drawn from such networks have been successfully used in various image retrieval benchmarks [9, 110, 8? , 101, 125] improving upon the state-of-the-art compact image representations for image retrieval.

Despite CNN-based descriptors performing remarkably well in retrieval benchmarks like Oxford and Paris Buldings, state-of-the-art solutions for more challenging datasets such as TRECVID Instance Search (INS) have not yet adopted pipelines that depend solely on CNN features. Current INS systems [88, 141, 148, 149] are still based on aggregating local hand-crafted features (like SIFT) using bag of words encoding [115] to produce very high-dimensional sparse image representations. Such high-dimensional sparse representations have several benefits over their dense counterparts. High-dimensionality means they are more likely to be linearly separable, which means that if we consider each instance as a class, it is possible to find an hyperplane that separates the different instances in the feature space. Also, the sparsity allows having relatively few non-zero elements, making the representations efficient both in terms of storage (only non-zero elements need to be stored), and computation (only non-zero elements need to be visited). Sparse representations can handle varying information content, and are less likely to interfere with one another when pooled. From an information retrieval perspective, sparse representations can be stored in inverted indices, which facilitates efficient selection of images that share features with a query. Furthermore, there is considerable evidence that biological systems make extensive use of sparse representations for sensory information [71, 129].

Many successful image retrieval engines combine an initial highly-scalable ranking mechanism on the full image database with a more computational demanding yet higher-precision re-ranking mechanism applied to the top retrieved items. This re-ranking mechanism often takes the form of geometric verification and spatial analysis [52, 142, 83, 141], after which the best matching results can be used for query expansion (pseudo-relevance feedback) [3, 26].

In this chapter, inspired by advances in CNN-based descriptors for image retrieval discussed in Chapter 3, yet still focusing on instance search, we revisit the Bag of Words encoding scheme using local features from convolutional layers of a CNN. This work presents three contributions:

- We propose a sparse visual descriptor based on a Bag of Local Convolutional Features (BLCF), which allows fast image retrieval by means of an inverted index.
- We introduce the assignment map as a new compact representation of the image, which maps pixels in the image to their corresponding visual words. The assignment map allows fast composition of a BoW descriptor for any region of the image.
- We take advantage of the scalability properties of the assignment map to

perform a local analysis of multiple regions of the image for reranking, followed by a query expansion stage using the obtained object localizations.

Using this approach, we present an image retrieval system that achieves state-of-the-art performance in CBIR benchmarks and outperforms current state-of-the-art CNN based descriptors for the task of instance search.

The remainder of the chapter is organized as follows: in Section 4.2 we first provide a description of the bag of words frameworks, then in Section 4.3 we describe the full retrieval system composed of an initial search and a re-ranking strategy. In Section 4.4 we describe the experimental setup: datasets considered as well as preliminary experiments. Sections 4.4.3 and 4.4.4 describe the results obtained by adding query expansion and a re-ranking strategy to the pipeline. Lastly, we compare and discuss the obtained results with other state-of-the-art approaches in Section 4.5. We evaluate the advantage of using high-dimensional representation over direct sum-pooling on the challenging TRECVID instance search benchmark in Section 4.6. Sections 4.7 and 4.8 contain the discussion and conclusions for the overall chapter.

4.2 Bag of words framework

The proposed pipeline for feature extraction uses the activations at different locations of a convolutional layer in a pre-trained CNN as local features. A CNN trained for a classification task is typically composed of a series of convolutional layers, followed by some fully connected layers, connected to a softmax layer that produces the inferred class probabilities. To obtain a fixed-sized output, the input image to a CNN is usually resized to be square. However, several authors using CNNs for retrieval [125?] have reported performance gains by retaining the aspect ratio of the original images. We therefore discard the softmax and fully connected layers of the architecture and extract CNN features maintaining the original image aspect ratio.

Each convolutional layer in the network has D different $N \times M$ feature maps, which can be viewed as $N \times M$ descriptors of dimension D. Each of these descriptors contains the activations of all neurons in the convolutional layer sharing the same receptive field. This way, these D-dimensional features can be seen as local descriptors computed over the region corresponding to the receptive field of an array of neurons. With this interpretation, we can treat the CNN as a local feature extractor and use any existing aggregation technique to build a single image representation.

We propose to use the Bag of Words model to encode the local convolutional features of an image into a single vector. Although more elaborate aggregation strategies to outperform BoW-based approaches for some tasks in the literature [54, 95], Bag of Words encodings produce sparse high-dimensional codes that can be stored in inverted indices, which are beneficial for fast retrieval. Moreover, BoW-based representations are faster to compute, easier to interpret, more compact, and provide all the benefits of sparse high-dimensional representations previously mentioned.

Bag of words models require constructing a visual codebook to map vectors to their nearest centroid. We use k-means on local CNN features to build this codebook. Each local CNN feature in the convolutional layer is then assigned its closest visual word in the learned codebook. This procedure generates the *assignment map*, i.e. a 2D array of size $N \times M$ that relates each local CNN feature with a visual word. The assignment map is therefore a compact representation of the image which relates each pixel of the original image with its visual word with a precision of $\left(\frac{W}{N}, \frac{H}{M}\right)$ pixels, where W and H are the width and height of the original image. This property allows us to quickly generate the BoW vectors of not only the full image, but also its parts.

Figure 4.1 shows the pipeline of the proposed approach. The *Bag of Local Convolutional Features (BLCF)* encodes the image into a sparse high dimensional descriptor which will be used as the image representation for retrieval.

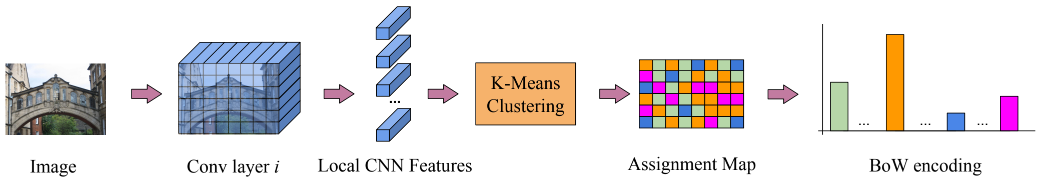


Figure 4.1: The Bag of Local Convolutional Features pipeline (BLCF).

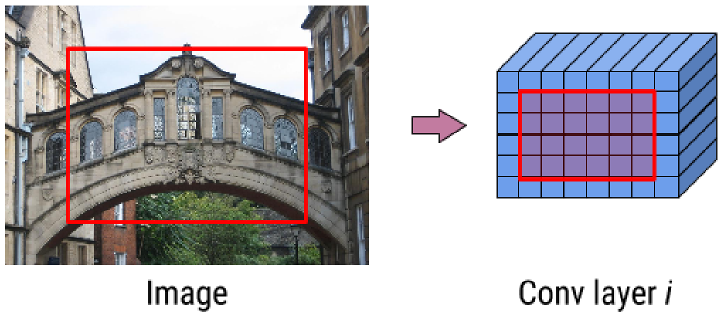


Figure 4.2: Selection of Local CNN features to construct the BoW query descriptor for the Local Search (LS).

4.3 Image retrieval

This section describes the image retrieval pipeline, which consists of an initial ranking stage, followed by a spatial re-ranking, and query expansion.

(a) Initial search: The initial ranking is computed using the cosine similarity between the BoW vector of the query image and the BoW vectors of the full images in the database. We use a sparse matrix based inverted index and GPU-based sparse matrix multiplications to allow fast retrieval. The image list is then sorted based on the cosine similarity of its elements to the query. We use two types of image search based on the query information that is used:

- Global search (GS): The BoW vector of the query is built with the visual words of all the local CNN features in the convolutional layer extracted for the query image.
- Local search (LS): The BoW vector of the query contains only the visual words of the local CNN features that fall inside the query bounding box.

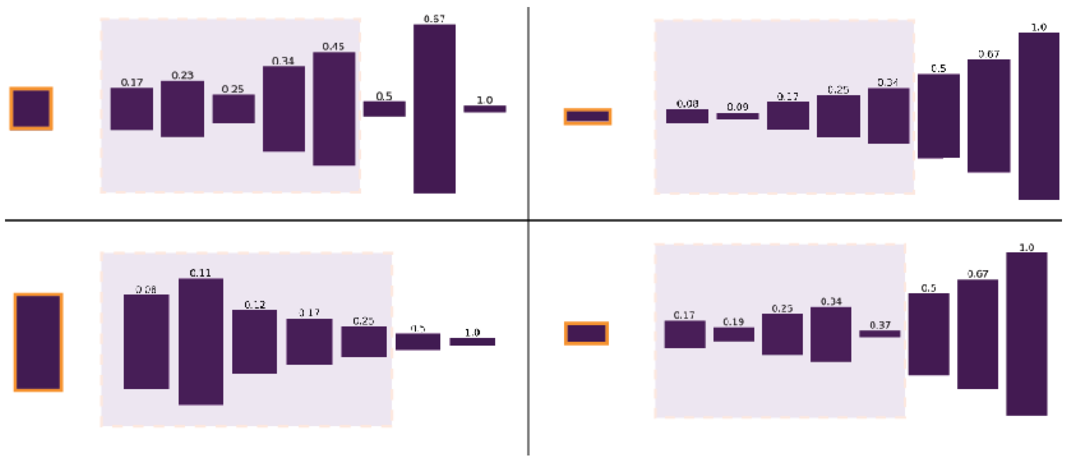


Figure 4.3: Windows sorted according with $score_w$ for four different query aspect ratio. Highlighted, the window aspect ratios selected given a threshold th.

(b) Local re-ranking (R): After the initial search, the top T images in the ranking are locally analyzed and re-ranked based on a localization score. We choose windows of all possible combinations of width $w \in \{W, \frac{W}{2}, \frac{W}{4}\}$ and height $h \in \{H, \frac{H}{2}, \frac{H}{4}\}$, where W and H are the width and height of the assignment map. We use a sliding window strategy directly on the assignment map with 50% overlap in both directions.

Since analyzing all the window is computational demanding, we perform a simple filtering strategy to discard those windows whose aspect ratio is too different to the aspect ratio of the query. Let the aspect ratio of the query bounding box be $\mathrm{Ar}_q = \frac{W_q}{H_q}$ and $\mathrm{Ar}_w = \frac{W_w}{H_w}$ be the aspect ratio of the window. The score for window w is defined as $score_w = \frac{\min(\mathrm{Ar}_w, \mathrm{Ar}_q)}{\max(\mathrm{Ar}_w, \mathrm{Ar}_q)}$. Figure 4.3 shows a set of candidate windows for four different query aspect ratios. The target windows are sorted by the normalized proposed score. For a non-square target image, a set of 8 different window resolutions are generated, we discard all the resolutions with a score higher than a threshold th.

For each of the remaining windows, we construct the BoW vector representation and compare it with the query representation using cosine similarity. The window with the highest cosine similarity is taken as the new score for the image (score max pooling).

We also enhance the BoW window representation with spatial pyramid match-

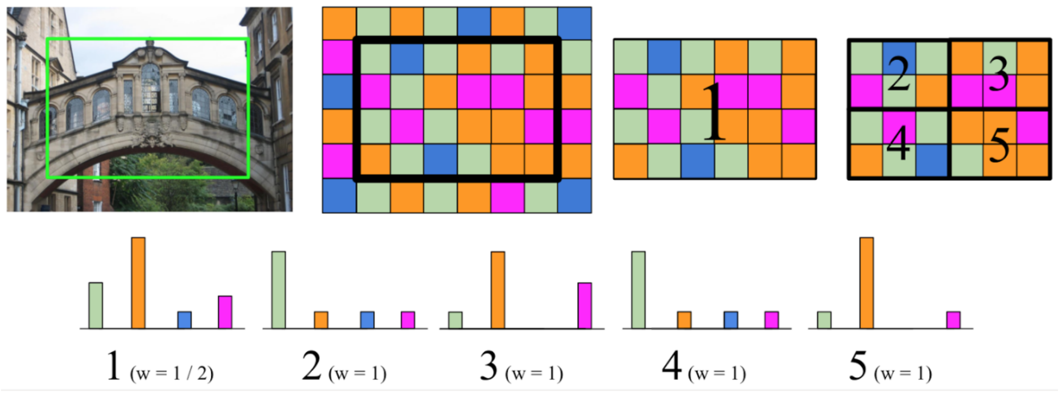


Figure 4.4: Spatial Pyramid matching on window locations.

ing [70]. A BoW representation is constructed for the full resolution and each of the subregions, as shown in Figure 4.4. Then, their contribution to the similarity score is weighted with inverse proportion to the resolution level l_r of the region ($l_r = 1$ for the full image, and $l_r = 2$ for the subregions). The cosine similarity of a sub region r to the corresponding query sub region is therefore weighted by $w_r = \frac{1}{2^{(L-l_r)}}$, where $L = 2$ indicates the total number of resolutions.

With this procedure, the top T elements of the ranking are sorted based on the cosine similarity, which is a weighted combination of the cosine similarity of the different subregions. This procedure also provides the region with the highest score as a rough localization of the object.

(c) Query expansion We investigate two query expansion strategies [27] based on global and local BoW descriptors:

- Global query expansion (GQE): The BoW vectors of the N images at the top of the ranking are averaged together with the BoW of the query to form the new representation for the query. GQE can be applied either before or after the local reranking stage.

- Local query expansion (LQE): Locations obtained in the local reranking step are used to mask out the background and build the BoW descriptor of only the region of interest of the N images at the top of the ranking. These BoW vectors are averaged together with the BoW of the query bounding box. The

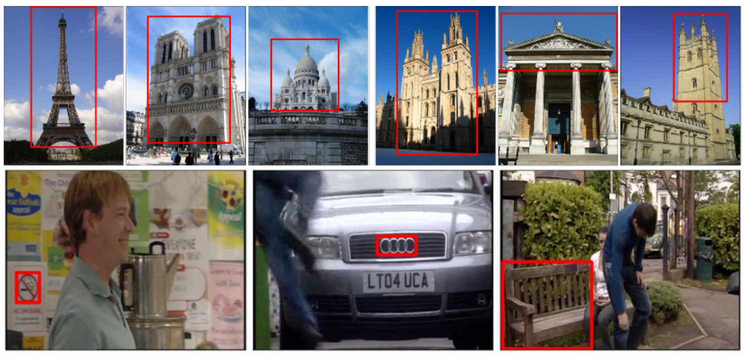


Figure 4.5: Query examples from the three different datasets. Top: Paris buildings (1-3) and Oxford buildings (4-6); bottom: TRECVID Subset.

resulting BoW vector is used to perform a second search.

4.4 Experiments

4.4.1 Datasets

We used two domain specific datesets Oxford [96] and Paris [97] buildings, and the more generic TRECVID Subset [116] dataset, to evaluated the proposed BLCF. For detailed detail of the datasets see Section 2.6. Figure 4.5 includes three examples of query objects from the three datasets.

4.4.2 Preliminary experiments

We extracted features from the last three convolutional layers (conv5_1, conv5_2 and conv5_3) and compared their performance on the Oxford 5k dataset. We experimented with different image input sizes: 1/3 and 2/3 of the original image. Following several other authors [8**?** , 125] as discussed in Chapter 3, we L^2-normalize all local features, followed by PCA, whitening, and a second round of L^2-normalization. The PCA models were fit on the same dataset as the test data in all cases.

Unless stated otherwise, all experiments used a visual codebook of 25,000 centroids fit using the (L^2-PCA-L^2 transformed) local CNN features of all images in the same dataset (1.7M and 2.15M for Oxford 5k and Paris 6k, respectively). We tested three

Table 4.1: Mean average precision (mAP) on Oxford 5k using different convolutional layers of VGG16, comparing the performance of different feature map resolutions (both raw and interpolated). The size of the codebook is 25,000 in all experiments. $N \times M$ denotes the spatial resolution of the input image.

	conv5_1	conv5_2	conv5_3
$N \times M$ raw	0.641	0.626	0.498
$2N \times 2M$ interpolated	$\bigcirc$ 0.653	0.638	0.536
$2N \times 2M$ raw	0.620	**0.660**	0.540

different codebook sizes (25,000, 50,000, and 100,000) on the Oxford 5k dataset, and chose the 25,000 centroids one because of its higher performance.

Inspired by the boost in performance of the Gaussian centre prior in SPoC features [8], we apply a weighting scheme on the visual words of an image to provide more importance to those belonging to the central part of the image. The weighting scheme $w(i, j)$ is described as:

$$w(i,j) = \frac{1}{\sqrt{(i-c_1)^2 + (j-c_2)^2}}, \tag{4.1}$$

where (i, j) represents the position of a visual word within the assignment map and (c_1, c_2) corresponds to the centre of the assignment map. The weights $w(i, j)$ are min-max normalized to provide scores between 0 and 1. Table 4.1 shows the mean average precision on Oxford 5k for the three different layers and image sizes. All results are obtained using this weighting criteria; for conv5_1 in Oxford 5k, this increases mAP from 0.626 to 0.653. The combination of the layers by concatenation did not provide any gain.

4.4.3 Query augmentation

Previous works [3, 126] have demonstrated how simple data augmentation strategies can improve the performance of an instance search system. Data augmentation consists in generating different versions of an image (by typically applying rotations, translations, and/or variations in the scale) with the purpose of obtaining a more robust image representation. Some of these apply augmentation strategies at the database side, which can be prohibitively costly for large datasets. For this reason, we use data augmentation on the query side only. We explore two different strategies to enrich the query before visual search: a horizontal flip (or mirroring) and a zoomed central crop (ZCC) on an image enlarged by 50%.

Figure 4.6 shows an example of the transformations, which give rise to 4 different versions of the query image. The feature vectors they produce are added together to form a single BoW descriptor by encoding their corresponding visual words into a single representation. Table 4.2 shows the impact of incrementally augmenting the query with each one of these transformations.

