Aggregating Deep Features For Image Retrieval

Mohammad Eiad Kazkaz
Aggregating Deep Features For Image Retrieval

Mohammad Eiad Kazkaz

Thesis submitted to the
Benjamin M. Statler College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Electrical Engineering

Victor Fragoso, Ph.D., Chair
Nasser M. Nasrabadi, Ph.D.
Xin Li, Ph.D.

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2017

Keywords: Convolutional Neural Network, Deep Learning, Image retrieval, off-the-shelf pre-trained model, Max-pooling, Aggregation, Image representation

Copyright 2017 Mohammad Eiad Kazkaz
Abstract

Aggregating Deep Features For Image Retrieval

Mohammad Eiad Kazkaz

Measuring visual similarity between two images is useful in several multimedia applications such as visual search and image retrieval. However, measuring visual similarity between two images is an ill-posed problem which makes it a challenging task. This problem has been tackled extensively by the computer vision and machine learning communities. Nevertheless, with the recent advancements in deep learning, it is now possible to design novel image representations that allow systems to measure visual similarity more accurately than existing and widely adopted approaches, such as Fisher vectors. Unfortunately, deep-learning-based visual similarity approaches typically require post-processing stages that can be computationally expensive. To alleviate this issue, this thesis describes deep-learning-based visual image representations that allow a system to measure visual similarity without requiring post-processing stages. Specifically, this thesis describes max-pooling-based aggregation layers that combined with a convolutional-neural-network-based produce rich image representations for image retrieval without requiring an expensive post-processing stages. Moreover, the proposed max-pooling-based aggregation layers are general and can be seamlessly integrated with any existing and pre-trained networks. The experiments on large-scale image retrieval datasets confirm that the introduced image representations yield visual similarity measures that achieve a comparable or better retrieval performance than state-of-the art approaches, without requiring expensive post-processing operations.
Contents

Abstract ii
List of Figures v
List of Tables viii

1 Introduction 1
   1.1 Problem Statement ........................................... 3
   1.2 Organization of the Thesis .................................. 4

2 Literature Review 5
   2.1 Local Features Based Image Retrieval ....................... 6
   2.2 Deep Learning Based Image Retrieval ....................... 8
      2.2.1 Using off-the-shelf Networks .......................... 10
      2.2.2 Fine-tuning using Classification Loss ............... 12
      2.2.3 Learning Distance Metric (Embedding) ............... 14

3 Methodology 17
   3.1 Convolutional Neural Networks .............................. 17
      3.1.1 Brief Review of Convolutional-Neural-Network Architectures 18
      3.1.2 Selecting the Convolutional layer for Image Features .... 19
      3.1.3 Deep features ........................................... 20
      3.1.4 Brief review of Max-pooling layers ................... 21
   3.2 Proposed Aggregation Method ............................... 22
      3.2.1 Max-Pooling Aggregation ............................... 23
      3.2.2 Discussion .............................................. 24
      3.2.3 Multi-Resolution Max-Pooling ......................... 24
      3.2.4 Global Average Pooling ............................... 25

4 Experiments and Results 29
   4.1 Comparison with the state of art ........................... 29
      4.1.1 Datasets .............................................. 30
      4.1.2 Evaluation metric ...................................... 32
   4.2 Experiments .................................................. 32
      4.2.1 Max-pooling window size ............................... 33
      4.2.2 Max-pooling stride size ............................... 34
CONTENTS

4.2.3 Truncate max-pooling layer of the last convolutional layer of VGG16 36
4.2.4 Multi-scale max-pooling ........................................... 39
4.3 Comparison with the state-of-the-art ................................ 44

5 Conclusion 52
5.1 Future directions .......................................................... 55

References 56

Approval Page i
List of Figures

1.1 Overview of an image retrieval system. First, the system receives a query image. Then, the system computes an image representation; the representation stage can be viewed as a mapping between images and high-dimensional feature vectors. Subsequently, the system compares the computed representation with all the image representations of an image database. This comparison measures distances or similarities between the query image representation and a representation corresponding to an image in a database. Once all the distances or similarity values are computed, the system ranks the images in the database from the most similar to the least similar. .............................. 2

2.1 VGG architecture. The left box (red-dashed lines) includes the convolutional layers that compute the image features for a visual recognition task. The right box (red-dashed lines) includes the fully connected layers to classify an input. However, in image retrieval, the classifier part of the network is not needed. The reason is that both tasks are different. However, the image features are informative for the task of image retrieval. ................................. 9

2.2 Computation of a feature map using a convolutional neural network (CNN). First, an image is passed through the feature extractor of a pre-trained CNN (e.g., the VGG network [1]). The output of this pass is a cubic array of numbers or tensor dubbed the feature maps. Deep-learning-based image representation use these feature maps to compute a single vector to encode the visual information in an image. ........................................... 10

3.1 Proposed aggregation method using off-the-shelf model. Imaged passed into the pre-trained model to extracted deep features, discriminative information is obtained by max-pooling aggregation, then global average pooling to provide a 512-dimensional vector, equals to the number of the convolutional filters in the last convolutional layer of the pre-trained model. ............................. 18