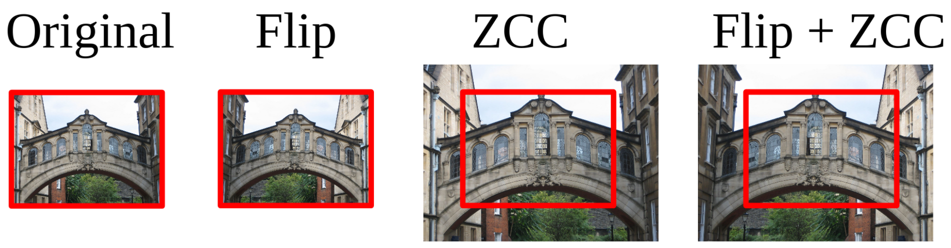


Figure 4.6: The four query images after augmentation.

Table 4.2: mAP on Oxford 5k for the two different types of query augmentation: the flip and the zoomed central crop (ZCC). 2× interpolated conv5_1 features are used in all cases.

	Query	+ Flip	+ ZCC	+ Flip + ZCC
Global Search (GS)	0.653	0.662	0.695	0.697
Weighted Search (WS)	0.706	0.717	0.735	0.743
Local Search (LS)	0.738	0.746	0.758	0.758

4.4.4 Re-ranking and query expansion

We apply the local re-ranking (R) stage on the top-100 images in the initial ranking, using the sliding window approach described in Section 4.3.

We scan different aspect ratio score thresholds th and evaluate the precision at the first 10 images (P@10) on the top-10 retrieved images. P@10 does not change for most of the thresholds, but the selected windows sometimes do and a lower threshold

Table 4.3: mAP on Oxford 5k and Paris 5k for the different stages in the pipeline introduced in Section 4.3. The Q_{aug} additional columns indicate the results when the query is augmented with the transformations introduced in Section 4.4.3.

	Oxford 5k		Paris 6k	
		$+Q_{aug}$		$+Q_{aug}$
GS	0.653	0.697	0.699	0.754
LS	0.738	0.758	0.820	0.832
GS + R	0.701	0.713	0.719	0.752
LS + R	0.734	0.760	0.815	0.828
GS + GQE	0.702	0.730	0.774	0.792
LS + GQE	0.773	0.780	0.814	0.832
GS + R + GQE	0.771	0.772	0.801	0.798
LS + R + GQE	0.769	**0.793**	0.807	0.828
GS + R + LQE	0.782	0.757	0.835	0.795
LS + R + LQE	**0.788**	0.786	**0.848**	**0.833**

value means faster computation (more candidate windows are discarded). We set $th = 0.4$ since it provides a small improvement from 0.80 to 0.84 in P@10 for the Oxford 5k dataset.

Query expansion is later applied considering the top-10 images of the resulting ranking. This section evaluates the impact in performance of both re-ranking and query expansion stages. Table 4.3 contains the results for the different stages in the pipeline for both simple and augmented queries (referred to as Q_{aug} in the table).

The results indicate that the local re-ranking is only beneficial when applied to a ranking obtained from a search using the global BoW descriptor of the query image (GS). This is consistent with the work by Tolias *et al.* [125], who also apply a spatial re-ranking followed by query expansion to a ranking obtained with a search using descriptors of full images. They achieve a mAP of 0.66 in Oxford 5k, which is increased to 0.77 after spatial re-ranking and query expansion, while we reach similar results (e.g. from 0.652 to 0.769). However, our results indicate that a ranking originating from a local search (LS) does not benefit from local re-ranking. Since the BoW representation allows us to effectively perform a local search (LS) in a database of full indexed images, we find the local re-ranking stage applied to LS to

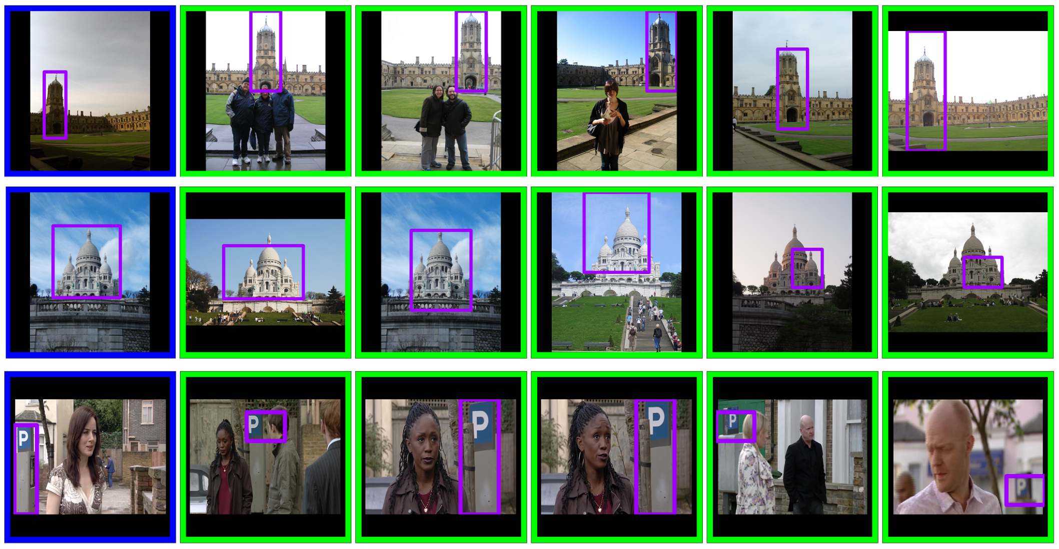


Figure 4.7: Examples of the top-ranked images and localizations based on local CNN features encoded with BoW. Top row: Christ Church from the Oxford Buildings dataset; middle row: Sacre Coeur from Paris Buildings; bottom row: query 9098 (a parking sign) from TRECVID INS 2013.

be redundant in terms of the achieved quality of the ranking. However, the local re-ranking stage does provide a rough localization of the object in the images of the ranking, as depicted in Figure 4.7. We use this information to perform query expansion based on local features (LQE).

Results indicate that query expansion stages greatly improve performance in Oxford 5k. We do not observe significant gains after re-ranking and QE in the Paris 6k dataset, although we achieve our best result with LS + R + LQE.

In the case of augmented queries ($+Q_{aug}$), we find query expansion to be less helpful in all cases, which suggests that the information gained with query augmentation and the one obtained by means of query expansion strategies are not complementary.

4.5 Comparison with the state-of-the-art

We compare our approach with other CNN-based representations using pre-trained networks that make use of features from convolutional layers on the Oxford and

Table 4.4: Comparison to state-of-the-art CNN representations (mAP). Results in the lower section consider re-ranking and/or query expansion.

	Oxford		Paris	
	5k	105k	6k	106k
Ng *et al.* [138]	0.649	-	0.694	-
Razavian *et al.* [101]	**0.844**	-	**0.853**	-
SPoC [8]	0.657	0.642	-	-
R-MAC [125]	0.668	0.616	0.830	**0.757**
CroW [**?**]	0.682	**0.632**	0.796	0.710
uCroW [**?**]	0.666	0.629	0.767	0.695
GS	0.652	0.510	0.698	0.421
LS	0.739	0.593	0.820	0.648
LS + Q_{aug}	0.758	0.622	0.832	0.673
CroW + GQE [**?**]	0.722	0.678	0.855	0.797
R-MAC + R + GQE [125]	0.770	**0.726**	**0.877**	**0.817**
LS + GQE	0.773	0.602	0.814	0.632
LS + R + LQE	0.788	0.651	0.848	0.641
LS + R + GQE + Q_{aug}	**0.793**	0.666	0.828	0.683

Paris datasets. Table 4.4 includes the best result for each approach in the literature. Our performance using global search (GS) is comparable to that of Ng *et al.* [138], the most similar to our approach. However, they achieve this result using raw VLAD features, which are more expensive to compute and, being a dense high-dimensional representation, do not scale as well to larger datasets. Similarly, Razavian *et al.* [101] achieve the highest performance of all approaches on both the Oxford and Paris benchmarks by applying a spatial search at different scales for all images in the database. Such an approach is prohibitively costly when dealing with larger datasets, especially for real-time search scenarios. Our BoW-based representation is highly sparse, allowing for fast retrieval in large datasets using inverted indices, and achieves consistently high mAP in all tested datasets.

We also compare our local re-ranking and query expansion results with similar approaches in the state-of-the-art. The authors of R-MAC [125] apply a spatial search for re-ranking, followed by a query expansion stage, while the authors of CroW [**?**] only apply query expansion after the initial search. Our proposed approach also achieves competitive results in this section, achieving the best result for Oxford 5k.

4.6 Experiments on TRECVID INS

In this section, we compare the Bag of Local Convolutional Features (BLCF) with the sum pooled convolutional features proposed in several works in the literature. We use our own implementation of the unweighted CroW representation[**?**], *uCroW*, and compare it with BLCF for the TRECVID INS subset. For the sake of comparison, we test our implementation of sum pooling using both our chosen CNN layer and input size (conv5_1 and 1/3 image size), and the ones reported in [**?**] (pool5 and full image resolution). For the BoW representation, we train a visual codebook of 25,000 centroids using 3M local CNN features chosen randomly from the INS subset. Since the nature of the TRECVID INS dataset differs significantly from that of the other ones used so far (see Figure 4.5), we do not apply centre priors to the features, to avoid down weighting local features from image areas where the objects might appear. Table 4.5 compares sum pooling with BoW on Oxford, Paris, and TRECVID subset datasets. As stated in earlier sections, sum pooling and BoW have similar performance in Oxford and Paris datasets. For the TRECVID INS subset, however, Bag of Words significantly outperforms sum pooling, which demonstrates its suitability for challenging instance search datasets, in which queries are not centered and have variable size and appearance. We also observe a different behaviour when using the provided query object locations (LS) to search, which was highly beneficial in Oxford and Paris datasets, but does not provide any gain in TRECVID INS. We hypothesize that the fact that the size of the instances is much smaller in TRECVID than in Paris and Oxford datasets causes this drop in performance. Global search (GS) achieves better results on TRECVID INS, which suggests that query instances are in many cases correctly retrieved due to their context.

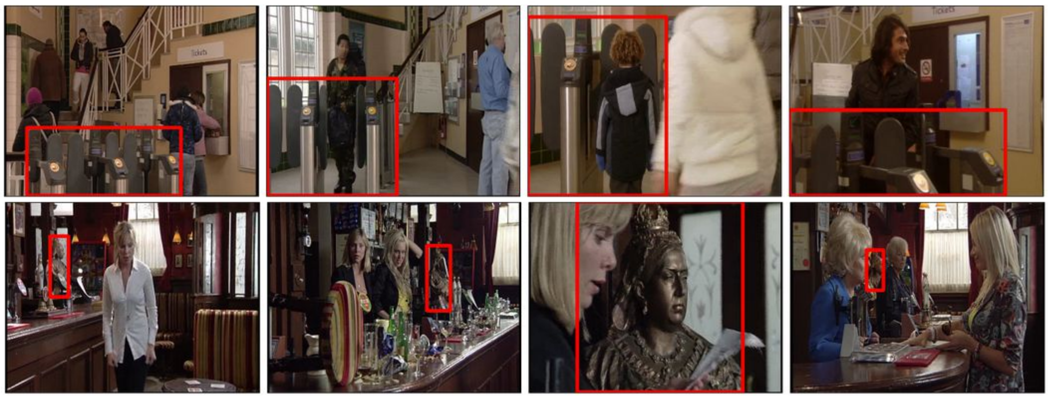


Figure 4.8: Appearances of the same object in different frames of TRECVID Instance Search.

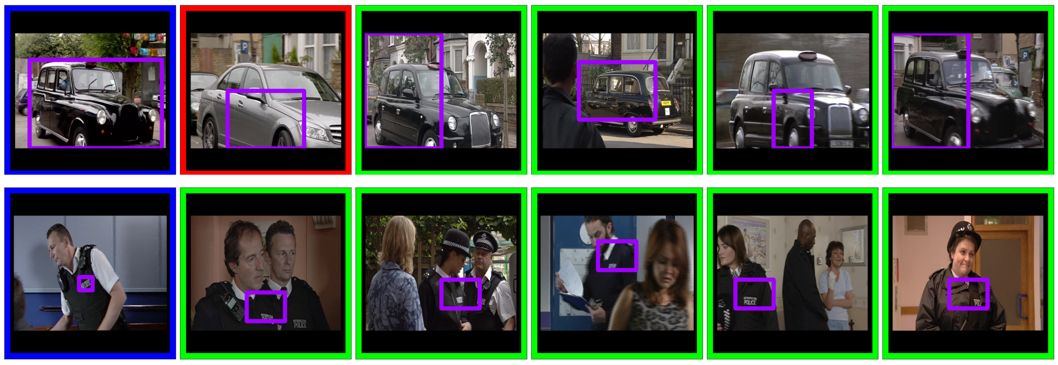


Figure 4.9: Top 5 rankings for queries 9072 (top) and 9081 (bottom) of the TRECVID INS 2013 dataset.

Table 4.5: mAP of sum pooling and BoW aggregation techniques in Oxford, Paris and TRECVID INS subset.

		Oxford 5k	Paris 6k	INS 23k
Ours	GS	0.650	0.608	**0.323**
	LS	**0.739**	**0.819**	0.295
Sum pool (as ours)	GS	0.621	0.712	0.156
	LS	0.583	0.742	0.097
Sum pool (as in [?])	GS	0.672	0.774	0.139
	LS	0.683	0.763	0.120

4.7 Discussion

The proposed BLCF representations benefit from the sparsity of BoW encoding. However, the strategy of interpolating the target features to generate higher resolution feature maps directly affects the efficiency of our representation. For instance, an image of resolution $(340, 256)$ would generate feature maps in conv5_1 of dimensions $(21, 16, 512)$ which, after the interpolation has $(42, 32, 512)$ dimensional feature maps. This results in obtaining a maximum number of visual words of 1344/image, in contrast to the 336/image of the non-interpolated feature maps.

We chose to interpolate only the feature maps of the query images. This produces a small drop in performance as shown in Table 4.6. However, the memory requirements are reduced by a factor of 4. For this reason, in the remainder of this thesis, we only interpolate the query features.

In comparison with sum/max-pooling approaches that generate a 512-dimensional vector per image [? 125, 60, 106, 8], the memory required for loading BLCF

Table 4.6: Performance of BLCF in Oxford 5k and Paris 5k when interpolating the feature maps of all the images within a dataset and interpolating only the query images.

	Oxford 5k		Paris 6k	
	GS	LS	GS	LS
Interpolating all	0.653	0.738	0.699	0.820
Interpolating queries	0.628	0.722	0.642	0.798

Table 4.7: Memory (MB) of the indexed dataset, and query time search (s) per image using BLCF representations. We also report the average number of non-zero elements in the BoW histogram. The time refers to the time required to perform an initial search. The re-ranking of the top 100 images increases the query time to 8.5s.

	Oxford	Paris	TRECVID
memory (MB)	3.47	4.11	31.47
words/image	171	160	285
query/time (s)	0.002	0.003	0.02

representation is much smaller. For instance, the amount of memory required for loading all images of Oxford 5k dataset is 3,47MB. A dense 512-dimensional representations would require 10.37MB. In the case of TRECVID, BLCF requires 31.47MB of RAM where the dense approach needs 48.63MB. Table 4.7 shows the memory required to load different datasets and the query time required to retrieve one query in an indexed dataset (using an Intel(R) Core(TM) i7-4790K CPU @ 4.00GHz with 30GB of RAM). While the retrieval time for a query is very efficient, the larger bottleneck is due to the re-ranking stages, which increases the query time to 8.5s.

The time required for the re-ranking stage could potentially be reduced by parallelizing the computations conducted in each image. However, in this thesis we focus on exploring different ways to improve the underlying BoW representation to keep the search time efficient (only considering an initial search with a simple post-processing step, such as average query expansion). In Chapter 5 we explore a fine-tuning strategy to improve the CNN local features, and in Chapter 6 we propose a better weighting scheme for aggregating the visual words into a BLCF. With these methods, we obtain high performance rankings without including any computationally demanding step at query time.

4.8 Conclusions

We proposed an aggregation strategy based on Bag of Words to encode features from convolutional neural networks into a sparse representations for instance search.

We use the assignment maps of the image to conduct spatial search to re-rank and improve precision of the retrieved images. This re-ranking is followed by average query expansion, which benefits from the previous re-ranking stage to provide more informative queries. We demonstrate the suitability of these bags of local convolutional features by achieving competitive performance with respect to other CNN-based representations in Oxford and Paris benchmarks, while being more scalable in terms of index size, cost of indexing, and search time. We also compared our BoW encoding scheme with sum pooling of CNN features for instance search in the far more complex and challenging TRECVID instance search task, and demonstrated that our method consistently performs significantly better.

This encouraging result suggests that BoW encoding, by virtue of being high dimensional and sparse, is more robust to scenarios where only a small number of features in the target images are relevant to the query. Our method does, however, appear to be more sensitive to large numbers of distractor images than methods based on sum and max pooling (SPoC, R-MAC, and CroW). We speculate that this may be because the distractor images are drawn from a different distribution to the original dataset, and may therefore require a larger codebook to better represent the diversity in the visual words. Future work will investigate this further.

In the next chapter, we explore the combination of the proposed system with a traditional SIFT-BoW system with spatial verification to further improve the obtained results. With the improved retrieval pipeline we sample images for fine-tuning a similarity model in an end-to-end manner with the aim of obtaining better CNN representations than those extracted from the pre-trained classification models.

Chapter 5

Fine-tuning CNN models for Instance Search

5.1 Introduction

In the previous chapter we proposed BLCF as an efficient pipeline to perform instance search. In this chapter, we explore a fine-tuning strategy to improve the local CNN representations and generate a more powerful BLCF representation.

Descriptors extracted from off-the-shelf CNN classification models trained on millions of labeled images have been proven to encode in their middle layers "general purpose" image information useful for a diverse range of computer vision problems [110]. However, the target task of these models is substantially different to instance search task. While classification is concerned with distinguishing between different classes, learning representations invariant to the possible intra-class variability (i.e the different appearances of an object), instance search is concerned with identifying concrete instances of a particular class.

For this problem, deep models tend to perform worse than conventional methods that rely on high dimensional handcrafted features with high discriminative power followed by geometric verification of local descriptors between a matched image pair and query expansion [96, 3, 86, 124].

Recently, different authors have proposed to explicitly learn the weights of a CNN model for the retrieval task [1, 100, 39], instead of using off-the-shelf models, showing a substantial improvement over off-the-shelf CNN representations and achieving a new state-of-the-art in the Oxford, Paris, and Holidays benchmarks. However, the proposed approaches require generation and annotation of a suitable training image set for similarity learning that, in the case of the best performing CNN-retrieval models [100, 39], is domain specific to landmarks and buildings.

In this chapter, we first explore a method to generate high quality ranks in a particular dataset by combining on off-the-shelf CNN retrieval system and SIFT-based system. Then, we use the merged system to generate a set of triplets to perform similarity learning of a simpler CNN model. We evaluate the effect of performing fine-tuning in different domains, finding that fine-tuning a CNN model for similarity learning is a good strategy to perform instance search in a particular image domain, but not to generate good general-purpose retrieval representations.

The remainder of this Chapter is organized as follows: Section 5.2 describes the two SIFT and CNN based systems used to obtain a high quality retrieval pipeline, Section 5.3 describes the network architectures considered as well as the loss function and training sample strategy to optimize a CNN for similarity learning. Section 5.4 presents the experiments and results obtained using different datasets for network optimization, and lastly Sections 5.6 and 5.7 present the summary and conclusions drawn from the experiments conducted.

5.2 Combining SIFT and CNN-based systems

Descriptors from pre-trained classification networks encode high-level information of the image but are highly invariant to intra class variability. On the other hand, invariant local descriptors, encode low-level image information robust to geometric transformation but they do not encode image semantics. In this subsection we propose to use the best of the two approaches by combining the rank scores obtained from our proposed BLCF system (Chapter 4) with the ones obtained from a SIFT-BoW system. The purpose is to generate a high-quality ranked list of images from which to sample training examples for similarity learning.

5.2.1 SIFT-BoW system

We use a retrieval system based on invariant hand-crafted features. More specifically, we use affine-Hessian interest points [85] from which we extract RootSIFT [3] local representations for each image. We learn a visual vocabulary of 500,000 centroids using approximate k-means and spatial re-ranking on the top-1000 retrieved results. BoW representations were built using tf-idf weighting [114]. For this, we use the Matlab implementation from Andrea Vedaldi [128]. We test the system on Oxford [96], Paris [97], and INSTRE [133] benchmarks. To evaluate the performance on INSTRE, we randomly sample 5 images from the 250 different object classes in the dataset generating a total of 1250 image queries. We use mean average precision as the evaluation metric, following the same procedure as in [48].

5.2.2 BLCF system

We use the same system proposed in Chapter 4 using the bounding box to represent the query and with global average query expansion of the top-10 ranked results. We replace the re-ranking strategy by saliency weighting (explained in detail in Chapter 6), which represents a more general solution in contrast to the Gaussian weighting, and allows us to achieve similar results without explicitly analyzing the images at a local level generating multiple window descriptors.

The procedure consists of first extracting local features from the pre-trained VGG16 network, working at one third of the original image resolution. Features are L^2-normalized, PCA-whitened and L^2-normalized again. A visual vocabulary of 25,000 clusters is learned, and the final BoW descriptor was built applying saliency weighting on the visual words for the BoW construction. In both SIFT (Section 5.2.1) and CNN based systems, the query bounding box is used to encode only the visual words within the relevant area to generate the query BoW representation.

5.2.3 Combined system

We evaluate the performance of a merged system by directly averaging the similarity scores obtained by SIFT and CNN methods. Table 5.1 shows how performance of the combined system consistently outperforms the individual pipelines obtaining nearly state-of-the-art performance in Oxford and Paris, and improving results in the INSTRE dataset. [1]

Table 5.1: Performance of merged SIFT and CNN BoW systems across different datasets.

	Oxford	Paris	INSTRE
BLCF	0.795	0.843	0.737
SIFT-BoW	0.865	0.803	0.382
merged	**0.904**	**0.914**	**0.739**

5.2.4 Quantitative comparison

The SIFT-BoW system achieves high performance in the landmark related datasets Oxford and Paris, obtaining 0.865 and 0.803 mAP respectively and drops its performance in INSTRE obtaining 0.382 mAP. BLCF obtains good performance on the landmark-related datasets, obtaining 0.795 and 0.843 mAP in Oxford and Paris. Similarly to the SIFT-based system, BLCF also experiences a drop on performance in the INSTRE dataset. However it is far less severe, obtaining 0.737 mAP.

This evaluation leads us to two conclusions. The first is that instance search methods that obtain high-performance in datasets such as Oxford and Paris do not generalize well for real world photo collections, where the background and context in

[1]The current state of the art in Oxford and Paris datasets is 0.947, and 0.966 mAP respectively. These results are reported [39], where the authors use a fine-tuned version of R-MAC (using ResNet [43] architecture), trained for similarity learning on landmark images (more details are provided in sections3.5.3.4 and section 3.5.5), including a database side augmentation and query expansion as post-processing steps [3]. The current state-of-the-art in INSTRE is 0.896 mAP, where a diffusion strategy on the similarity scores is used to re-fine the results from the fine-tuned R-MAC representations [39].

which the instance may appear are more diverse, and instances are not restricted to a particular domain or specific visual patterns. The second is that CNN-based descriptors encode information in such a way that they outperform low-level SIFT representations in this kind of scenario. As such, this suggests that the approaches are complementary and performance will often improve if they are combined.

In the case of INSTRE, images returned by the merge system are more or less unchanged in most of the cases, and SIFT system only seem to add insignificant noise to the meaningful results returned by the CNN system. However, for some of the queries, CNN fails in returning significant results. In those cases SIFT system helps to improve the performance of the BLCF system.

5.2.5 Qualitative comparison

Whilst both methods present generally high-performance in buildings and landmark related queries, a more exhaustive analysis of INSTRE allows us to compare the kinds of scenarios in which one outperforms the other. We observe that SIFT is superior in planar and textured instances such as logos as shown in Figure 5.1. In the two examples, it is possible to observe how the semantics captured in the CNN representations make the system fail to identify the two particular logos.

On the other hand, Figure 5.2 contains two examples where BLCF outperforms SIFT-BoW. We observe that 3D objects with textureless and/or reflective surfaces or instances where colour can be one of the most discriminative features are cases where CNN outperforms SIFT-based system (first example of Figure 5.2). We also observe how the lack of semantics of SIFT-BoW causes some failures. Particularly, in the case of retrieving instances of the *Mount Rushmore*, SIFT-BoW system mistakes the instance of the mountain with other pictures with a rocky background.

Instead of considering both systems independently, the qualitative analysis suggests that the information encoded by the different representations can be complementary, since for some of the instances one method clearly outperforms the other.

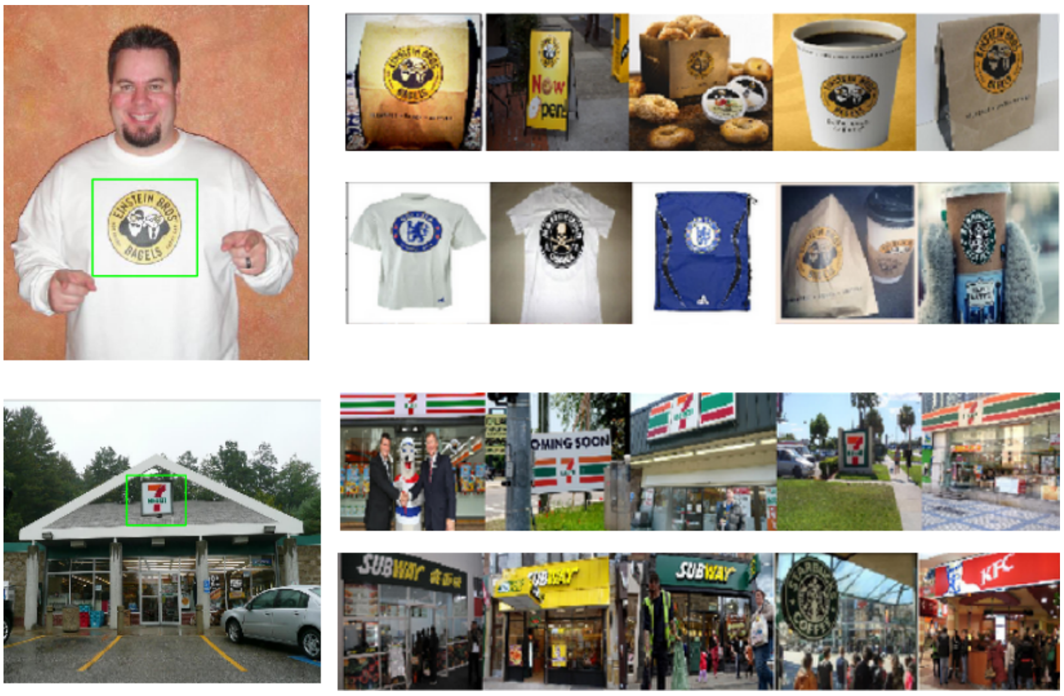


Figure 5.1: Two visual examples containing instance of logos where SIFT outperforms the CNN system. For each example, the top-5 results are displayed (first row SIFT, the second CNN). The first example, the *Einstein Bros* logo achieves 0.946 Average Precision (AP) with SIFT whereas CNN achieves 0.159 AP, where the system retrieves logos but fails in identifying the specific instance. Similarly, the *Seven Eleven* logo achieves 0.927 in SIFT and 0.048 AP in CNN, where the system retrieves buildings with a logo at the top but fails in identifying the specific instance.

5.3 Fine-tuning a CNN model for similarity learning

Inspired by the discussion of fine-tuning in Chapter 4, we note that the combination of CNN and SIFT features provides high quality ranks that we can use to sample triplets for similarity learning (described in Chapter 3). The aim is to fine-tune a “simpler” CNN model to behave like the complex and expensive merged system without the need for crawling and annotating additional data.

5.3.1 Network architecture

We start from a pre-trained VGG16 model [113] without the fully-connected layers. Local descriptors are encoded with the following configurations:

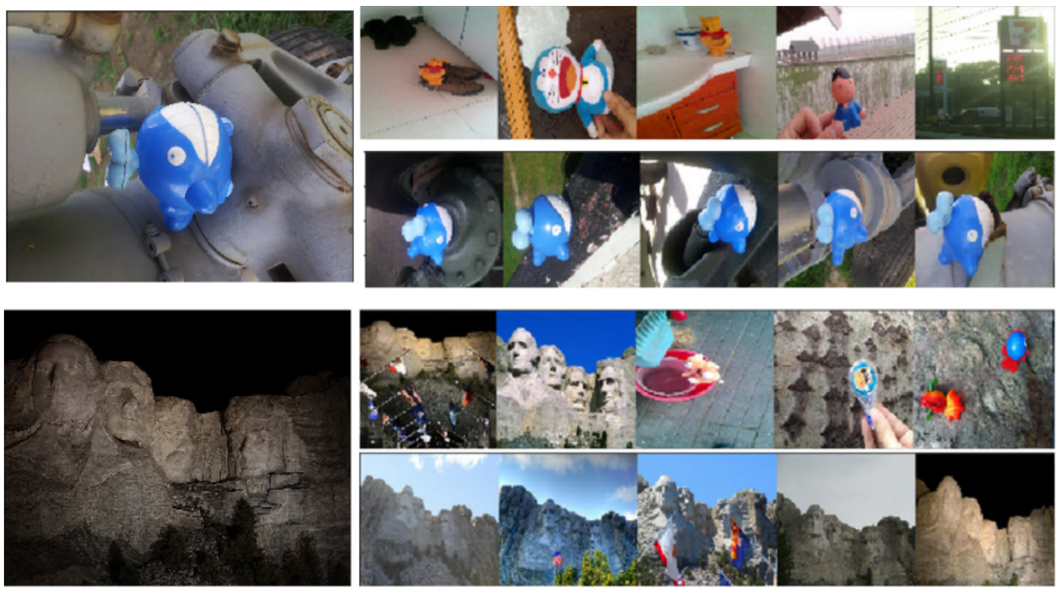


Figure 5.2: Two visual examples containing instances where CNN outperforms the SIFT system. For each instance the top-5 ranked results are displayed (first row for SIFT descriptors, and second row for CNN). The first example contains an instance of a whale toy. In this case, one of the most discriminative features is the colour of the object, which is the reason why the system achieves low performance using SIFT descriptors 0.133 AP, but very high performance using the CNN representation 0.964 AP. In the second example, we observe how the lack of semantics of the SIFT-based system produces a ranked list with similar textured images, but fails in identifying the instance achieving only 0.033 AP whereas the CNN system achieves 0.984 AP.

1. (Network A) *Direct sum-pooling*: Descriptors from the *pool5* layer are sum-pooled generating a single representation per image of 512 dimensions, that is further L^2-normalized.

2. (Network B) *Direct sum-pooling in a high dimensional space*: We add a new convolutional layer on top of *pool5* with 4096 filters of size $(1 \times 1 \times\ 512)$ to project the local descriptors into a higher dimensional space. Descriptors are sum-pooled generating a single representation of 4096 dimensions that is L^2-normalized.

5.3.2 Loss function

We use the triplet loss function introduced in Chapter 3:

$$\mathcal{L}(\mathbf{x_a}, \mathbf{x_p}, \mathbf{x_n}) = \frac{1}{2} \max(0, D(\mathbf{x_a}, \mathbf{x_p}) - D(\mathbf{x_a}, \mathbf{x_n}) + \alpha), \tag{5.1}$$

where $D(\mathbf{x_a}, \mathbf{x_p})$ represents the cosine distance between an anchor and a positive example, $D(\mathbf{x_a}, \mathbf{x_n})$ is cosine distance between the anchor and a negative example and α is the margin. The non-zero margin value can be chosen arbitrarily since the CNN can learn to globally scale the embedding proportional to α. The only important factor is the relative scale of the margin and the embedding at initialization [11]. We set $\alpha = 0.5$ without experiencing any divergence problem during training.

5.3.3 Triplet sampling

We explore the similarity scores obtained from the SIFT-CNN merged system to sample training triplets from a dataset: for a particular rank, high-scored images are associated with images containing the same instance whereas low ranked correspond to unrelated images.

We query all the images within a dataset using the combined SIFT and CNN system to generate a ranked list, which corresponds to the average of the similarity scores of both systems. We randomly sample one of the images (or generated ranked lists) to sample a triplet for training. For each ranked list, the similarity scores[2] larger than one are associated with those images that had been spatially verified with RANSAC on the SIFT local representations. We select a positive pair (anchor and positive image) randomly, selecting two images with score larger than 15 (sharing more than 15 local SIFT matches) to ensure that the positive pair was geometrically verified and shares a large amount of SIFT local matches. The negative pair is selected from all images within a ranked list with score between 0.5 and 1, where 1 is the maximum similarity score achieved by images that have not been spatially verified with the SIFT pipeline. Additionally, we conduct negative mining by computing

[2]For the CNN system, the similarity scores are the cosine similarity between image representations. For the SIFT system, the similarity score consists in the cosine similarity of the BoW representation in addition to the number of SIFT matches obtained by RANSAC, being a number larger than one for the spatially verified images.

the loss associated with random triplets and sampling those with higher value for training.

5.4 Experiments and results

Training images were processed by taking a squared random crop of dimensions equal to the smaller image side. The crop was resized to 224×224 dimensions. Due to memory restrictions, we only consider fine-tuning the weights from the last convolutional layers (*conv5_1*, *conv5_2*, and *conv5_3*). We tested different optimization methods (SGD, Adaboots, and RMSProp), selecting RMSProp [123] since it was the only which we achieved loss convergence. We used RMSprop with learning rate set to 0.01, with epochs of 5,000 triplets for training and 3,000 for validation. For testing the performance of the fine-tuned representations, images were resized keeping their original aspect ratio and setting their maximum dimension to 340 pixels.

5.4.1 Fine-tuning using Oxford query ranks

In this first experiment, we sample images from rankings related to the Oxford queries, similarly to the training strategy followed in [131], where the authors directly use the ground-truth labels to sample the triplets. We trained Network A for 60 epochs, doing negative mining every 15 epochs with a batch size of 100 triplets. Performance of the baseline network was substantially improved on the Oxford dataset. The baseline system of L^2-normalized sum-pooled *pool5* descriptors from pre-trained VGG16 generated 0.480 mAP on Oxford, which was improved by the fine-tuned network to 0.733 mAP.

However, sampling triplets only from query rankings made the network highly overfit to the Oxford queries as shown in Table 5.2. Even when evaluating the performance of the learned representations in a domain related dataset, such as the Paris dataset, the descriptors showed very poor performance in comparison to the pre-trained descriptors.

Table 5.2: Performance (mAP) of fine-tuned models using ranks related to the Oxford queries. Network A refers to direct sum-pooling configuration. The baseline is the performance of the off-the-shelf features.

	Oxford	Paris
Baseline	0.480	**0.698**
Network A	**0.733**	0.247

5.4.2 Fine-tuning using ranks from all Oxford dataset

In this experiment, we follow the same experimental setup described as the previous subsection, but instead of restricting the training data to the Oxford queries, which only represents 11 different landmarks, we sample triplets from the full dataset. In this case the learned representation presents less overfitting to the Oxford dataset, but also less improvements: Network A only achieved 0.575 mAP on Oxford.

With the same setup, we also fit an additional set of weights to project local representations to a higher space (Network B): Local information is less likely to interfere when pooling the descriptors via sum-pooling in this higher dimensional space. In this case the batch size for training is reduced to 20 images due to memory restrictions. However, the gains obtained after training this configuration are equal to the ones provided by adding a set of random weights (Baseline B), which suggests that even though the network decreases its loss during training, the weights of this layer are not capturing any useful information.