3.2 Example of a max-pooling operation. The max-pooling window size is (2 × 2) and the stride is (2×2). With this configuration, the max-pooler only analyzes the four windows shown in the left matrix (i.e., blue, green, pink, and purple). Out of those colored regions, the max-pooling operation returns the maximum value in the colored patch, which is of size (2 × 2); matrix on the right. .... 21
3.3 General processing pipeline for deep-learning-base aggregation for image retrieval. Extract and aggregate features has three stages: 1) Using pre-trained model as a features extractor; 2) aggregation method; and 3) post processing steps. The output is a compact vector with discriminative information called an image representation. ........................................... 22

3.4 Proposed max-pooling aggregation method. Applying overlapping max-pooling on a 3D feature maps, outputs another 3D tensor with lower spatial dimensionality while still maintaining discriminative information. .................. 24

3.5 Overview of the multi-scale max-pooling. First, the method applies different max-pooling operations on the 3D tensor. For every output of each max-pooling, the system computes a single vector representing the input image. Lastly, the method combines all the obtained vectors to generate a single image representation. ........................................... 25

3.6 Global Average Pooling. Global average pooling takes the average of all the values at a spatial location and outputs a scalar value. For example a spatial feature (green) turns into a scalar value (green). .................. 26

3.7 Performing global average pooling over the 3D tensor of discriminative feature acquired from max-pooling aggregation results in a vector of dimensionality equals the number of channels in the feature maps. .................. 27

4.1 Effect of max-pooling window size when keeping max-pooling layer after the last convolutional layer. Blue curve is the performance for database retrieval. The performance decreases by decreasing max-pooling window size. ........ 33

4.2 Effect of max-pooling stride size when keeping max-pooling layer after the last convolutional layer. The stride achieves a maximum performance at $h/3$ and then the performance stays flat. ........................................... 35

4.3 Effect of max-pooling window size when truncating max-pooling layer after the last convolutional layer. ........................................... 38

4.4 Effect of max-pooling stride size when truncating max-pooling layer after the last convolutional layer. ........................................... 38

4.5 Comparing between multi-scale max-pooling aggregation when summing and concatenating all $N$ vectors produced by $N$ max-pooling. Evaluation when retrieving all the images in the database. Summation is more robust and outperforms concatenation in case of database retrieval. .................. 42

4.6 Comparing between multi-scale max-pooling aggregation when summing and concatenating all $N$ vectors produced by $N$ max-pooling. Evaluation when retrieving positive images for queries. Summation is more robust and outperforms concatenation in case of positive retrieval. .................. 42

4.7 Comparing the four tested methods of max-pooling aggregation on Oxford dataset. Retrieving positive only. Single max-pooling aggregation when keeping max-pooling after last CNN layer (green), truncationg max-pooling (purple), multi-scale max-pooling aggregation by summing all vectors of $N$ max-poolings (blue), concatenation $N$ vector (red). While all the four method have the same maximum performance, multi-scale max-pooling when summing $N$ vectors is more robust to spatial resolution changes. .................. 47
4.8 Comparing the four tested methods of max-pooling aggregation on Oxford dataset. Retrieving all images in the dataset (i.e., database). Single max-pooling aggregation when keeping max-pooling after last CNN layer (green), truncating max-pooling (purple), mult-scale max-pooling aggregation by summing all vectors of $N$ max-poolings (blue), concatenation $N$ vector (red). While all the four method have the same maximum performance, multi-scale max-pooling when summing $N$ vectors is more robust to spatial resolution changes. ................................................................. 48

4.9 Visualization of the image representation of a query image, relevant image and non-relevant image which was mistakenly retrieved as a relevant image. Similar visual patterns appear in image representations, which confirms that max-pooling preserving discriminative information. White cell correspond to the maximum value and black cells correspond to minimum values in the image representation ................................................................. 49

4.10 Retrieval of top 10 images for the query associated to a landmark. Query images are cropped in cyan bounding box defined by the Oxford dataset. Red box show false positive images that were mistakenly retrieve as relevant image 50

4.11 Retrieval of top 10 images for the query associated to a landmark. Query images are cropped in cyan bounding box defined by the Oxford dataset. Red box show false positive images that were mistakenly retrieve as relevant image 50
List of Tables

4.1 Image retrieval datasets used in this thesis. .................................................. 31
4.2 Effect on the window size of the max pooler when keeping the last max-pooling layer. Performance decreases by decreasing max-pooling window size. Also, overlapping between consecutive max-pooling is important. About quarter of the max-pooling windows are overlapped. Stride is fixed to tune the window size. .................................................. 34
4.3 Effect on the stride of the max-pooler when keeping max-pooling at the last convolutional layer. .................................................. 36
4.4 Effect on the window size of the max pooler when truncating the last max-pooling layer .................................................. 37
4.5 Truncate max-pooling from the last CNN layer. Stride size effect ................. 39
4.6 Mutli-scale max-pooling aggregation. Window size effect when summing the image representation associated with each max-pooling aggregation vector. Oxford dataset .................................................. 40
4.7 Mutli-scale max-pooling aggregation. Window size effect when concatenation the image representation associated with each max-pooling aggregation vector. Oxford dataset .................................................. 41
4.8 Mutli-scale max-pooling aggregation. Window size effect when concatenation the image representation associated with each max-pooling aggregation vector. Oxford dataset .................................................. 43
4.9 Experiments. ......................................................................................... 51
Chapter 1

Introduction

Recent advances in machine learning and computer vision [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 1, 13, 14] have shown that computers can achieve a remarkable visual recognition performance by following a supervised learning strategy. To achieve such a great performance, these recent methods require powerful computational resources and a vast amount of data to learn from. Specifically, this set of data must contain a large collection of exemplars of every concept to learn, e.g., thousands of images depicting dogs, cars, airplanes, among other objects.