It is interesting to see how the extra layer added in Network B represents an increase in performance by itself, even with randomly initialized weights. This behaviour has been observed in [107], where the authors notice that certain feature learning architectures can yield useful features for object recognition tasks even with untrained random weights. Random projections are also exploited in Extreme Learning Machines (EKM [46]), which are supervised learning architectures consisting of one hidden layer where the projections between the input and the hidden neurons

Table 5.3: Performance (mAP) of fine-tuned models using all ranks generated in the Oxford dataset. Network A refers to direct sum-pooling configuration. Network B refers to adding an extra high-dimensional layer. Baselines are the performance of off-the-shelf (Network A) and off-the-shelf with randomly initialized weights (network B) in the additional layer.

	Oxford	Paris
Baseline A	0.480	0.698
(F)Network A	**0.575**	0.302
Baseline B	0.503	**0.716**
(F)Network B	0.502	0.715

are randomly set and the only parameters updated during training are the weights connecting the hidden layer with the output layer.

Results show that using all the dataset images in the Oxford dataset is a way to reduce the substantial over-fitting that occurred in the previous section. However, the information learned in Network A is still not useful in the Paris dataset. One of the reasons may be that most of the images are references to one of the 11 queries, and the rest of the images are completely unrelated images. The low diversity of the instance classes justifies the drop in performance in the Paris dataset, despite representing a similar domain. Also, the limited size of the dataset makes it infeasible to fit the large number of weights introduced in the additional last layer of Network B.

5.4.3 Fine-tuning using INSTRE class labels

In this section we use the instance classes provided in the INSTRE dataset to perform fine-tuning of the two proposed architectures: Two random images are sampled from a particular class (anchor and positive examples) and the negative is randomly sampled from another class. We do not use the unsupervised sampling strategy described in Subsection 5.3.3 to avoid sampling noisy triplets in this first experiment.

Additionally we add an extra configuration by including a classification loss to predict the class of the anchor image during training. For this we add a dense layer with a softmax activation function on top of the sum-pooled 4096 dimensional feature. The new dense layer has 250 dimensions, which correspond to the 250 different instance classes of the dataset. This architecture is identified as "(F+cl) Network B" in the results of Table 5.4. The final loss is an equally weighted combination of the similarity loss defined in Equation (5.1) and the categorical cross entropy.

Table 5.4: Performance (mAP) of fine-tuned models using INSTRE images for training. Network A refers to the direct sum-pooling configuration. Network B refers to adding an extra high dimensional layer. Baselines are the performance of off-the-shelf (Network A) and off-the-shelf with randomly initialized weights (Network B). (F) refers to fine-tuning for similarity learning. (F+cl) refers to fine-tuning for classification and similarity learning simultaneously.

	Oxford	Paris	INSTRE
Baseline A	0.480	0.698	0.275
(F)Network A	0.071	0.147	0.257
Baseline B	**0.503**	**0.716**	0.268
(F)Network B	0.498	0.712	0.268
(F+cl)Network B	0.246	0.242	**0.587**

We observe that similarity learning in all cases but one does not improve mAP in the INSTRE dataset, even when using the image labels to generate "clean" triplets: The more diverse setup presented in the INSTRE dataset makes it challenging even to over-fit the training set for instance search. Only when we combine similarity learning with classification loss can we observe substantial gains in the INSTRE dataset. However, the learned descriptors only become useful in the INSTRE dataset as they do not generalize well to other domains, as shown in Table 5.4 with the (F+cl) Network B configuration.

5.5 Discussion

Even-though the fine-tuned descriptors show a strong over-fitting to the dataset from where we generate the training data, we nonetheless test the performance of BLCF encoding on the derived local CNN features. We use the Network A (subsection 5.4.1), which achieves 0.733 mAP performance with sum-pooled features on *pool5* in Oxford.

Table 3.3 shows the results using the descriptors from *conv5_1* of VGG16. BLCF on *pool5* obtains lower performance compared with sum-pooling aggregation (0.660 mAP vs 0.773), which is not surprising since the network has been tuned to aggregate *pool5* features with sum-pooling. However, we observe that BLCF on the lower layer *conv5_1* benefits from the improved CNN representations, and their performance can be furthered improved by applying a weighting scheme on the obtained assignment maps.

Future work will investigate a way to formulate a soft version of BoW so that it can be included as a layer in the CNN architecture, as proposed with VLAD encoding in [1]. However, the high dimensionality of the BoW vocabularies represents a challenge for formulating a soft version of BoW as an additional layer for the CNN. For instance, using 25,000 centroids and implementing the encoding as a fully connected layer followed with a softmax means that it would be necessary to tune 12,800,512 parameters. This proposed layer would contain more parameters than the full VGG16 network (8,313,3344 parameters), which is an unrealistic solution for practical reasons (this amount of parameters might not fit in memory at training/testing time), and the large amount of data required to conduct the training. Techniques used in natural language processing for training large vocabularies [40] could be applied for fitting the vocabulary layer. Methods such as hierarchical softmax [87] use tree structures to make the softmax computation more efficient.

Table 5.5: Performance of BLCF on *conv5_1* encoding using using different weightings in the Oxford 5k dataset. GS refers to global search (encoding all the visual words of an image), LS refers to encoding only the visual words within the query bounding box. “None” uses equal weights on the visual words for constructing the histogram, “Gaussian” assigns more weight to the visual words located in the center of the image (Chapter 4), “Saliency” uses the weights obtained from a Saliency model [143], proposed in chapter 6)).

	None		Gaussian		Saliency	
	GS	LS	GS	LS	GS	LS
off-the-shelf	0.628	0.722	0.666	0.728	0.670	0.746
fine-tuned	**0.669**	**0.739**	**0.708**	**0.736**	**0.731**	**0.750**

Also, importance sampling [12] can be applied to allow efficient training of the high dimensional layer. We plan to explore these avenues in future work.

In this chapter we have observed that larger training datasets are crucial to achieve generalization and better performance. Recent works show that with a suitable dataset, off-the-shelf classification models can be improved for retrieval [39, 100].

In the remainder of this thesis we investigate how designing better weighting schemes can substantially improve the performance of BLCF based on off-the-shelf CNN features, representing an approach that generalizes well on different retrieval domains without requiring the collection of a suitable training dataset.

5.6 Summary

In this chapter we have explored a similarity learning strategy to fine-tune a CNN network for instance search, evaluating its generalization across different domains. We focused on exploring existing datasets to sample triplets without requiring additional images and manual annotations.

We first proposed the combination of CNN and SIFT-BoW models to build an accurate retrieval system that takes the best from both representations. We justify this by showing that both methods are complementary and their combination improves results over individual pipelines.

We explored different fine-tuning strategies, mainly differing in the training samples used:

- The first strategy consisted of sampling triplets from rankings belonging to the query images within the Oxford dataset. The approach provided substantial gains in the Oxford dataset at the cost of over-fitting to the query images, which makes the learned model effectively useless for retrieving different queries, even domain related ones.

- The second strategy consisted of using rankings related to all Oxford images instead of only the rankings of the image queries. We found that this approach still overfits the Oxford dataset images. We speculate that this is due to the low diversity of the instance classes (related mainly to the 11 buildings) for fine-tuning.

- The third strategy consisted of sampling triplets from the general-purpose instance retrieval dataset INSTRE. We found that training a network for similarity learning in this setup did not generate image representations that produced any gains in the performance of instance search. In this case, the high diversity on the query domains makes the similarity learning task very challenging. Only when we add a classification loss did we observe gains in the INSTRE dataset. However, this was at the cost of overfitting the dataset.

5.7 Conclusions

Exploring existing retrieval benchmarks to perform fine-tuning of a CNN for similarity learning was also explored in Want *et al.* [131], where the authors use the ground-truth labels to generate the training images, reporting substantial gains with respect to the performance of the baseline system based on pre-trained features. We follow a similar approach, proposing a merged SIFT-CNN BoW system to generate a set of rankings from which to sample images in an unsupervised manner. However, using datasets such as Oxford or Paris for similarity learning is likely to generate models that highly overfit the training set, due to the low diversity of the landmark instances.

In contrast, using images from the INSTRE dataset, which is much more diverse in the domain of the queries, results in making the task of learning a good general image representation for instance search task very challenging. We speculate that to successfully fine-tune a CNN model for similarity learning it is necessary to have a large diversity in the instances but also to restrict those instances to be of a specific domain such as in [1, 100, 39]. This also suggests that many similarity learning approaches that rely on fine-tuning either overfit the dataset or overfit the particular type of task being evaluated (i.e landmarks, buildings), making them less likely to generalize well to more general instance search scenarios. We could possibly improve the performance of the proposed CNN networks by working with a larger and more suitable training set of images, by collecting a new set of images specific to a particular domain from which to generate a set of ranked lists for sampling the triplets.

However, pre-trained CNN models already generate good "general purpose" features, that we could further explore for the task of instance search, without restricting them to any particular domain. Therefore, in the remainder of this thesis, we will focus on improving the performance of instance search system using only pre-trained representations. Specifically, in the next chapter we propose to explore saliency models to weight the contribution of convolutional features.

Chapter 6

Saliency Weighted Convolutional Features

6.1 Introduction

In this chapter we evaluate different weighting schemes in an attempt to improve BLCF representations. As a result, we obtain a high performance and efficient retrieval pipeline that does not require any additional re-ranking step involving the individual processing of different regions of the image. The approach also does not require the construction of a suitable training dataset for fine-tuning.

One common factor that seems to improve performance of a CNN model (fine-tuned or not) used in a retrieval pipeline is to individually analyze different regions of the images. This can be performed by pooling small regions within a convolution layer or by aggregating the visual words associated to a particular image region. In particular, R-MAC has become a popular architecture for image retrieval [39, 100, 73, 17, 69]. Originally proposed by Tolias *et al.* [125], R-MAC is specifically designed for the instance search task. The main feature of their approach consists of building a compact image representation by sum-pooling features of different regions of the image. They follow a sliding window approach to building the region features, considering different window sizes. In contrast to Razavian *et al.* [101], their method maps the window coordinates within the last convolutional layer (similar to FastRCNN [37]), so only a single image has to be forwarded through the network to generate all the region descriptors via max-pooling. All the regions are post-processed with am L^2-normalization, PCA whitening, and a second round of L^2-normalization. The final representation is constructed by sum-pooling all the region descriptors followed by a L^2-normalization.

Intuitively, one can see that pooling descriptors from different regions and then subsequently pooling all of the resulting regions together is similar to applying a weight to the original convolutional features, since region descriptors and globally pooled descriptors are built from the same set of local features. In this chapter we first investigate whether R-MAC can be re-interpreted as a non-parametric weighting scheme on the convolutional features. Second, we evaluate different ways to weight the convolutional features for the task of instance search, finding saliency weighting an especially beneficial approach on different datasets. Finally, we show that our proposed bags of convolutional features can benefit more from saliency weighting than direct pooling approaches, achieving comparable results to models that have been explicitly tuned for the image retrieval task.

The chapter is organized as follows: Section 6.2 provides a brief overview of the taxonomies for weighting schemes for convolutional activations. Section 6.3 motivates the use of sum-pooling as a direct pooling mechanism over max-pooling. Then, Section 6.4 provides a re-interpretation of R-MAC as a weighting scheme, which allows us to re-interpret the object proposal algorithms as a kind of weighting as shown in Section 6.5. Section 6.6 provides a brief overview about saliency models. Section 6.7 describes the experiments conducted. Sections 6.8 and 6.10 present the quantitative and qualitative results obtained. Section 6.9 compares the our system with the state-of-the-art. And finally, Sections 6.11 Sections 6.12 and 6.13 conclude the chapter with a summary and conclusions for this study.

6.2 Taxonomy of weighting schemes

The major advantage of using convolutional layers over fully-connected ones is that they retain the spatial information of the local image patterns. A convolutional layer generates a tensor of activations $\mathcal{X}$ in $\mathbb{R}^{H\times W\times D}$, where (H, W) is the spatial dimension of the feature maps and D the total number of feature maps. $\mathcal{X}_{i,j,k}$ refers to an activation located in the spatial location (i, j) in the feature map k. As discussed in Section 3.4.1, the volume of activations can be reinterpreted as $N = H \times W$ local descriptors $f(i, j) \in \mathbb{R}^D$ arranged in a 2D space. Before aggregating the activations into a single representation, it is possible to apply a weighting scheme $w_{i,j,k}$ to weight the contribution of each activation, so that weighted activations $\mathcal{X}'_{i,j,k}$ are computed by $\mathcal{X}'_{i,j,k} = w_{i,j,k}\mathcal{X}_{i,j,k}$.

6.2.1 Types of weighting

We can distinguish three types of weighting: weights applied across feature maps of dimension D, weights across the spatial dimension (H, W) of the feature maps, or hybrid approaches.

6.2.1.1 Feature weighting β_k

Each dimension of a local descriptor $f(i, j)$ represents the response associated with a particular feature detector. Different weights can be applied to the features of the local vectors, so that the weighting is uniform across the spatial dimensions of the feature maps $w_{i,j,k} = \beta_k$. For instance in [**?**], β_k is a weighting scheme that weights the contribution of each dimension according to the sparsity associated with their global feature map k in $\mathcal{X}$. The proposed scheme up-weights the contribution of rare-features (dimensions corresponding to feature maps with higher sparsity levels) before aggregation. This procedure is similar to the traditional *tf-idf* weighting scheme [114], that also boosts the importance of the most distinctive and rare visual words within an image in BoW encoding.

6.2.1.2 Spatial weighting $\alpha_{i,j}$

In this approach different weights are applied to $f(i,j)$ depending on its spatial location. In this case the final weighting is uniform across different feature maps $w_{i,j,k} = \alpha_{i,j}$. For instance, $\alpha_{i,j}$ is defined using a Gaussian weighting scheme in [8] and, similarly, in [**?**] each local feature is weighted with its associated L^2 norm.

6.2.1.3 Hybrid weighting $w_{i,j,k}$

In this approach activations are weighted by combining β_k and $\alpha_{i,j}$ schemes. An example of this is the Cross-Dimensional Weighting scheme (CroW) [**?**] where the weighting is defined as $w_{i,j,k} = \alpha_{i,j}\beta_k$. The weight assigned to a particular activation is a combination of the weights derived from the channel sparsity and the weights associated to the strength (L^2-norm) of each local descriptor.

More specifically, subsections (6.2.1.2) and (6.2.1.1) describe a weighting types can be seen as a particular case of the hybrid weighting CroW, where $\alpha_{i,j} = 1$ in the case of feature weighting and $\beta_k = 1$ for the spatial weighting types.

6.2.2 Spatial weighting schemes

Spatial weighting $\alpha_{i,j}$ weights the contribution of each local feature $f(i,j)$ before the aggregation stage. We can further classify four different schemes for this type of weighting: *fixed*, *image-dependent*, *query-dependent*, and *hybrid weighting schemes*.

6.2.2.1 Fixed weighting schemes

In this approach, the weighting is a function that does not depend on the image content (or local descriptors). Examples include global sum-pooling of unweighted activations ($\alpha_{i,j} = 1$) or global sum-pooling with spatial weighting following a Gaussian centre-prior is defined by:

$$\alpha_{i,j} = \exp - \left(\frac{(i - \frac{H}{2})^2 + (j - \frac{W}{2})^2}{2\sigma^2} \right). \tag{6.1}$$

These schemes do not depend on the image content (as can be seen from Equation 6.1): the weighting only depends on the image resolution.

6.2.2.2 Image-dependent weighting schemes

Here weights are a function of the image content. Schemes can be classified into two main categories:

1. *Non-parametric image-dependent weighting* where weights do not depend on any learned parameters. For instance, in global sum-pooling of features spatially weighted with the contribution of the local L^2-norms [**?**], weights are defined as

$$\alpha_{i,j} = \|f(i,j)\|^2 \tag{6.2}$$

   where $\alpha_{i,j}$ is a non-parametric function that depends on the local descriptor. Similarly, in the same spirit as R-MAC [125], local sum-pooling of different regions followed by L^2 region normalization and global pooling is also an example of this scheme, where each region is inversely weighted according to its associated L^2-norm (as will be shown in Section 6.4.2).

2. *Parametric image-dependent weighting* where the weighting function is parametric and fit using supervised learning. One example is the local sum-pooling of regions obtained from a learned region proposal algorithm followed by L^2 region normalization and global pooling, as in the fine-tuned R-MAC version [39]. Another example is the saliency weighting of convolutional activations. This scheme is particularly interesting because saliency models are trained to directly up-weight parts of the image that humans consider important. The saliency weighting scheme is furthered discussed in Section 6.6.

The above categorization ignores PCA whitening for the sake of simplicity. PCA whitening performs an additional linear transformation of the weights that equalizes the expected covariance of the features. PCA is not a weighting scheme, as such, but rather a projection of the features onto an alternative basis. It can be added to any of the above approaches as a post-processing step, either after local pooling or global pooling.

6.2.2.3 Query-dependent weighting schemes

In this case the spatial weighting is a function of the query image (or its representation). As with the image-dependent weighting schemes, the weighting can be non-parametric or parametric.

An example of this weighting is described in [17], where the authors propose a query-adaptive image search re-ranking where convolutional activations are locally pooled within different regions similarly to [125]. However, instead of directly pooling all the regions via sum-pooling into a final representation, a soft merging function of the set of regions is optimized for each query. The function up-weights the contribution of the regions that are more related to the query. Due to the fact that the total amount of dataset regions can be prohibitively large for the optimization algorithm, the method is only applied as a re-ranking step, considering the regions from the top retrieved images obtained from an initial search.

Although this weighting scheme represents a potentially interesting avenue of research, its main drawback is that it needs to be optimized at query time, adding a significant computational computational load as well as memory demand, since all region descriptors need to be stored, making the system less scalable.

6.2.2.4 Hybrid schemes

This type of scheme involves both image-dependent and query-dependent weighting. Although this is theoretically possible, the scheme would suffer from the same drawbacks as the query-dependent schemes. Again, it could be an interesting avenue of future research.

6.3 Sum/Max-pooling

While sum-pooling seems to outperform max-pooled features when whitening is applied as a post processing step [8**?** , 125], R-MAC is based on max-pooling activations within the different regions. Here we investigate if one of these two pooling strategies out-performs the other. We find that there is no clear evidence favoring one pooling strategy over the other, since hyperparameters such as image resolution, feature post-processing (applying PCA whitening or not), the layer of the network considered, weighting scheme (SPoC, CroW, etc.), or even the testing dataset in which we evaluate the retrieval pipeline can play a role in deciding whether max-pooling outperforms sum-pooling.