The main component boosting the visual recognition performance of the aforementioned advancements is the convolutional neural network (CNN) [15, 16]. The CNN achieves such a great visual recognition performance since it is comprised of a series of bank of filters which are sequentially applied to an image to detect patterns. However, unlike previous methods relying on human-engineered pattern or feature detectors [17, 18], the CNN is trained to learn the important features in an image and the bank of image filters to detect them. Given that a CNN of learns from a vast set of labeled exemplars, the CNN is capable of outperforming previous methods using human-engineered pattern or feature detectors.

While CNN are useful for visual recognition tasks, they turned to be useful for other computer vision applications as well. Among those applications, the one this thesis focuses on is the task of image retrieval. In short, image retrieval aims to get the most similar images from a database given a query image. In comparison with a common web search engine that retrieves relevant documents on the web using a text string, image retrieval focuses only on getting relevant or similar images given a query image. Fig. 1.1 shows an illustration of an image retrieval system. First, the system receives a query image. Then, the system computes an image representation, typically with a high-dimensional feature vector. Subsequently, the
Figure 1.1: Overview of an image retrieval system. First, the system receives a query image. Then, the system computes an image representation; the representation stage can be viewed as a mapping between images and high-dimensional feature vectors. Subsequently, the system compares the computed representation with all the image representations of an image database. This comparison measures distances or similarities between the query image representation and a representation corresponding to an image in a database. Once all the distances or similarity values are computed, the system ranks the images in the database from the most similar to the least similar.

Image retrieval can have different applications. For instance, enabling an image-based search modality in web search engines, organize large photo collections based on image similarities, image inpainting by retrieving similar images from the web, among others.

Image retrieval is a different problem in comparison with the problem of image recognition. The task in image retrieval is to return a list of the most similar images in a database. On the other hand, the goal of image recognition is to recognize objects in an image. For instance, identify if the Eiffel tower is present in the image. Although both problems have different goals, they depend on an image representation component that captures useful information to either recognize objects in an image or to measure image similarity. Recent advancements have focused mainly on the recognition task, and many recent image retrieval systems leverage the image representations that recognition systems have developed. While visual recognition systems have achieved a certain maturity thanks to the use of CNNs, image retrieval is still in the quest of techniques that can improve the performance of retrieving the most similar images given a query image.
1.1 Problem Statement

Given the difference in goals between image retrieval and recognition, the learning problems are posed differently. The goal in a visual recognition system is well defined. It is to recognize certain objects with clearly defined classes. On the other hand, the goal in image retrieval is to obtain the most similar images given a query image. Rather than recognizing an object, image retrieval systems have to use an informative image representation and measure visual similarity to rank the entries in a database prior to return the set of the most similar instances. Unlike the recognition problem which is well defined because visual categories are well defined, the learning problem of finding a visual similarity function is not well defined. This is because image similarity can face subjective criterion which is hard to formalize in a mathematical problem.

Given the challenging task of defining a learning problem for measuring visual similarity, many approaches have tried different ways to pose the learning of a visual similarity function in a mathematical fashion. Among the approaches that focus only on learning a visual similarity function, the most salient ones are based on learning a distance [19, 20, 21] or similarity function [22, 23] to measure image similarity, and directly learning a function that ranks [24, 25, 26] given an visual query. While these methods successfully find useful functions to measure image similarity, they assume that the image representation is rich in useful information for measuring visual similarity. With the recent advancements in visual recognition, many recently developed deep-learning-based methods [27, 28, 29] focus more on learning an image representation than a visual similarity function. This is because an informative representation contributes the most to the image retrieval performance, as in the recognition problem.

While the recent advances use deep learning to learn informative image representations for image retrieval, their representation process executes an expensive phase where the high-dimensional vector (i.e., the image representation) is “normalized” so that they can be used in combination with a distance or similarity measure. Many methods use Principal Component Analysis (PCA) to reduce the dimensionality of an image representation and whitening the compressed vector in their normalization stage; this stage is typically the last stage when computing the image representation. While this normalization stage brings benefits in retrieval performance, it can require a significant amount of memory and computational time. This is because PCA requires the computation of a covariance matrix which can be
large when the image representation uses a high-dimensional vector.

The focus of the thesis is to find an inexpensive method that allows a faster normalization stage without requiring PCA. To this end, the thesis describes a method based on max-pooling operations, which are common in a CNN and are efficient to implement. The proposed method not only is inexpensive to compute, but also allows a user to define the desired dimensionality of the image representation. The proposed method is significantly more efficient since this method does not estimate a covariance matrix for dimensionality reduction and whitening.