As an example, we describe a small experiment to evaluate the performance of max and sum-pooling. We represent the images by spatially pooling all the activation within the last convolutional layer of a pre-trained VGG16. As a preprocessing step the image is rescaled so that the larger dimension is 340 pixels (maintaining the original aspect ratio). The two-sided test is calculated to evaluate whether the null hypothesis is true or false. In this case, the null hypothesis is that the mAP obtained by max and sum-pooling are equal. P-values larger than 0.05 indicate that the null hypothesis cannot be rejected.

Table 6.1 shows that there is no statistically significant difference between choosing one form of pooling over the other. When PCA whitening is used, sum pooling becomes significantly better than max-pooling.

Table 6.1: mAP comparison using max and sum-pooling over regions on the Oxford, Paris, and INSTRE datasets. Images processed at 340 pixels (larger size). The features are extracted from the pool5 layer of the VGG16 network. Results are reported with and without PCA-whitening post-processing.

	Oxford			Paris			INSTRE		
	Max	Sum	p-value	Max	Sum	p-value	Max	Sum	p-value
w/o PCAw	0.548	0.528	0.762	0.744	0.745	0.890	0.375	0.374	2.9e-22
w PCAw	0.551	0.591	0.001	0.750	0.773	0.016	0.274	0.279	0.008

Table 6.2: mAP comparison of R-MAC and R-SUM on Oxford, Paris, and INSTRE. Descriptors are post-processed with L^2 normalization and PCA whitening (L^2-norm + PCAw + L^2-norm). Images are processed at 1040 (larger size). The features are extracted from the pool5 layer of the VGG16 network.

Oxford		Paris		INSTRE	
R-MAC	R-SUM	R-MAC	R-SUM	R-MAC	R-SUM
0.710	0.704	0.834	0.803	0.352	0.296

We also report results with the default configuration of R-MAC in Table 6.2, testing both encodings (we identify R-MAC as R-SUM when sum-pooling is performed to build the region features). Even though this configuration was specifically optimized for max-pooling, R-SUM still provides similar performance in almost all the datasets tested.

In general, sum pooling is simpler to analyze than max pooling. As both provide similar performance, we focus on sum-pooling in the remainder of this chapter.

6.4 Region pooling as a weighting scheme

Let $\mathcal{X}$ be the tensor of activations of a convolutional layer, with dimension (H, W, D) where D is the number of convolutional filters, and (H, W) are the spatial dimension of the feature maps. Global sum-pooling the activations within that particular layer can be expressed as $F = \sum_{i=1}^{W} \sum_{i=1}^{H} f(i, j)$, where $f(i, j)$ is a vector in $\mathbb{R}^D$. Pooling over regions (using sum-pooling), followed by global sum-pooling of the region vector (R-SUM), can be expressed as:

$$F = \sum_{i=1}^{W} \sum_{i=1}^{H} \alpha(i, j) \odot f(i, j), \tag{6.3}$$

where the α is a D-dimensional vector that weights the contribution of a particular local convolutional feature $f(i, j)$, and $\odot$ is the elementwise multiplication. This weight is a non-parametric image-dependent spatial scheme composed of two factors: the first depends on the **window location** (how many times a particular local feature is considered for the final pooling) and the second on the **region post processing**

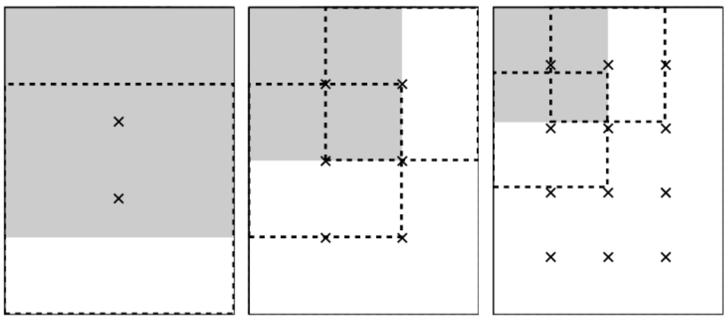


Figure 6.1: Sample regions extracted at 3 different scales ($l = 1...3$), source [19].

(typically L^2-normalization followed by PCA whitening and L^2-normalization again).

6.4.1 Window location factor

If we do not take into account the post-processing applied to each region, we can express the final global descriptors directly with Equation 6.3, where $\alpha(i, j)$ is the weighting factor that explicitly depends on the region proposal strategy applied.

R-MAC follows a multi-scale sliding window approach defined as follows. Let (H, W) be the dimensions of a feature map. The size of the window w is defined as $w = 2\min(H, W)/(l + 1)$, where $l = 1, \ldots, L$ defines the scale of the window. The stride of the window S is set to $S = 0.4w$ (forcing a 40% of overlap between windows). Figure 6.1 illustrates three different scales and window locations of the described R-MAC multi-scale sliding window strategy. Scale $l = 1$ contains only two sub-windows of square shape equal to the smaller dimension of the original image $w_1 = W/2$, $l = 2$ contains 6 windows with dimension $w_1 = 2W/3$, and $l = 3$ contains 12 sub-windows of $w_1 = 2W/4$.

Figure 6.2 shows the weight maps $\alpha(i, j)$ for descriptors from pool5 of VGG16. Working at full image resolution, the feature maps at that layer have dimension $(21, 32, 512)$. The number of windows per scale l considered depends on the resolution of the feature maps. For instance, when only one scale is considered ($L = 1$), only two square windows of $(21, 21)$ are generating, when $L = 2$, a total of 8 windows are generated: 6 windows of size $(14, 14)$ ($l = 2$), and 2 of $(21, 21)$ ($l = 1$). When $L = 3$, a total of 20 windows are generated, being 12 of size $(10, 10)$ and $l = 3$, and the rest from the larger resolutions. This sample strategy provides more importance to the central region of the image, independently of the image content.

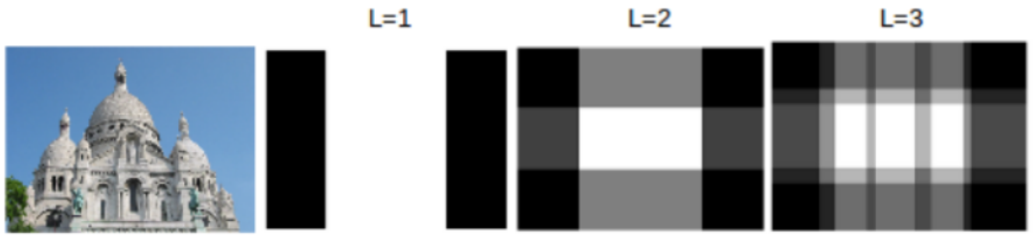


Figure 6.2: Weighted maps $\alpha(i, j)$ obtained from the sliding-window strategy applied in R-MAC at three different scales L.

6.4.2 Region post-processing factor

In R-MAC, all region pooling is followed by a post-processing scheme which includes L^2-normalization followed by PCA whitening followed by L^2-normalization again. For simplicity we just analyze the contribution of the first L^2-normalization. Pooling all the different region vectors after L^2-normalization can be expressed by:

$$F = \frac{\sum_{i=1}^{w}\sum_{j=1}^{w} f(i,j)}{\|\sum_{i=1}^{w}\sum_{j=1}^{w} f(i,j)\|^2} + \frac{\sum_{i=S}^{w+S}\sum_{j=1}^{w} f(i,j)}{\|\sum_{i=S}^{w+S}\sum_{j=1}^{w} f(i,j)\|^2} + \ldots + \frac{\sum_{i=mS}^{H}\sum_{j=nS}^{W} f(i,j)}{\|\sum_{i=mS}^{H}\sum_{j=nS}^{W} f(i,j)\|^2} \tag{6.4}$$

The regions considered are the same as the ones described in the previous subsection, but now each region contributes with a weight inversely proportional to its L^2-norm. Figure 6.3 provides the visualization of this weighting. In contrast to the window factor, this generates a weighting that is dependent on the image content that tends to upweight non-salient regions. This weighting seems to balance the contribution of the most active descriptors (those with higher L^2-norms) also taking into account how many times that a particular region is visited.

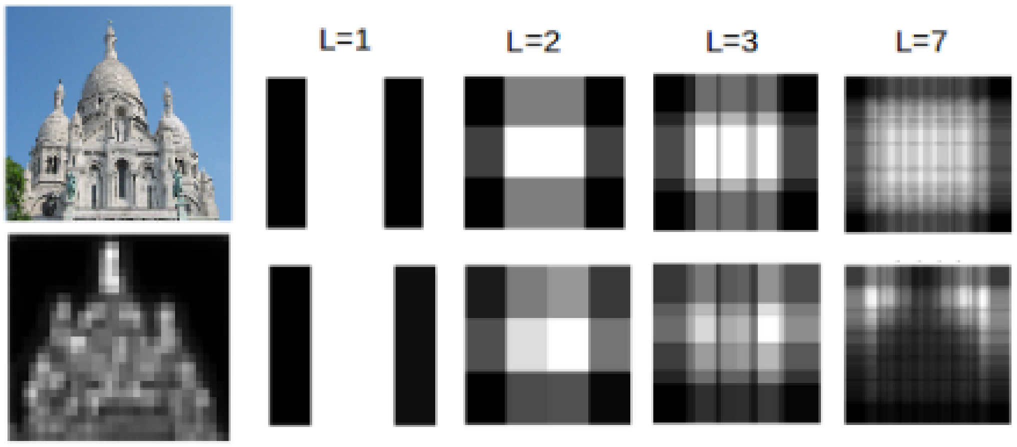


Figure 6.3: Weighted maps $\alpha(i, j)$ obtained from the sliding-window strategy applied in R-MAC (top row) and from the L^2-norms (bottom row) at four different scales L. On the left the original image and its L^2-norm map visualization. Weighting from post-processing factor with L^2 normalization appear to upweight non-salient regions.

6.4.3 Limitations of the fixed region sampling

Two factors contribute to the final weighting produced by aggregating descriptors from different regions: one depends on the considered locations and the other on the content of the images. As such we can think of R-MAC as a non-parametric weighting scheme, similar to CroW [**?**].

One of the main limitations of this approach is the fixed-grid used for sampling the regions. As illustrated in Figure 6.3, a fixed grid means more importance is placed on the centre of the image. This strategy, might be suitable in datasets such as Oxford and Paris where the main target instance is usually located in the centre [8]. However, it is not necessarily a good strategy in more challenging scenarios with more variability in terms of scale, location, or occlusion of the target instances. Such scenarios are more common in challenging benchmark datasets like INSTRE or TRECVID instance search.

Although the weighting obtained from the post-processing derives a component that depends on the image content, the contribution is directly linked to the window sampling. This weighting is somehow not intuitive, since it down-samples relevant parts of the image instead of boosting them.

In the approach proposed in this chapter, we discard the idea of individually processing different regions of the image. Instead, we globally apply an image-dependent spatial weighting scheme that allows us to control the influence of the relevant regions.

6.5 Object proposal weighting

One possible solution to overcome the limitation of using a fixed grid is to use an object proposal algorithm [45] to generate the candidate regions. Selective Search [127] is one of the most popular algorithms that merges superpixels based on engineered low-level features to generate a set of candidate regions.

More recently, Region Proposal Networks (RPN) [102] appeared as a data-driven object proposal method. RPN operate on top of a convolutional layer and is optimized to learn an "objectness" score for a particular region. At test time, the network predicts a set of N regions and corresponding bounding box coordinates that are more likely to contain an object. One of the main advantages of this network over algorithms like Selective Search is the possibility of optimizing the proposals for the final target task. As an example, Faster-RCNN [102] uses a RPN that shares parameters with the object detection network, so the model can be trained end-to-end for the final object detection task. In the scope of retrieval, Gordo *et al.* [39] optimized a RPN on landmarks images. Here, the RPN shares parameters with a retrieval network (trained using triplet loss and R-MAC encoding) as an alternative to using a fixed-grid to generate the proposals. As such the RPN is not only optimized to generate proposals, but also to learn those regions that are most informative for the final task, albeit with the drawback of requiring annotated data.

Following the same procedure described in Section 6.4.1, we use an object proposal algorithm to plot the contribution of each region instead of using a fixed

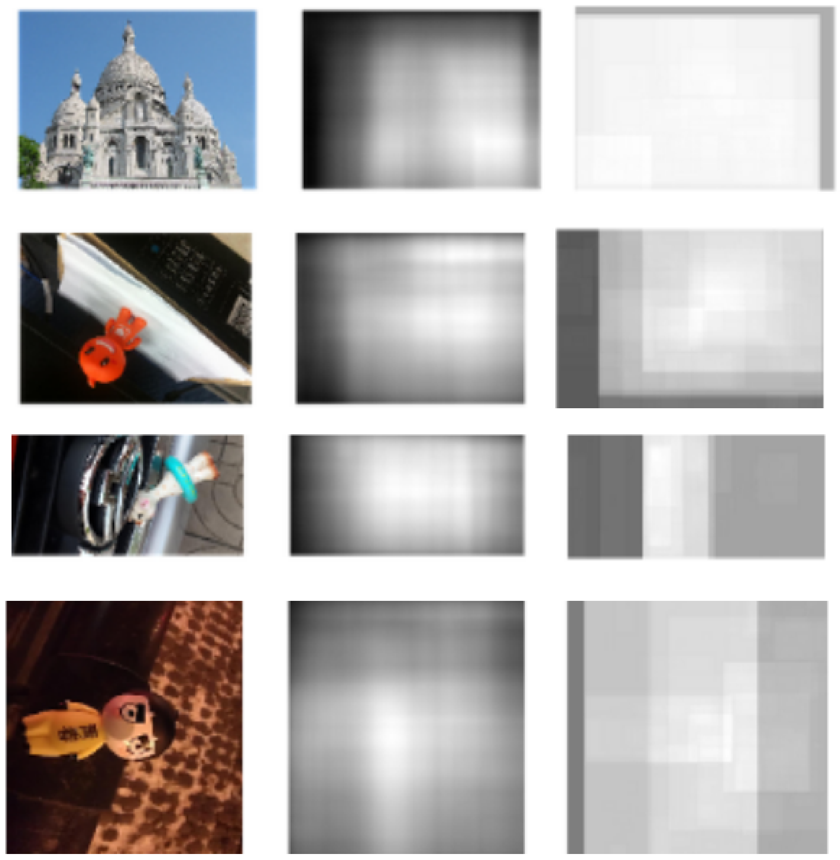


Figure 6.4: Weighted maps $\alpha(i,j)$ obtained from the object proposal algorithms. Each image is followed by the weights obtained by Faster R-CNN (middle) and Selective Search (right). Both methods provide a rough estimation of the relevant parts of the image.

grid. Figure 6.4 contains the weighting scheme obtained for the window location factor from Faster-RCNN trained on PASCAL VOC2007 (model and implementation from [35]) and from the Selective Search algorithm for four different images (the L^2 normalization factor is not included in the visualization). The RPN from Faster-RCNN generates 300 proposals (in the plot we represent up to 6000 regions corresponding to the bounding box regression of the 21 classes of PASCAL) while Selective Search results in 2000 proposals per image approximately. In contrast to the fixed grid approach, higher weights are applied to the most salient regions, obtaining a coarse localization of the most salient parts of the image.

Region proposal algorithms represent a smarter approach to pool region descriptors instead of using a fixed grid like in R-MAC. However, when local descriptors are sum-pooled into a single representation, the factor derived from the window sampling strategy can only provide a rough estimation of the relevant parts of the image. Motivated by this fact, in the next section we introduce saliency weighting as a more accurate image dependent weighting strategy that we will use to directly weight the contribution of the local convolutional features, instead of generating multiple regions.

6.6 Saliency weighting

One of the main features of the human vision system its capacity to actively focus on the most salient regions and movements. Visual saliency is a process that detects regions different from their surroundings, producing feature maps that contain the most prominent regions within an image.

Saliency prediction has been a problem traditionally addressed with hand-crafted features inspired by neurology studies. With the emergence of challenges such as the MIT saliency benchmark [16] and the Large-Scale Scene Understanding Challenge [143] (LSUN), and the appearance of large-scale public annotated datasets such as iSUN [136], SALICON [58], MIT300 [16], or CAT2000 [14], data-driven

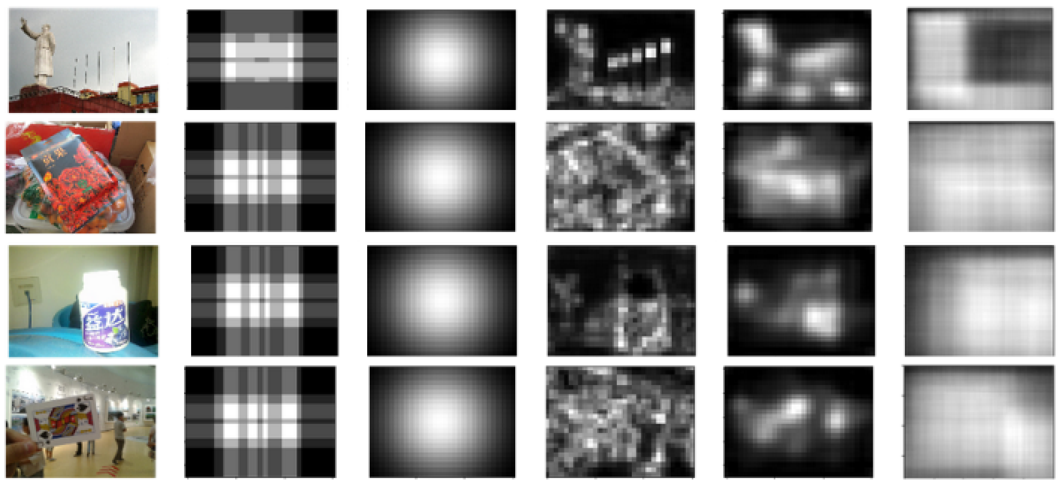


Figure 6.5: Weighted maps $\alpha(i, j)$ obtained for four images obtained by: fixed sliding window from R-MAC [125], Gaussian-centre prior [8], L^2-norms [61], saliency prediction from deep SalNet [94] and RPN [102].

approaches based on CNN models trained end-to-end have become the dominant approach to address this problem, generating more accurate models every year to predict human fixations [67, 58, 65, 93, 94].

While there has to date been a clear research focus on developing more accurate saliency models to improve performance on the MIT and LSUN benchmarks, less focus has been given to applying existing saliency models in other computer vision tasks. In particular, state-of-the-art saliency models could be applied to weight the contribution of local convolutional features as an alternative of using a fixed-grid region weighting as used in R-MAC. Figure 6.5 shows different weighting approaches for local CNN features. R-MAC fixed-grid and centre prior only depend on the image resolution, which it is a limitation in retrieval scenarios where the target instances are not centered located. L^2-norms, saliency, and RPN depend on the image content. However, saliency weighting provides a more accurate location of the relevant parts of the images. Instead, L^2-norm generates a noisy saliency map, when the background is highlighted in most of the cases, and RPN generates a low resolution saliency of the relevant objects due to that the algorithm generates thousands of bounding box instead of accurate pixel level saliency.

6.7 Experimental setup

In this subsection we describe the different spatial weighting schemes considered for the instance search task. We consider two aggregation methods: global sum-pooling and BoW of the weighted convolutional features. The method is evaluated in general retrieval benchmarks (Oxford and Paris) as well as in the challenging instance search benchmark INSTRE.

6.7.1 Deep Salnet

We generate saliency maps using the *Deep Salnet* network proposed in [94]. This network is a fully convolutional network consisting of ten layers: one input layer, eight convolution and one "deconvolution" layer [140] [1]. The ReLU is used as an activation function and pooling layers follow the first two convolutional layers, effectively reducing the width and height of the feature maps in the intermediate layers by a factor of four. The final deconvolution generates the final prediction, up-sampling the resolution to the original input size. A transfer learning strategy is applied on this network by re-using and adapting the weights from the three convolutional layers from a pre-trained VGG architecture [113]. This acts as a regularizer and improves the final network result. The loss function used during training is the mean squared error computed pixel-wise between the saliency prediction and the ground truth map.

6.7.2 Weighted sum-pooling

We use the pre-trained VGG16 model for image classification [113] for extracting the local features. The images are processed at full resolution (1024 for the larger dimension in Oxford and Paris and 1000 for INSTRE) and we encode *pool5* features as they are the best performing features for direct sum-pooling aggregation [61].

[1]Also called "transposed convolutions". The purpose of this layer is to go in the opposite direction to convolution (i.e if a convolution compresses a 3×3 activation window in a single neuron, a deconvolution generates a 3×3 neurons from a single activation). It can be seen as an up-sampling layer with learnable parameters.

Images are pre-processed with mean subtraction prior to being forwarded to the network. We apply five different spatial weighting schemes following Equation 6.3: fixed R-MAC, Gaussian centre prior, L^2-norm, saliency, and RPN (Figure 6.5). All weights are normalized to have scale between 0 and 1. All weighted descriptors are aggregated via sum-pooling into a compact descriptor of 512 dimensions. Standard post-processing of the features is performed: L^2 normalization, PCA whitening (512 dimension), and a second L^2 normalization. We follow the standard procedure in [125, 8**?**] and fit the PCA transform on the Paris dataset when testing on the Oxford dataset and on the Oxford dataset when testing on Paris. In INSTRE, we fit the PCA transform on the same dataset, since the image domain in Oxford and Paris is significantly different. Mean Average Precision (mAP) is used for all the datasets as the evaluation metric.

6.7.3 Weighted BLCF

We investigate applying the proposed five different weighting schemes to the visual words in the BLCF scheme. In this case, images are resized to have a maximum size of 340 pixels (1/3 of the original resolution) before performing mean subtraction prior to being forwarded to the pre-trained VGG16 network. Features from *conv5_1* are L^2-normalized, PCA whitened (512D), and L^2-normalized again before being clustered. Clustering is performed using k-means with $k = 25000$ words, following the pipeline described in Section 4.2. As with sum pooling, the PCA transform and visual vocabulary in INSTRE are fit on the the same dataset. Images are not interpolated, meaning that the maximum number of words required to store is equal to the number of local features ($\approx$340).

6.7.4 Query Processing

The queries are processed in two ways: using the bounding box of the object (*Local*), and pooling all the features (*Global*). For the *Local* approach, we map the coordinates of the bounding box on the *pool5* layer (in the case of direct sum-pooling) and in the assignment map (in the case of BLCF) and encode the features/visual words contained within the region. In the case of BLCF, features are interpolated to generate larger convolutional maps (as in Chapter 4).

6.8 Quantitative results

We present the results testing different weighting approaches with direct sum-pooling in Table 6.3 and Bags of Convolutional features in Table 6.4. Results show that:

Table 6.3: Evaluation of different weighting schemes on direct sum-pooling.

	Oxford		Paris		INSTRE	
Weighting	**Global**	**Local**	**Global**	**Local**	**Global**	**Local**
None	0.680	0.686	0.779	0.765	0.261	0.406
R-MACw	0.683	**0.691**	0.794	0.787	0.297	0.459
Gaussian	**0.684**	0.688	**0.795**	**0.793**	0.318	0.484
L^2-norm	0.671	0.676	0.782	0.782	0.321	0.466
Saliency	0.668	0.681	0.776	0.778	**0.352**	**0.519**

Table 6.4: Evaluation of different weighting schemes on direct BLCF.

	Oxford		Paris		INSTRE	
Weighting	**Global**	**Local**	**Global**	**Local**	**Global**	**Local**
None	0.628	0.722	0.642	0.798	0.350	0.526
R-MACw	0.650	0.731	0.677	0.806	0.403	0.568
Gaussian	0.666	0.728	0.701	0.809	0.447	0.591
L^2-norm	0.666	0.740	0.711	**0.817**	0.468	0.612
Saliency	**0.670**	**0.746**	**0.726**	0.814	**0.547**	**0.654**

- Centre-bias weights (R-MACw, Gaussian) are in general beneficial in all datasets, which suggests that objects tend to be located in the center in all cases. Although, in Oxford and Paris the improvement with respect to applying no weighting is again less significant than in the INSTRE dataset.
- BLCF aggregation specially benefits from content-dependent weighting schemes. Encoding local features into a high dimensional sparse space helps to ensure that the information is more likely to be linearly separable in comparison with

dense pooling, as shown in Chapter 4. In this scenario, fine-grained detail provided by saliency is shown to be more beneficial than center-bias approaches.

- Performance on the INSTRE dataset particularly benefits from the content-dependent approaches. Specifically, saliency weighting increases performance 0.1 mAP points in the case of *Local* search in sum-pooled features. In the case of BLCF, the increase is nearly 0.13 mAP points.
- In comparison with sum-pooling, BLCF produces a high-dimensional and sparse representation that benefits more from the different weightings. It is notable to remark that even though the BoW dimension is 25k, it is very sparse: only a maximum of 320 dimensions (corresponding to the total local features encoded) need to be stored. This makes this approach even more computationally and memory efficient that direct pooling, which results in dense 512D vectors.

6.9 Comparison with the state-of-the-art

In this section we compare our best performing configuration (BLCF with saliency weighting) with state-of-the art approaches on the Oxford, Paris, and INSTRE datasets. Results are compared with R-MAC, which can be re-interpreted as another weighting version of local convolutional features, and its fine-tuned version for image retrieval [100, 39]. We also include results with average query expansion (QE) [3], where we take the top 10 (as in [61, 125]) retrieved images and generate a new query vector by aggregating their assignments with the ones belonging to the original query. A new ranked list is created by issuing this new query.

We observe that in comparison with R-MAC our approach performs similarly in the Oxford and Paris datasets in, but clearly outperforms R-MAC in the INSTRE dataset. The weighting scheme obtained by the R-MAC approach is restricted by the fixed window locations (centre-bias) and the post processing applied to the features (image dependent factor). While this strategy generates good results in domain specific datasets such as Oxford and Paris, its performance is limited in the instance

Table 6.5: Comparison of saliency weighting BLCF and state-of-the-art on Oxford, Paris and INSTRE. In parenthesis we include the results when combining BLCF with SIFT-BoW rankings (Chapter 5).

	Local Search			Local Search+QE		
	Oxford	Paris	INSTRE	Oxford	Paris	INSTRE
ours	0.746	0.814	0.654	0.795(0.904)	0.843(0.914)	**0.737(0.739)**
R-MAC [125]	0.669	0.830	0.352*	-	-	-
(f)R-MAC [100]	0.770	0.841	0.470	0.854†	0.884†	0.573†
(f)ResNet101 [39]	**0.861**	**0.945**	**0.626**†	**0.896**†	**0.953**†	0.705†

* Result generated with our own implementation.
† Results from Iscen *et al.* [48].

search datasets such INSTRE, where saliency weighting is a better approach.

Regarding the fine-tuned versions (f)R-MAC [100] and (f)R-MAC ResNet101 [39], results indicate that fine-tuning for retrieval (even when using landmark images for training) is always beneficial. However, this approach has the extra cost of finding the adequate training data with its ground truth annotations. Our method exploits off-the-shelf models from image classification and saliency prediction, which generalize well in different datasets and achieves similar or superior performance to models fine-tuned on landmarks.

As seen in the previous chapter, BLCF and Saliency weighting can be applied to the finetuned features. This could potentially improve the performance as well as the efficiency of the obtained CNN representations.

6.10 Qualitative results

Instance search aims to locate general and diverse instances within a dataset of images. Oxford and Paris datasets have the limitation of being restricted to the particular domain of landmark buildings. In this domain, the original unweighted CNN features or a simple centre-prior weighting may suffice, since instances tend to occupy a large part of the image and usually the background contains useful information. We found that queries related to landmark buildings in INSTRE dataset tend to generate higher average precision (AP) using this strategy. For instance,

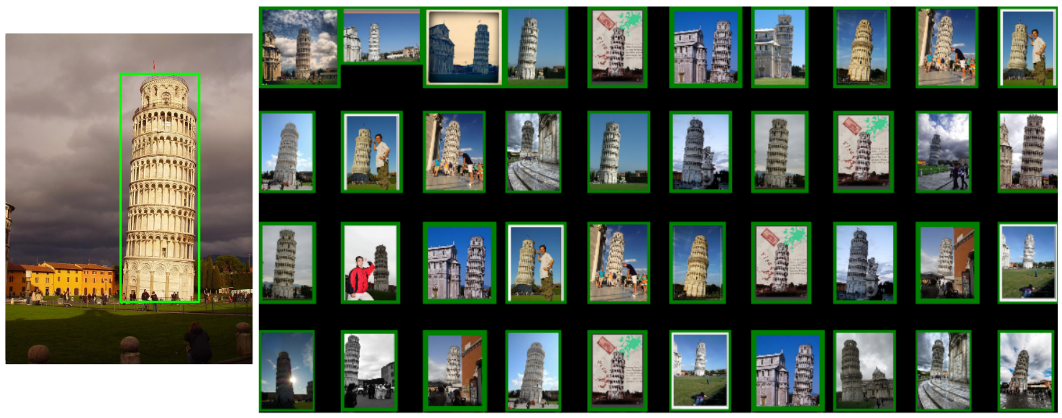


Figure 6.6: The top-10 ranked results for query 766 of the INSTRE dataset. Sum-pooling aggregation is used as feature encoding. The first row contains of unweighted features (AP = 0.805); the second, Gaussian weighted features (AP = 0.774); the third, L^2-norm weighted features (AP = 0.806); and last saliency weighted features (AP = 0.744).

Figure 6.6 contains ranked results for a building query, where relevant images contain the query building centred and usually with a similar size.

Saliency weighting turns out to be particularly beneficial for instances like faces, logos, and small objects. As an example, Figure 6.7 contains the top-10 ranked images for a picture of a face. Non-content dependent weightings perform poorly because of the high diversity of locations and backgrounds where the instance can appear. Saliency, which provides maps biased to boost the importance of regions containing human faces, since it is trained with real human fixations, significantly improves performance over all other approaches. L^2-norm weights, in contrast, result in much noisier maps that are less informative in finding the target instance.

Figure 6.8 contains another example where the query is a small salient object. Gaussian weighting improves with respect the baseline but it still retrieves non-relevant results with similar properties (a salient object centred in the middle of the image). L^2-norms generate noisy saliency masks such that results are still dependent on the background. In this case, saliency weights clearly locate the relevant instance and we retrieve results of the instance located in a much larger diversity of backgrounds.

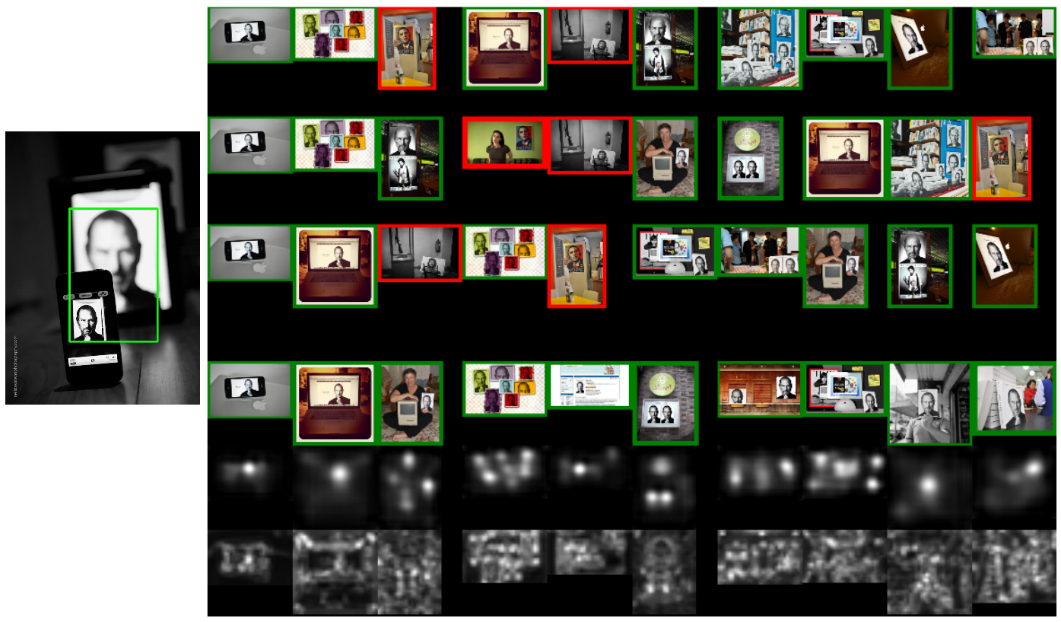


Figure 6.7: Top-10 ranked results query 722 of the INSTRE dataset. Sum-pooling aggregation is used as the feature encoding. The first row contains for the unweighted features (AP = 0.152). The second row, Gaussian weighted features (AP = 0.184). The third row L^2-norm weighted features (AP = 0.110) and last row saliency weighted features (AP = 0.353). Additionally we add visualization of saliency masks (row 5) and L^2-norm masks (row 6) related to the saliency results.

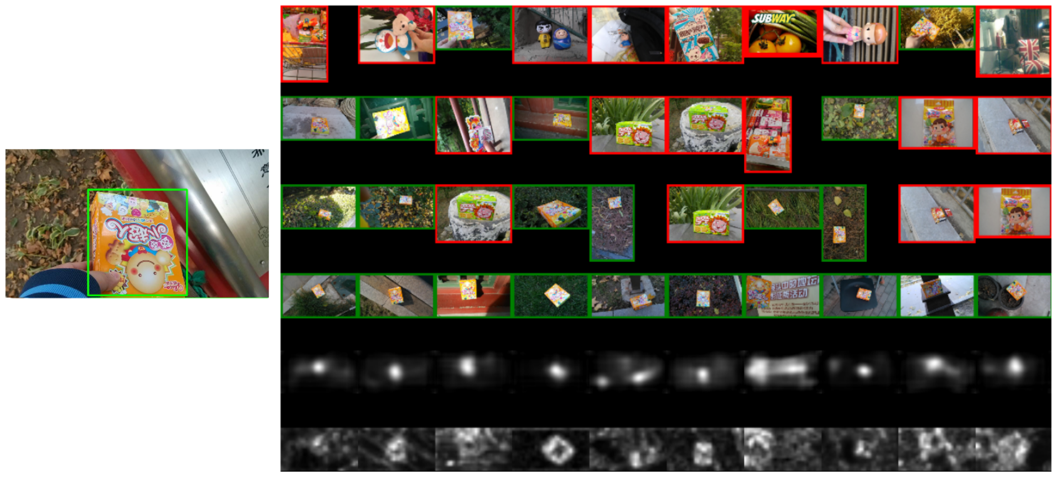


Figure 6.8: Top-10 ranked results query 842 of the INSTRE dataset. Sum-pooling aggregation is used as the feature encoding. The first row contains results for unweighted features (AP = 0.332). The second row, Gaussian weighted features (AP=0.600). The third row L^2-norm weighted features (AP = 0.654) and last row saliency weighted features (AP = 0.811). Additionally we add visualization of saliency masks (row 5) and L^2-norm masks (row 6) related to the saliency results.

6.11 Discussion

Saliency models substantially improve the performance of CNN features, especially in BLCF encoding. Visual saliency modeling is an active research topic where every year more sophisticated models are proposed improving the state-of-the in benchmarks such as the MIT saliency benchmark [16] or the Large-Scale Scene Understanding [143] benchmark. Future work investigate whether more accurate saliency models will improve out retrieval system.

We also evaluated the performance on the TRECVID subset. We observe that saliency weighting improves results over centre-prior weighting but does not improve over the baseline of applying no weighting. We speculate that since the images are keyframes extracted from videos, the nature of the data is substantially different from a collection of pictures. In an image collection the object of interest tends to be centred. Even in the very diverse INSTRE dataset, center prior weighting schemes are shown to be beneficial.