1.2 Organization of the Thesis

The thesis is organized as follows. In Chapter 2, the thesis discusses prior work on image retrieval, and in particular describes image retrieval techniques based on CNNs. Then, the thesis describes the proposed approach to image retrieval in Chapter 3. Subsequently, the thesis presents a series of experiments comparing the proposed approach with state-of-the-art image retrieval methods, and describes the results of several evaluations that demonstrate the efficiency and efficacy of the proposed approach for image retrieval in Chapter 4. Finally, in Chapter 5, the thesis discusses a series of conclusions.
Chapter 2

Literature Review

Many methods have been proposed and developed for image retrieval. These approaches can be grouped in two categories:

- Local features based image retrieval: These methods are based on local features such as SIFT [18]. These methods represent an image by aggregating all the descriptors (feature vectors) of salient patterns in an image. The aggregation is carried out by clustering the feature vectors, and generating another high-dimensional vector encoding the distribution of these image descriptors.

- Deep learning based image retrieval: These methods are based on deep features: patterns detected by a convolutional neural network (CNN). These methods use pre-trained models to compute the deep features of an image and use post-processing stages to aggregate the extracted deep features in order to produce an image representation. The common aggregation of deep features are summation, fine-tuning, or the training of a ranking loss to get compact representations.

An ideal image representation, regardless of the image representation approach, meets the following criteria:

- Compact image representation to keep memory footprint low.

- Has discriminative information.

- Fast-to-compute and compare for image search in a large dataset (e.g., millions of images).
2.1 Local Features Based Image Retrieval

The general approach of these methods is detecting salient points in an image, cluster the patches around the salient points or features into a chosen number of groups, then aggregate or summarize the distribution of the features with respect to the clusters into an image representation. To detect salient point in an image, Lowe [18] introduced scale invariant feature transform (SIFT), which is robust to scale, rotation invariant, and affine distortion. SIFT features are repeatable and often are re-detectable which makes them useful for image retrieval.

The earliest use of SIFT descriptors for image retrieval was introduced by Sivic et al. [30]. Inspired by text-based document retrieval, they proposed Bag-of-words (BoW) for image retrieval - extracted from videos using SIFT descriptors. They used SIFT descriptors because of their robustness to scale, rotation invariance and affine distortion. In text-based document retrieval [31], each document is parsed into words; words are represented by their roots, for example ‘go’, ‘going’ and ‘goes’ would be represented as ‘go’. Very frequent words are ignored because they do not provide any discriminative information. The remaining words (i.e., the non-frequent words) contribute the most to the final document representation. Each document is represented by a vector containing the frequency of the words in the document.

To get the visual words for an image, the method extracts SIFT descriptors, then clusters them according to their closest visual word using K-means. Each cluster, calculated by K-means, is called a visual word, and the set of clusters is called visual vocabulary. This method chooses the number of cluster empirically to give the best performance using a training set. Each vector represents weighted word frequencies: $V_d = (t_1, ..., t_i, ..., t_k)^T$, where

$$t_i = \frac{n_{id}}{n_d} \log \frac{N}{n_i},$$

(2.1)

$n_{id}$ is the occurrences of the word $i$ in the document $d$, $n_d$ is the total number of words in the document, $n_i$ is the number of occurrences of term $i$ in the whole database, and $N$ is the number of documents in the whole database. In the retrieval phase, the method of BoW ranks vectors of visual words according to weight similarity between each query vector and all weighted vectors. The weighting is the outcome of the product of the word frequency ($n_{id}/n_d$) and the inverse document frequency $\log (N/n_i)$.

BoW has a limitation for large-scale images. It requires various clustering passes to a vast set of features, a large memory footprint, and it is not always applicable to various encoding
techniques. Perronnin et al. [32] proposed to use Fisher Kernel for image retrieval since the Fisher kernel is good for categorization, dimensionality reduction, and saliency detection. They represented the images with Fisher vectors (FV) which encode what makes an image different from other images. Intuitively, the FV first clusters a vast set of SIFT features, describes the distribution of those features with respect to a Gaussian Mixture Model [33], and computes an image representation based on the GMM. A crucial step to increase the performance in image retrieval is to normalize the vector representation using an L2 norm. Several previous works [28, 32, 34, 35, 36] report that this step alone increases the retrieval performance significantly. After computing a vectorial representation of the image based on the calculated GMM, Perronnin et al. [32] also normalized their Fisher vector which crucially increases the retrieval accuracy. Finally, they applied standard binary encoding techniques since Fisher vector is dense.