In the case of TRECVID, some of the instances are objects that always tend to appear in the same room. Taking into account the background for those instances is very beneficial. However, other instances related to people or logos tend to appear in different scenarios. Figure 6.9 contains visual examples of this behaviour. With this motivation, future work will explore adapting the saliency weighting approach to allow it to take into account more or less background depending on the query instance. This could be used as a re-ranking approach that would increase the query time but that could represent a solution for achieving accurate rankings in these kinds of scenarios.

6.12 Summary

In this chapter we have provided another view of the popular R-MAC encoding. We have re-interpreted region pooling as a non-parametric image dependent weighting scheme that relies on two factors: the window sampling strategy and the post-processing applied to each region. This approach has the limitation of working with fixed grid of locations that provides a centre-bias to the final weighting. The post-processing step adds weights directly related to the image content. However, this weighting is directly affected by the window sampling strategy.

We have explored different weighting alternatives such as using the spatial weights obtained from region proposal algorithms such as a substitution of the fixed grid utilized in R-MAC. Those algorithms, en essence, provide a rough localization of the relevant parts of the image, a fact that lead us to directly consider state-of-the-art saliency models to weight the contribution of local convolutional features prior the aggregation step.

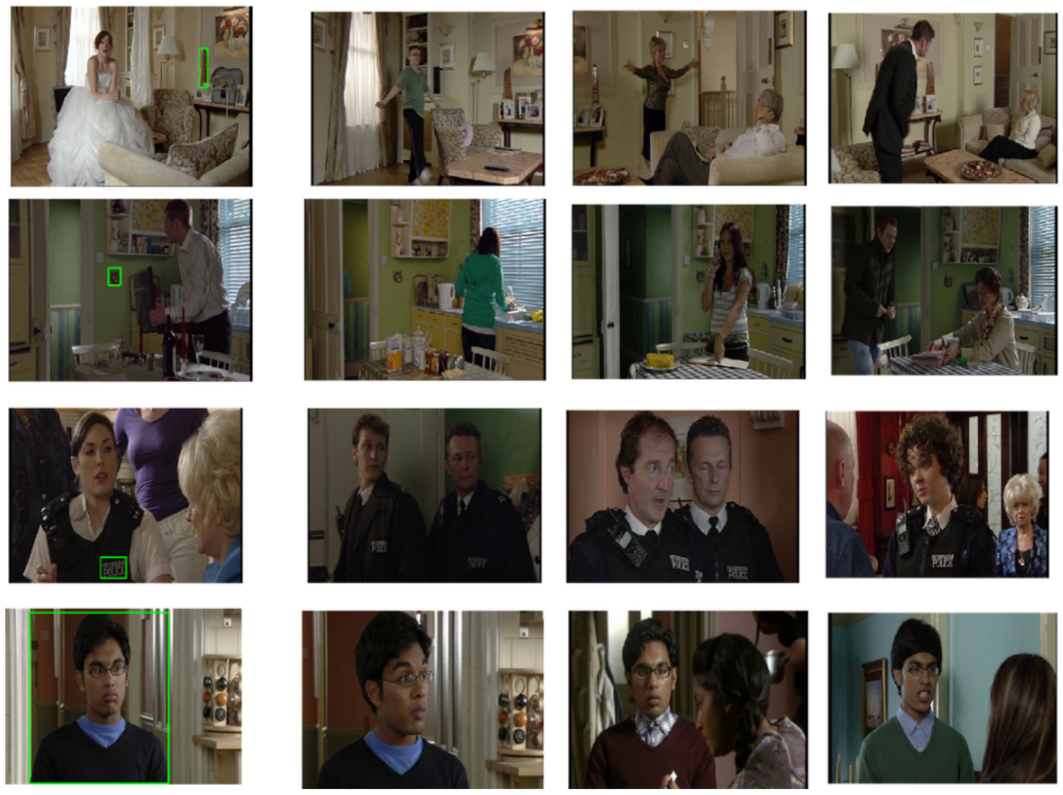


Figure 6.9: Examples of query instances of TRECVID. The first column contains the query instance, the following three rows contain visual examples of relevant images. Most of the instances in TRECVID are highly related to its context (first two rows). Instances where using the query bounding box is beneficial are logos or people, which are more likely to appear in different scenarios.

We have presented a quantitative evaluation of different weighting schemes: non-image dependent schemes such as the centre-bias Gaussian weighting, non-parametric image dependent schemes such as L^2-norms of convolutional features and parametric image dependent schemes such as saliency weighting. We target two well known retrieval benchmarks, Oxford and Paris, and a specific instance search designed dataset: INSTRE.

Results indicate that saliency weighting outperforms other weighting schemes such as R-MAC in INSTRE. Moreover, off-the-shelf convolutional features with our proposed BLCF encoding outperform even the state-of-the art fine-tuned version of R-MAC, with more efficient representations (only non zero elements are stored, which leads to storing a maximum of 300 integer numbers as opposed to the 2048 float numbers required to store the dense representation from the fine-tuned R-MAC).

6.13 Conclusions

We have proposed a method for instance search that relies on a BoW encoding of local convolutional features. For this, we have used a pre-trained off-the-shelf classification model and a state-of-the art saliency model. Visual words derived from the convolutional features are weighted with saliency weights providing a scalable and general instance search pipeline. Our framework achieves state-of-the-art results on the challenging INSTRE dataset, outperforming methods that have been specifically trained for retrieval and furthermore we provide a rationalization for the generalization of the proposed pipeline to generic instance retrieval.

As shown in the previous chapter, the performance of the proposed system could benefit even further from the complementary information of SIFT-BoW pipelines. When we combine both systems, we achieve state-of-the-art results at the cost of increasing the processing time and memory requirements. However, the proposed pipeline does not over-fit any particular domain, and it is demonstrated to be a good generic instance search solution without requiring fine-tuning for similarity learning.

Chapter 7

Conclusions

The traditional bag-of-visual-words (BoW) encoding has been widely used to aggregate hand-crafted local descriptors such as SIFT. Typically BoW represents an initial filtering stage followed by a spatial analysis step to improve the retrieval results at the cost of increasing query time.

In this thesis, we have studied the adaptability of BoW to local CNN representations derived from off-the-shelf classification networks (BLCF) achieving very competitive performance with respect to other CNN off-the-shelf methods. Similar to traditional approaches, the achieved performance can be further improved by including a spatial verification step, albeit compromising the query time.

We have investigated two methods for generating better CNN representations as an alternative to expensive re-ranking strategies. The first is a unsupervised fine-tuning strategy, which improves the underlying CNN representation at the cost of losing the generalizability of the CNN representations. The second explores the usage of attention mechanisms, in particular saliency models, that significantly improve the performance of CNN representations. We discovered an efficient representation that achieves very competitive performance across different retrieval benchmarks. Spatial re-ranking and query-adaptive methods could further improve the performance of the proposed system at the cost of increasing the query time.

In this chapter, Section 7.1 summarizes the main contributions of this work.

Section 7.2 reviews the hypotheses and research questions introduced in Chapter 1 based on the knowledge gained during this research. Section 7.3 concludes the work with recommendations and directions for future research.

7.1 Research contributions

The key contributions of this thesis are the following:

- We conduct an investigation of the traditional bag of visual words encoding on local CNN features to produce a scalable image retrieval framework that generalizes well across different retrieval domains.
- We explore a fine-tuning strategy for similarity learning without requiring additional annotated labels to improve descriptors of a pre-trained network for a particular retrieval benchmark. The obtained results suggest that the performance improvement is at the cost of losing generalization of the representations on different kinds of instances.
- We propose a general solution that achieves state-of-the-art performance in different benchmarks by exploring CNN saliency models, which have the advantage of not requiring additional training data or over-fitting to a particular image domain.

7.2 Hypotheses and research questions

In this section we review the hypotheses and research questions of this work (stated in Section 1.4) based on the investigations conducted:

H1) **When building a global image representation from local CNN image descriptors for retrieval, the aggregation of those descriptors into a high-dimensional sparse representations is "better" (in terms of performance and efficiency of the representation) than aggregating them within the original local feature space.**

Experiments conducted in Chaper 4 suggest that the high dimensionality of BCLF allows us to better preserve the local information within the images. Using the visual words corresponding to the instance within the query image produces a boost in performance over direct encoding of all the visual content within the image. Tables 4.3, 4.4 and 6.4 show this effect, where Local Search (LS) clearly outperforms Global Search (GS), with the exception of the TRECVID dataset in Table 4.5, due to the fact that some of the instances are highly related to the background. One benefit of better preserving the local information is it allows better advantage of more sophisticated spatial weighting schemes. In Section 6.8 we observe how BLCF achieves more competitive performance than direct pooling strategies [8**?**] after applying different weighting strategies. Despite the high dimensionality of BLCF, the aggregation method represents the image in a maximum of 300 non-zero elements as indicated in Table 4.7, which leads in a very efficient representation of the image content.

With the experiments conducted we can conclude that it is indeed possible to use traditional BoW encoding on off-the-shelf local representations to address the task of image retireval (*H1-Q1*).

The method achieves competitive performance compared to other state-of-the-art methods in popular retrieval datasets such as the Oxford and Paris datasets (in terms of quality of the generated rankings). BLCF outperforms other off-the-shelf CNN methods [125, 8**?**] in the more challenging TRECVID and INSTRE datasets, while requiring less memory to store the image representations (*H1-Q2*).

Finally, we showed how the same method can be successfully applied to diverse datasets without requiring training data, yielding good results and showing the generalization of the BLCF (*H1-Q3*).

H2) **Re-training a pre-trained CNN for the task of instance retrieval is an**

appropriate procedure to build specialized but not generic instance search systems.

With the experiments conducted on the Oxford, Paris, and INSTRE datasets in Chapter 5, we can conclude that limiting the training data to these datasets "can" lead to a CNN network highly over-fit to the training domain. This suggests that other methods such as [131, 106] achieve a performance improvement at the cost of losing generalization in the representations. Our method, however, considers an unsupervised approach to generate training samples exploiting a combined SIFT-CNN system. Recent works [39, 100] suggest that the generation of a much large training may prevent overfitting. This improvement is at the cost of requiring the construction of a suitable training set, which is computationally demanding. For this reason, we cannot conclude that "in general" re-training a pre-trained CNN for the task of instance retrieval always yields a specialized instance search systems, because the performance of the obtained CNN model depends heavily on the training data used.

Based on the experiments performed, we can conclude that it is indeed possible to perform similarity learning in a particular dataset without requiring manual annotations (*H2-Q4*) in the case of the Oxford/Paris datasets. However, in the INSTRE dataset, we found it necessary to use classification labels to achieve a good performing model due to the high diversity of the instances within the dataset.

Limiting the training data to those datasets results in very poor generalization of the CNN models (*H2-Q5*).

H3) **Visual attention models are useful for the task of generic instance retrieval**.

Experiments performed in Chapter 6 indicate how saliency models can be used to improve performance of off-the-shelf CNN representations (*H3-Q6*). Visual attention models help to construct a more powerful representation for instance retrieval, either using direct pooling strategies such as sum-pooling or the proposed BLCF framework (Section 6.8). In the case of BLCF, the proposed saliency weighting scheme represents an alternative for efficient retrieval without having to apply expensive re-ranking stages, where individual regions are encoded and compared to a query. However, spatial re-ranking (Section 4.4.4) can be complementary to an initial search based on saliency weighting, possibly boosting the achieved performance at the cost of increasing the query time.

7.3 Recommendations and future work

While most related work has focused on comparing the performance of CNN descriptors with traditional methods based on hand-crafted local features, our results provide evidence that both representations can be complementary. Future work could explore more sophisticated strategies to fuse both representations.

Fine-tuning a CNN model for instance retrieval is still a challenging task. One challenging factor is collecting a suitable training set, that is generally domain specific, since the notion of similarity is application dependent. Future work will explore larger and more suitable datasets for similarity learning [1, 100, 39] to include BLCF aggregation as a layer of a CNN network.

Also, while most of the discussed state-of-the-art used the VGG16 architecture, recent works show how deeper CNN architectures, such as ResNet101, generate image representations that achieve better performance. Future research will investigate how BLCF and attention models adapt to these deeper models.

Attention mechanisms based on saliency prediction were found to be useful in weighting different parts of an image during the encoding of an image representation for instance retrieval. It would be very interesting to see how more sophisticated saliency models perform in the instance search task. Whilst there has been much research focused on developing better saliency models, it is unclear how crucial high accuracy is in practical applications such as instance retrieval. Future research will investigate this further.

Finally, a scalable query-adaptive approach can be further investigated. Methods that customize the spatial weighting schemes to minimize the distance between query and target image, as in [17], or methods exploiting the image manifolds within the feature space using diffusion mechanisms to propagate the similarity with a particular query [48], demonstrate a boost in performance. In particular, it would be very interesting to investigate a query-adaptive approach based on saliency weighting, where it is possible to weight the contribution of the background depending on the query. In the case of the TRECVID dataset, where some instances present high dependency with the background and some do not, this method could allow a significant performance improvement.

7.4 Closing remarks

Content-based image retrieval, like many other computer vision tasks, has been transformed by deep learning methods in recent years. Deep representations have been proven to be very effective for the task of instance search, especially when a suitable training dataset is available [39, 100].

In this thesis we showed how deep representations can benefit from traditional techniques. We have seen that SIFT and CNN representations contain complementary information that can be combined to produce more effective representations. In practical scenarios where memory is not a restriction, SIFT and CNN retrieval systems can be combined to generate more accurate ranked lists. Here we only investigated a very simple SIFT/CNN fusion scheme; more sophisticated approaches have the potential to provide further improvements in future.

In this vein, we have seen that learning from "what was done before" (pre-deep learning era) can be very beneficial in image retrieval but also in other computer vision tasks. In particular, the usage of the traditional bag of visual words encoding

(BoW) allowed us to generate more efficient and accurate retrieval systems than other state-of-the-art approaches. The encoding does not require any additional training data and can be applied to any CNN model. In future we plan to investigate how this method performs in more powerful models than VGG16, such as ResNet, as well how to formulate BoW as a differentiable layer for a CNN.

One exciting finding of this work has been the usage of attention models for the instance search task. During recent years, with the creation of large annotated datasets, much research has focused on developing more accurate saliency models, while less attention has been given to practical usages of state-of-the-art saliency models. We believe that more focus should be given to study how image retrieval and other computer tasks can benefit from state-of-the-art visual attention models.

Bibliography

[1] Arandjelović, R., Gronat, P., Torii, A., Pajdla, T., and Sivic, J. (2016). NetVLAD: CNN architecture for weakly supervised place recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[2] Arandjelović, R. and Zisserman, A. (2011). Smooth object retrieval using a bag of boundaries. In *IEEE International Conference on Computer Vision*.

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

[4] Arandjelovic, R. and Zisserman, A. (2013). All about VLAD. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '13, pages 1578–1585, Washington, DC, USA. IEEE Computer Society.

[5] Arthur, D. and Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics.

[6] Awad, G., Kraaij, W., Over, P., and Satoh, S. (2017). Instance search retrospective with focus on trecvid. *International Journal of Multimedia Information Retrieval*, 6(1):1–29.

[7] Ba, J., Mnih, V., and Kavukcuoglu, K. (2014). Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*.

[8] Babenko, A. and Lempitsky, V. (2015). Aggregating local deep features for image retrieval. In *Proceedings of the IEEE international conference on computer vision*, pages 1269–1277.

[9] Babenko, A., Slesarev, A., Chigorin, A., and Lempitsky, V. (2014). Neural codes for image retrieval. In *Computer Vision–ECCV 2014*, pages 584–599.

[10] Beis, J. and Lowe, D. (1997). Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 1000–1006. IEEE.

[11] Bell, S. and Bala, K. (2015). Learning visual similarity for product design with convolutional neural networks. *ACM Transactions on Graphics (TOG)*, 34(4):98.

[12] Bengio, Y. and Sénécal, J.-S. (2003). Quick training of probabilistic neural nets by importance sampling. In *Proceedings of the conference on Artificial Intelligence and Statistics (AISTATS)*.

[13] Borji, A. and Itti, L. (2013). State-of-the-art in visual attention modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):185–207.

[14] Borji, A. and Itti, L. (2015). Cat2000: A large scale fixation dataset for boosting saliency research. *arXiv preprint arXiv:1505.03581*.

[15] Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. (1994). Signature verification using a" siamese" time delay neural network. In *Advances in Neural Information Processing Systems*, pages 737–744.

[16] Bylinskii, Z., Judd, T., Borji, A., Itti, L., Durand, F., Oliva, A., and Torralba, A. MIT saliency benchmark.

[17] Cao, J., Liu, L., Wang, P., Huang, Z., Shen, C., and Shen, H. T. (2016). Where to focus: Query adaptive matching for instance retrieval using convolutional feature maps. *arXiv preprint arXiv:1606.06811*.

[18] Carlevaris-Bianco, N. and Eustice, R. M. (2014). Learning visual feature descriptors for dynamic lighting conditions. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 2769–2776. IEEE.

[19] Chang, N.-S. and Fu, K. S. (1980). A relational database system for images. In *Pictorial Information Systems*, pages 288–321.

[20] Chang, S. K. and Hsu, A. (1992). Image information systems: where do we go from here? *IEEE Transactions on Knowledge and Data Engineering*, 4(5):431–442.

[21] Chang, S. K., Yan, C. W., Dimitroff, D. C., and Arndt, T. (1988). An intelligent image database system. *IEEE Transactions on Software Engineering*, 14(5):681–688.

[22] Chatfield, K., Lempitsky, V. S., Vedaldi, A., and Zisserman, A. (2011). The devil is in the details: an evaluation of recent feature encoding methods. In *British Machine Vision Conference, BMVC 2011, Dundee, UK, August 29 - September 2, 2011. Proceedings*, pages 1–12.

[23] Cheng, E., Xie, N., Ling, H., Bakic, P., Maidment, A., and Megalooikonomou, V. (2010). Mammographic image classification using histogram intersection. In *Biomedical Imaging: From Nano to Macro, 2010 IEEE International Symposium on*, pages 197–200. IEEE.

[24] Chopra, S., Hadsell, R., and LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546 vol. 1.

[25] Chum, O. and Matas, J. (2010). Large-scale discovery of spatially related images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(2):371–377.

[26] Chum, O., Mikulk, A., Perdoch, M., and Matas, J. (2011). Total recall II: Query expansion revisited. In *CVPR 2011*, pages 889–896.

[27] Chum, O., Philbin, J., Sivic, J., Isard, M., and Zisserman, A. (2007). Total recall: Automatic query expansion with a generative feature model for object retrieval. In *IEEE International Conference on Computer Vision*, Rio de Janeiro, Brazil.

[28] Delhumeau, J., Gosselin, P., Jégou, H., and Pérez, P. (2013). Revisiting the vlad image representation. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 653–656. ACM.

[29] Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255.

[30] Deng, L. and Yu, D. (2014). Deep learning: Methods and applications. *Foundation and Trends in Signal Processing*, 7:197–387.

[31] Donahue, J., Hendricks, L. A., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., and Darrell, T. (2017). Long-term recurrent convolutional networks for visual recognition and description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):677–691.

[32] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages I–647–I–655. JMLR.org.

[33] Fischler, M. A. and Bolles, R. C. (1987). Readings in computer vision: Issues, problems, principles, and paradigms. chapter Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated

Cartography, pages 726–740. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[34] Friedman, J. H., Bentley, J. L., and Finkel, R. A. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software.*

[35] Fu-Hsiang Chan, P. (2017). Faster-RCNN_TF. `https://github.com/smallcorgi/Faster-RCNN_TF`.

[36] Gemert, J. C., Geusebroek, J., Veenman, C. J., and Smeulders, A. W. (2008). Kernel codebooks for scene categorization. In *Proceedings of the 10th European Conference on Computer Vision: Part III*, ECCV '08, pages 696–709, Berlin, Heidelberg. Springer-Verlag.

[37] Girshick, R. (2015). Fast R-CNN. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 1440–1448, Washington, DC, USA. IEEE Computer Society.

[38] Gong, Y., Wang, L., Guo, R., and Lazebnik, S. (2014). Multi-scale orderless pooling of deep convolutional activation features. In *European conference on computer vision*, pages 392–407. Springer.

[39] Gordo, A., Almazán, J., Revaud, J., and Larlus, D. (2017). End-to-end learning of deep visual representations for image retrieval. *International Journal of Computer Vision*, 124(2):237–254.

[40] Grave, É., Joulin, A., Cissé, M., Grangier, D., and Jégou, H. (2016). Efficient softmax approximation for gpus. *arXiv preprint arXiv:1609.04309.*

[41] Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742.

[42] Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Manchester, UK.

[43] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

[44] Hoffer, E. and Ailon, N. (2015). Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer.

[45] Hosang, J., Benenson, R., and Schiele, B. (2014). How good are detection proposals, really? In *Proceedings of the British Machine Vision Conference*. BMVA Press.

[46] Huang, G., Zhu, Q., and Siew, C. (2006). Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1):489 – 501. Neural Networks.

[47] Huiskes, M. J., Thomee, B., and Lew, M. S. (2010). New trends and ideas in visual concept detection: The mir flickr retrieval evaluation initiative. In *MIR '10: Proceedings of the 2010 ACM International Conference on Multimedia Information Retrieval*, pages 527–536, New York, NY, USA. ACM.

[48] Iscen, A., Tolias, G., Avrithis, Y., Furon, T., and Chum, O. (2016). Efficient diffusion on region manifolds: Recovering small objects with compact cnn representations. *arXiv preprint arXiv:1611.05113*.

[49] Jaakkola, T. S. and Haussler, D. (1999). Exploiting generative models in discriminative classifiers. In *Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II*, pages 487–493, Cambridge, MA, USA. MIT Press.

[50] Jégou, H. and Chum, O. (2012). *Negative Evidences and Co-occurences in Image*

Retrieval: The Benefit of PCA and Whitening, pages 774–787. Springer Berlin Heidelberg, Berlin, Heidelberg.

[51] Jégou, H., Douze, M., and Schmid, C. (2008). Hamming embedding and weak geometric consistency for large scale image search. *Computer Vision–ECCV 2008*, pages 304–317.

[52] Jégou, H., Douze, M., and Schmid, C. (2010a). Improving bag-of-features for large scale image search. *International Journal of Computer Vision*, 87(3):316–336.

[53] Jégou, H., Douze, M., and Schmid, C. (2011). Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128.

[54] Jégou, H., Douze, M., Schmid, C., and Pérez, P. (2010b). Aggregating local descriptors into a compact image representation. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3304–3311.

[55] Jegou, H., Perronnin, F., Douze, M., Sánchez, J., Perez, P., and Schmid, C. (2012). Aggregating local image descriptors into compact codes. *IEEE transactions on pattern analysis and machine intelligence*, 34(9):1704–1716.

[56] Jégou, H. and Zisserman, Z. (2014). Triangulation embedding and democratic aggregation for image search. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3310–3317.

[57] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM International Conference on Multimedia*, pages 675–678. ACM.

[58] Jiang, M., Huang, S., Duan, J., and Zhao, Q. (2015). Salicon: Saliency in context. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1072–1080.

[59] Jianxin, W. and Rehg, J. (2011). Centrist: A visual descriptor for scene categorization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(8):1489–1501.

[60] Jimenez, A., Alvarez, J. M., and Giro-i Nieto, X. (2017). Class-weighted convolutional features for visual instance search. In *28th British Machine Vision Conference (BMVC)*.

[61] Kalantidis, Y., Mellina, C., and Osindero, S. (2016). Cross-dimensional weighting for aggregated deep convolutional features. In *European Conference on Computer Vision VSM Workshop*, pages 685–701. Springer International Publishing.

[62] Karpathy, A. and Fei-Fei, L. (2017). Deep visual-semantic alignments for generating image descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):664–676.

[63] Kokare, M., Chatterji, B. N., and Biswas, P. K. (2003). Comparison of similarity metrics for texture image retrieval. In *TENCON 2003. Conference on Convergent Technologies for Asia-Pacific Region*, volume 2, pages 571–575 Vol.2.

[64] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[65] Kruthiventi, S., Ayush, K., and Babu, R. (2017). Deepfix: A fully convolutional neural network for predicting human eye fixations. *IEEE Transactions on Image Processing*, 26:4446–4456.

[66] Kulis, B. (2013). Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364.

[67] Kümmerer, M., Theis, L., and Bethge, M. (2014). Deep gaze i: Boosting saliency prediction with feature maps trained on imagenet. *arXiv preprint arXiv:1411.1045*.

[68] Larlus, D. and Gordo, A. (2017). Beyond instance-level image retrieval: Leveraging human captions to learn representations for semantic visual search. In *IEEE Conference on Computer Vision and Pattern Recognition.*

[69] Laskar, Z. and Kannala, J. (2017). Context aware query image representation for particular object retrieval. In *Scandinavian Conference on Image Analysis*, pages 88–99. Springer.

[70] Lazebnik, S., Schmid, C., and Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 2169–2178. IEEE.

[71] Lennie, P. (2003). The cost of cortical computation. *Current biology*, 13(6):493–497.

[72] Li, Y., Crandall, D. J., and Huttenlocher, D. P. (2009). Landmark classification in large-scale image collections. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1957–1964. IEEE.

[73] Li, Y., Xu, Y., Wang, J., Miao, Z., and Zhang, Y. (2017). MS-RMAC: Multiscale regional maximum activation of convolutions for image retrieval. *IEEE Signal Processing Letters*, 24(5):609–613.

[74] Lin, Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

[75] Lindeberg, T. (1994). *Scale-Space Theory in Computer Vision.* Kluwer Academic Publishers, Norwell, MA, USA.

[76] Liu, J. (2013). Image retrieval based on bag-of-words model. *arXiv preprint arXiv:1304.5168.*

[77] Lloyd, S. (2006). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137.

[78] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440.

[79] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

[80] Luo, W., Li, Y., Urtasun, R., and Zemel, R. (2016). Understanding the effective receptive field in deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 4898–4906.

[81] McGuinness, K., Mohedano, E., Zhang, Z., Hu, F., Albatal, R., Gurrin, C., O'Connor, N., Smeaton, A., Salvador, A., Giró-i Nieto, X., and Ventura, C. (2014). Insight centre for data analytics (DCU) at TRECVid 2014: instance search and semantic indexing tasks. In *2014 TREC Video Retrieval Evaluation Notebook Papers and Slides*.

[82] McLachlan, G. and Peel, D. (2004). *Finite Mixture Models*. Wiley, Newark, NJ.

[83] Mei, T., Rui, Y., Li, S., and Tian, Q. (2014). Multimedia search reranking: A literature survey. *ACM Computing Surveys (CSUR)*, 46(3):38.

[84] Mikolajczyk, K. and Matas, J. (2007). Improving descriptors for fast tree matching by optimal linear projection. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8.

[85] Mikolajczyk, K. and Schmid, C. (2004). Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86.

[86] Mikulík, A., Perdoch, M., Chum, O., and Matas, J. (2012). Learning vocabularies over a fine quantization. *International Journal of Computer Vision*, 103:163–175.

[87] Morin, F. and Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In Cowell, R. G. and Ghahramani, Z., editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 246–252. Society for Artificial Intelligence and Statistics.

[88] Nguyen, V., Nguyen, D., Tran, M., Le, D., Duong, D., and Satoh, S. (2015). Query-adaptive late fusion with neural network for instance search. In *International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6.

[89] Nister, D. and Stewenius, H. (2006a). Scalable recognition with a vocabulary tree. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, pages 2161–2168, Washington, DC, USA. IEEE Computer Society.

[90] Nister, D. and Stewenius, H. (2006b). Scalable recognition with a vocabulary tree. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 2161–2168. Ieee.

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

[92] Ong, E., Husain, S., and Bober, M. (2017). Siamese network of deep fisher-vector descriptors for image retrieval. *arXiv preprint arXiv:1702.00338*.

[93] Pan, J., Canton, C., McGuinness, K., O'Connor, N., Torres, J., Sayrol, E., and Giro-i Nieto, X. (2017). Salgan: Visual saliency prediction with generative adversarial networks. *arXiv preprint arXiv:1701.01081*.

[94] Pan, J., Sayrol, E., Giro-i Nieto, X., McGuinness, K., and O'Connor, N. (2016). Shallow and deep convolutional networks for saliency prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 598–606.

[95] Perronnin, F., Liu, Y., Snchez, J., and Poirier, H. (2010). Large-scale image retrieval with compressed fisher vectors. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3384–3391.

[96] Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. (2007). Object retrieval with large vocabularies and fast spatial matching. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

[97] Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. (2008). Lost in quantization: Improving particular object retrieval in large scale image databases. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*

[98] Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145 – 151.

[99] Radenovic, F., Schnberger, J. L., Ji, D., Frahm, J. M., Chum, O., and Matas, J. (2016). From dusk till dawn: Modeling in the dark. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5488–5496.

[100] Radenović, F., Tolias, G., and Chum, O. (2016). CNN image retrieval learns from BoW: Unsupervised fine-tuning with hard examples. In *European Conference on Computer Vision*, pages 3–20. Springer International Publishing.

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

[102] Ren, S., He, K., Girshick, R., and Sun, J. (2017). Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149.

[103] Romberg, S., Pueyo, L. G., Lienhart, R., and van Zwol, R. (2011). Scalable logo recognition in real-world images. In *Proceedings of the 1st ACM International*

Conference on Multimedia Retrieval, ICMR '11, pages 25:1–25:8, New York, NY, USA. ACM.

[104] Rubner, Y., Tomasi, C., and Guibas, L. J. (2000). The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121.

[105] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA.

[106] Salvador, A., Giró-i Nieto, X., Marqués, F., and Satoh, S. (2016). Faster r-cnn features for instance search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 9–16.

[107] Saxe, A., Koh, P. W., Chen, Z., Bhand, M., Suresh, B., and Ng, A. Y. (2011). On random weights and unsupervised feature learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 1089–1096.

[108] Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823.

[109] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*.

[110] Sharif Razavian, A., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). CNN features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813.

[111] Simo-Serra, E., Trulls, E., Ferraz, L., Kokkinos, I., and Moreno-Noguer,

F. (2014). Fracking deep convolutional image descriptors. *arXiv preprint arXiv:1412.6537*.

[112] Simonyan, K., Parkhi, O. M., Vedaldi, A., and Zisserman, A. (2013). Fisher Vector Faces in the Wild. In *British Machine Vision Conference, BMVC 2013, Bristol, UK, September 9-13, 2013*.

[113] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[114] Sivic, J. and Zisserman, A. (2003). Video google: a text retrieval approach to object matching in videos. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 1470–1477 vol.2.

[115] Sivic, J. and Zisserman, A. (2009). Efficient visual search of videos cast as text retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):591–606.

[116] Smeaton, A. F., Over, P., and Kraaij, W. (2006). Evaluation campaigns and TRECVid. In *Proceedings of the 8th ACM international workshop on Multimedia information retrieval*, pages 321–330. ACM.

[117] Smeulders, A., Worring, M., Santini, S., Gupta, A., and Jain, R. (2000). Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380.

[118] Sohn, K. (2016). Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*, pages 1857–1865.

[119] Swain, M. and Ballard, D. (1991a). Color indexing. *International Journal of Computer Vision*, 7:11–32.

[120] Swain, M. J. and Ballard, D. H. (1991b). Color indexing. *International Journal of Computer Vision*, 7(1):11–32.

[121] Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1701–1708.

[122] Tamura, H. and Yokoya, N. (1984). Image database systems: A survey. *Pattern Recognition*, 17(1):29 – 43. Knowledge Based Image Analysis.

[123] Tieleman, T. and Hinton, G. (2012). Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.

[124] Tolias, G. and Jégou, H. (2014). Visual query expansion with or without geometry: Refining local descriptors by feature aggregation. *Pattern Recognition*, 47(10):3466 – 3476.

[125] Tolias, G., Sicre, R., and Jégou, H. (2016). Particular object retrieval with integral max-pooling of cnn activations. In *International Conference on Learning Representations*.

[126] Turcot, P. and Lowe, D. G. (2009). Better matching with fewer features: The selection of useful features in large database recognition problems. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 2109–2116. IEEE.

[127] Uijlings, J. R., Van De Sande, K. E., Gevers, T., and Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*, 104(2):154–171.

[128] Vedaldi, A. (2014). Visualindex. `https://github.com/vedaldi/visualindex`.

[129] Vinje, W. E. and Gallant, J. L. (2000). Sparse coding and decorrelation in primary visual cortex during natural vision. *Science*, 287(5456):1273–1276.

[130] Vo, N. N. and Hays, J. (2016). Localizing and orienting street views using overhead imagery. In *European Conference on Computer Vision*, pages 494–509. Springer.

[131] Wan, J., Wang, D., Hoi, S. C. H., Wu, P., Zhu, J., Zhang, Y., and Li, J. (2014). Deep learning for content-based image retrieval: A comprehensive study. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 157–166. ACM.

[132] Wang, J., Song, Y., Leung, T., Rosenberg, C., Wang, J., Philbin, J., Chen, B., and Wu, Y. (2014). Learning fine-grained image similarity with deep ranking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1386–1393.

[133] Wang, S. and Jiang, S. (2015). Instre: A new benchmark for instance-level object retrieval and recognition. *ACM Transactions on Multimedia Computing Communications and Applications*, 11(3):37:1–37:21.

[134] Wei, X. S., Luo, J. H., J. Wu, J., and Zhou, Z. H. (2017). Selective convolutional descriptor aggregation for fine-grained image retrieval. *IEEE Transactions on Image Processing*, 26(6):2868–2881.

[135] Wu, J. and Rehg, J. (2009). Beyond the euclidean distance: Creating effective visual codebooks using the histogram intersection kernel. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 630–637. IEEE.

[136] Xu, P., Ehinger, K. A., Zhang, Y., Finkelstein, A., Kulkarni, S. R., and Xiao, J. (2015). Turkergaze: Crowdsourcing saliency with webcam based eye tracking. *arXiv preprint arXiv:1504.06755*.

[137] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, pages 3320–3328, Cambridge, MA, USA. MIT Press.

[138] Yue-Hei Ng, J., Yang, F., and Davis, L. S. (2015). 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.

[139] Zagoruyko, S. and Komodakis, N. (2015). Learning to compare image patches via convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4353–4361.

[140] Zeiler, M. D., Taylor, G. W., and Fergus, R. (2011). Adaptive deconvolutional networks for mid and high level feature learning. In *2011 International Conference on Computer Vision*, pages 2018–2025.

[141] Zhang, W. and Ngo, C. (2015). Topological spatial verification for instance search. *IEEE Transactions on Multimedia*, 17(8):1236–1247.

[142] Zhang, Y., Jia, Z., and Chen, T. (2011). Image retrieval with geometry-preserving visual phrases. In *Computer Vision and Pattern Recognition (CVPR)*, pages 809–816.

[143] Zhang, Y., Yu, F., Song, S., Xu, P., Seff, A., and Xiao, J. Large-scale scene understanding challenge: Eye tracking saliency estimation.

[144] Zheng, L., Yang, Y., and Tian, Q. (2017). SIFT meets CNN: A decade survey of instance retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

[145] Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. (2014a). Object detectors emerge in deep scene CNNs. In *International Conference on Learning Representations (ICLR).*

[146] Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. (2016). Learning deep features for discriminative localization. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2921–2929.

[147] Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., and Oliva, A. (2014b). Learning deep features for scene recognition using places database. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 487–495. Curran Associates, Inc.

[148] Zhou, X., Zhu, C.-Z., Zhu, Q., Satoh, S., and Guo, Y.-T. (2014c). A practical spatial re-ranking method for instance search from videos. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 3008–3012. IEEE.

[149] Zhu, C.-Z. and Satoh, S. (2012). Large vocabulary quantization for searching instances from videos. In *Proceedings of the 2nd ACM International Conference on Multimedia Retrieval*, page 52. ACM.