Specifically, to generate a Fisher vector, they computed SIFT descriptors \( \{x_t\} \) for each image. As mentioned earlier, Fisher vectors use a GMM instead of a k-means for clustering and represent the SIFT descriptors. Recall that a Gaussian Mixture Model:

\[
p(x) = \sum_{i=1}^{N} w_i p_i(x), \tag{2.2}
\]

has three parameters: mixture weight \( w_i \), mean \( \mu_i \) and covariance matrix \( \Sigma_i \), where

\[
p_i(x) = \mathcal{N}(x; \mu_i, \Sigma_i), \tag{2.3}
\]

and \( \mathcal{N} \) is the multivariate Gaussian probability density function (pdf). Note that the covariance matrix is diagonal, which means that the assumed GMM models each dimension of the SIFT descriptors independently. In Eq.(2.2) Gaussian distribution \( p_i \) can be seen as a visual word and \( N \) as the vocabulary size. Given the GMM, each Fisher descriptor is calculated by computing the following terms:

\[
g^X_i = \frac{1}{T \sqrt{w_i}} \sum_{t=1}^{T} \gamma_t(i) \left( \frac{x_t - \mu_i}{\sigma_i} \right), \tag{2.4}
\]

where

\[
\gamma_t(i) = \frac{w_i p_i(x_t)}{\sum_{j=1}^{N} w_j p_j(x_t)}. \tag{2.5}
\]

In addition to the terms \( g^X_i \), the Fisher vector computes the following terms:

\[
v^X_i = \frac{1}{T \sqrt{2 w_i}} \sum_{t=1}^{T} \gamma_t(i) \left[ \frac{(x_t - \mu_i)^2}{\sigma_i^2} - 1 \right]. \tag{2.6}
\]
The Fisher vector of one image is computed by stacking the $g^X_i$ and $v^x_i$ computed using Eq. (2.4) and (2.6). Thus, the dimensionality of the Fisher vector is $2 \times d \times k$, where $d$ is the dimensionality of the local descriptors (e.g., SIFT), and $k$ is the total number of clusters. While Fisher vectors introduced a useful and powerful image representation for large-scale images retrieval, it does not scale for large databases because of their high dimensionality. To overcome this issue, Jegou et al. [35] considered the problem of searching for the most similar image to the query image in a very large image database while maintaining a good accuracy and using a compact representation. They proposed the Vector of Locally Aggregated Descriptor (VLAD). To compute a VLAD vector, they first learn a code-book of $k$-centroids using K-means. Then, VLAD assigns every local invariant descriptor $x$ (such as SIFT) to the nearest visual word using k-NN (nearest center) $c$. Subsequently, it computes the following residual vector $(x - c)$, where $x$ and $c$ are a local descriptor and a centroid, respectively. Assuming that the local descriptor is of $d$-dimensional, the VLAD representation is a $D = K \times d$-dimensional produced by concatenating the following terms:

$$v_i = \sum_{x \text{ such that } NN(x) = c_i} (x_t - c_i),$$

(2.7)

where $x_t$ is the $t$-th local descriptor, and $c_i$ is the $i$-th centroid. After concatenating the terms, the final vector is L2-normalized.

The Fisher and VLAD vectors were the state-of-the-art in image retrieval and other image recognition tasks (e.g., image classification). Nevertheless, with the recent advancements in deep learning, various approaches devised image representations based on convolutional neural networks (CNN). Inspired by several visual recognition advancements, image retrieval also adopted CNNs as their main engine to represent images.

### 2.2 Deep Learning Based Image Retrieval

Deep convolutional neural networks based methods are the state of the art for many machine learning and computer vision tasks, as discussed in Chapter 1. The work of deep-learning-based image retrieval can be categorized in three groups: 1) aggregating deep features using off-the-shelf pre-trained model; 2) learning distance metrics so that similar examples are mapped close to each other and dissimilar examples are mapped farther apart through a given margin; and 3) fine-tuning for the pre-trained network.
Figure 2.1: VGG architecture. The left box (red-dashed lines) includes the convolutional layers that compute the image features for a visual recognition task. The right box (red-dashed lines) includes the fully connected layers to classify an input. However, in image retrieval, the classifier part of the network is not needed. The reason is that both tasks are different. However, the image features are informative for the task of image retrieval.

The basic tool for deep-learning-based image representations is the convolutional neural network (CNN). CNNs are composed of sequences of convolutional layers (arrays of image filters), and fully connected layers of a CNN, which perform the classification. In Fig 2.1, we show the VGG-CNN [1] architecture to illustrate the general pipeline of a CNN for image retrieval. In that figure, we show two boxes (red-dashed line). The left box includes the layers in charge of extracting image features, while the right box includes the fully connected layers that are in charge of classification. It has been shown [37] that the convolutional layers extract informative image descriptions for several visual tasks, including image retrieval. This is because deep features are compact and contain rich discriminative information. A forward pass of an image through the feature extractor of a pre-trained CNN model produces a cubical array of numbers dubbed feature maps. Many deep-learning based image representation aggregate the information in these feature maps to compute an image representation vector. In Fig 2.2, we show an illustration of how to obtain the feature maps using a pre-trained model; in this case the VGG architecture. First, the image is passed through the feature extractor of the CNN, and the extractor produces a tensor (or a cubic array of numbers) which represents the feature maps.
Figure 2.2: Computation of a feature map using a convolutional neural network (CNN). First, an image is passed through the feature extractor of a pre-trained CNN (e.g., the VGG network [1]). The output of this pass is a cubic array of numbers or tensor dubbed the feature maps. Deep-learning-based image representation use these feature maps to compute a single vector to encode the visual information in an image.

2.2.1 Using off-the-shelf Networks

Features extracted with a pre-trained deep neural network tend to be generic [38]. As such, these features work well in several computer vision tasks. Based on this observation, Razavian et al. [37] went farther and showed that these deep features are powerful for many computer vision tasks including image retrieval. They stated that the object of interest might be at different locations and scales in an image. Motivated by this observation, their approach divides the image into multiple overlapping regions of different sizes, and then extracts deep features for each region. They used OverFeat pre-trained model [39] and used the first fully connected layer as the features representation followed by L2 normalization. They defined the distance between a reference image and query sub-region as the minimum L2 distance between the query sub-region and the reference image sub-region. The distance between the reference image and the query image is the average distance of each query sub-region and the reference image. The image representation dimensionality is 4096. To reduce the dimensionality, they applied PCA and whitening followed by an L2 normalization.

Different from the approach proposed by Razavian et al. [37], Babenko et al [28] observed that the region of interest in retrieval task usually exists in the center of the image. They also considered that deep features extracted from the last convolutional layer is more powerful than the first fully connected layer. To this end, they proposed Sum Pooling center prior (SPoC) aggregation. Based on their observation, they suggested that the object in the center of the image contributes more than other regions of the image in the image representation.
Using this observation, they assigned more weight to the features corresponding to the center of the image via a center prior weighting of spatial deep features $\alpha(x,y)$:

$$\alpha(x,y) = \exp\left\{-\frac{(y - \frac{H}{2})^2 + (x - \frac{W}{2})^2}{2\sigma^2}\right\},$$

where $\sigma$ is one third of the distance between the center of a spatial feature location and closest boundary. $H, W$ are the height and width of each of the 2D-slice in the feature maps, respectively. $(x, y)$ are the coordinate of this 2D slice.

Sum Pooling aggregation is performed by summing the value of each coordinate $f(x, y)$ along $x$-axis then $y$-axis:

$$\psi_1 = \sum_{y=1}^{H} \sum_{x=1}^{W} f(x, y) \tag{2.9}$$

The proposed sum pooling center prior (SPoC) aggregation is the product of $\alpha(x,y)$ and $f(x,y)$ for each value of the 2D slice of a spatial location of the feature maps.

$$\psi_2 = \sum_{y=1}^{H} \sum_{x=1}^{W} \alpha(x,y) f(x,y), \tag{2.10}$$

SPoC aggregation, is followed by L2-normalization, PCA, and whitening. Babenko et al. [28] consider the importance of the spatial resolution for the input image. For this reason, they resized the image into $586 \times 586$ before introducing it into the network rather than resizing to $224 \times 224$.

Kalantidis et al. [36] revisited sum pooling aggregation but proposed a different way of weighting, which depends on the maximum activations across the feature maps. They proposed Cross Dimensional Weighting (CroW) through a combination of two feature weighting methods. One method is carried out for each spatial features while the other method operates across the depth channels in the feature tensor. Then they applied sum-pooling as detailed in [28]. Subsequently, L2-normalization and PCA for dimensionality reduction is applied. The output of CroW aggregation is a tensor or feature map. Spatial weighting is achieved by summing all the spatial features of the 3D feature map across all the depth channels. As the result of the aforementioned spatial weighting, is a single weighting matrix $S'$ of dimensionality equals to $H \times W$, which is the height and width of each 2D slice of the feature map, respectively.

$$S' = \sum_{k} c^{(k)}, \tag{2.11}$$
where $c^{(k)}$ is a single 2D feature map at the $k$-depth channel. Spatial weighting matrix $\alpha$ is calculated from $S'$ after normalization and power-scaling. This weighting matrix $S\alpha$ is multiplied by each spatial feature map, then followed by sum aggregation proposed by [28]. L2 normalization and power factor are applied on the aggregated vector. They expect that similar images have similar occurrence rate for a given feature, so discriminative information can be obtained from the sparsity pattern in the feature maps. Based on this expectation they performed sparsity weighting by computing the proportion of non-zero responses in each local spatial feature maps, then divided by the total number of elements in the spatial location. CroW weighting is:

$$x'_{kij} = \alpha_{ij} \beta_k x_{kij},$$  \hspace{1cm} (2.12)$$

where $x'_{kij}$ is weighted tensor at the $(k, i, j)$-th pixel in the new tensor, $\alpha_{ij}$ is a spatial weighting along horizontal and vertical direction, $\beta_k$ is another weight for the $k$-th channel. After re-weighting the tensor producing $x'$, then sum pooling [28] are applied on the weighted features:

$$f_k = \sum_{i=1}^{W} \sum_{j=1}^{H} x'_{kij}.$$  \hspace{1cm} (2.13)$$

More work related to using off-the-shelf models were tested using different aggregation methods. Tolias et al. [40] proposed using regional maximum activation of convolutions (R-MAC). They extracted deep features from the last convolutional layer after pooling, divided the feature maps into regions, and then applied max-pooling on each region. Max-pooling applied on the a region of the image gives a vector representing each region. By summing all the vectors that related to one image, the final image representation for an image is obtained. Region selection has some criteria such that each region is as large as possible and 40% overlapping between consecutive regions.

### 2.2.2 Fine-tuning using Classification Loss

Babenko et al. [41] studied the features extracted at the last CNN layers which they call Neural Codes. They found that using the features as an image descriptor, where the network trained using a large datatset such as ImageNet, is effective for image retrieval, even if this network was trained for classification tasks and the training images are not related to the image retrieval images. They showed that the features in the middle of the fully connected layers (last convolutional layers and the first two fully connected layers) are effective for
retrieval. They proposed fine-tuning these pre-trained network using a similar dataset to the images will be used for retrieval using a classification loss. They also found that compressing these neural codes using PCA provides a compact representation which can solve the memory issue for retrieval with approximately no loss in retrieval performance. This is true for 128-dimensional image descriptor. Also, they showed that training PCA using a dataset different from the training one is effective and can work better than that of performing PCA in the same training dataset. This can help systems to get a very compact representation for large-scale images. To this end, they assembled a large-scale images dataset (landmark dataset) similar to landmark benchmark for retrieval (Oxford Building [42], Paris [43], and Holidays [34] which consider standard datasets for image retrieval) and fine-tuned AlexNet [11] pre-trained network.

Arandjelovic et al. [27] proposed a trainable VLAD-layer for a convolutional neural network name Net-VLAD. After L2-normalization they performed PCA and whitening for dimensionality reduction from 4096 to get the final image representation as a 256-dimensional vector. They perform end-to-end training for the network (i.e., form the input image to the image representation), by extracting deep features from the last convolutional layer, then designing a pooling layer similar to VLAD by getting the residual \( (c_i - c_k) \); see Section 2.1. This pooling layer provides descriptors with a fixed size for image representation. Similar to VLAD, this representation is computed as following:

\[
V(j, k) = \sum_{i=1}^{N} a_k(X_i)(x_i(j) - c_k(j)),
\]

(2.14)

\( a_k(X_i) \) is the membership of the deep-descriptor \( X_i \) to the \( k \)-th visual word, \( k \) is the number of clusters, \( x_i(j) \) is the \( j \)-th dimensional of the \( i \)-th descriptor. The membership \( a_k(X_i) \) is 1 if the centroid \( c_k \) is the closest to the descriptor \( x_i \), or 0 otherwise.

A major difference between Net-VLAD and VLAD is that its computation requires computing a matrix \( V \). Thus, \( V(j, k) \) represents a single element in this matrix. After computing this matrix, \( V \) is L-2 normalized column wise (intra-normalized) to convert it into a vector. After reshaping matrix \( V \) into a vector, Net-VLAD performs an L2-normalization.

In order to train the proposed VLAD layer via backpropagation, the layer’s operation must be differentiable. This condition is satisfied by replacing \( a_k(X_i) \) with:

\[
\tilde{a}_k(X_i) = \frac{e^{-\alpha \|X_i - C_k\|^2}}{\sum_{k'} e^{-\alpha \|X_i - C_{k'}\|^2}}.
\]

(2.15)
Eq. (2.15) can be re-written after expanding the square distance as follows:

\[
\bar{a}_k(X_i) = \frac{e^{w_k^T X_i + b_k}}{\sum_{k'} e^{w_{k'}^T X_i + b_{k'}}}
\]  

(2.16)

where \( w_k = 2ac_k \) and \( b_k = -a\|c_k\|^2 \). Net-VLAD considers \( w_k \) and \( b_k \) as independent variables which are learned with the other network parameters. Finally, the resulting Vlad descriptor is:

\[
V(j, k) = \sum_{i=1}^{N} \frac{e^{w_k^T X_i + b_k}}{\sum_{k'} e^{w_{k'}^T X_i + b_{k'}}} (x_i(j) - x_k(j))
\]  

(2.17)

### 2.2.3 Learning Distance Metric (Embedding)

Radenovic et al. [44] proposed fine-tune a VGG pre-trained network in a fully automated manner using ranking loss. The Siamese Network consists of two branches sharing the same weights. The input to this network is a pair of images. Label is 1 if the pair of images are matching; and 0 if the pairs are non-matching. Training the Siamese network using hard negative and hard positive examples (images) increases the performance. Radenovic et al. [44] used structure from motion (SFM) to generate 3D models useful to generate hard positive and negative data in fully unsupervised manner. They built their aggregation method based on R-MAC [40] by taking the maximum response of the feature maps associated with a patch. Then, they summed vectors representing each patch to generate the image representation \( f(i) \). They performed end-to-end training using a Siamese network to provide an informative image representation using ranking loss. Consequently, images related to one landmark is mapped closer to each other and further from the other images related to other landmarks through a margin. First, they extracted deep features from the last convolutional layer of VGG16 [1], then they performed a global max-pooling for each spatial location of the feature maps and L2 normalization for the image representation vector. Each branch of the Siamese network is a VGG16 model. Contrastive loss is calculated in Eq. (2.18)

\[
L(i, j) = \frac{1}{2} \left( \|Y(i, j)\| \|\bar{f}(i) - \bar{f}(j)\|^2 + (1 - Y(i, j))(\max \{0, \tau - \|\bar{f}(i) - \bar{f}(j)\|\})^2 \right),
\]  

(2.18)

where \( \bar{f}(i), \bar{f}(j) \) are l2-normalization for image representation vector, \((i, j)\) indicate the image representation produced by each branch for each image; \( \tau \) margin, \( Y(i, j) \) label for input image pairs.

Gordo et al. [45] built on the work by Tolias et al. [40], as well. Their method extracts features from the last convolutional layer (using a VGG16 pre-trained network). Then they
divide the feature maps into regions, and applied max-pooling for each region. As mention earlier, R-MAC [40] gives the final image representation by summing every vector associated with a patch of the feature map element-wise. Subsequently, Gordo et al. [45] perform sum-aggregation for the region descriptor vectors. In [40] they used rigid grid: a grid of blobs that consist of spatial and depth regions in the feature maps. Gordo et al. [45] proposed using flexible region extracted using Region Proposal Network (RPN) [46]. They found that all the previous operations are differential (RPN and max-pooling) and also PCA projection can be implemented with a shifting and a fully connected layer (FC), so they proposed fine-tuning using ranking loss. They used Landmark dataset assembled by [41] to perform fine-tuning in two stages. In the first stage, they performed fine-tuning for VGG-16 using classification loss. Then, they re-sized all the images in 224 × 224. In the second stage, they trained a Siamese network using Triplet loss to fine-tune the network from the first stage. Let $I_q$ be a query image with a descriptor $q$, $I^+$ be a relevant image with descriptor $d^+$, and $I^-$ be a non-relevant image with descriptor $d^-$ then the triplet loss used by this work is:

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

(2.19)

where $m$ is the margin. To train the Siamese network, they built a new dataset. To do so, they deleted all the images that intersect with any of retrieval benchmarks (Oxford Building [42], Paris [43], Holidays [34]), deleted the classes which have few images, and cropped images to reduce the effect of clutter in the images.

In this work, we follow the approach of using off-the-shelf network, that is, that we use pre-trained and publicly available deep convolutional neural networks. This is effective since no fine-tuning is performed. Babenko et al. [41] performed fine-tuning using classification loss but this did not show much improvement than using off-the-shelf network. Also fine-tuning using a ranking loss by training a Siamese network improves the performance but this requires intense data annotation, especially for training using hard examples (hard negative and hard positive). Gordo et al. [45] proposed first fine-tuning VGG16 on landmark dataset using classification loss, then fine-tuning using ranking loss. For efficient image retrieval, a low memory footprint is essential. In many works it was considered to have a compact descriptor while preserving a discriminative information. In order to get a compact descriptor, PCA was performed to have a low-dimensional image descriptor. PCA is consider to be expensive in terms of computational complexity, especially when using a large database.
and high-dimensional vectors. In this work, we proposed to avoid using PCA, yet the image descriptor we introduce is compact, preserves discriminative information, and operates using images of any size as an input to the network. We preserve the aspect ratio of the images without changing the dimensionality of the image descriptor.
Chapter 3

Methodology

This section presents the proposed approach for creating a powerful image representation by aggregating deep features acquired from a CNN pre-trained model. First, this section introduces convolutional neural network and their powerful generic features for image retrieval. Then it describes the aggregation method’s steps, and finally, it shows that no expensive post-processing steps are required to get the image representation used for similarity measurement. Framework Overview: Algorithm 1 summarizes the steps of aggregating convolutional features using the proposed max-pooling aggregation as a pseudocode. Also Figure 3.1 shows the steps that this method uses to get the final image representation used for image retrieval.

3.1 Convolutional Neural Networks

A traditional model of a pattern recognition system starts from using hand-crafted feature extractor followed by a classifier. This classifier is usually trained on a specific dataset to classify an instance from different classes. A powerful advantage of CNN is that feature extractors and the classifier are both trained through the back-propagation algorithm [16]. In most of artificial neural networks, images can be pre-processed to reduce image resolution. Image resizing and/or hand-crafted-feature extraction is performed to be introduced to the network. However, raw images can be introduced to the network without any pre-processing steps. After the seminal work of [11], the computer vision community adopted the convolutional neural network as the main tool to solve different visual recognition tasks.
Figure 3.1: Proposed aggregation method using off-the-shelf model. Image passed into the pre-trained model to extracted deep features, discriminative information is obtained by max-pooling aggregation, then global average pooling to provide a 512-dimensional vector, equals to the number of the convolutional filters in the last convolutional layer of the pre-trained model.

3.1.1 Brief Review of Convolutional-Neural-Network Architectures

Convolutional Neural Networks (CNN) are composed of several elements constituted by layers. The first element is a set of several convolutional layers. A convolutional layer is a bank of image filters whose job is to extract features from input images. The set of several convolutional layers, which are connected sequentially, constitute several bank of image filters that extract low- and high-level features; low-level features typically are edges and corners, while high-level features are attributes such as tires or windows. In a CNN, this block of convolutional layers operate effectively as a feature extractor. The second element is a set of fully connected layers. These connected layers effectively apply several affine transformations followed by a non-linear operation. A series of fully connected layers in a CNN effectively works as the core of a classifier component. The last layer in a CNN is a soft-max layer. This layer effectively makes the classifier decision. It operates by calculating the probability of a query instance to be part of a class.
Algorithm 1 Aggregation of Convolutional Features

1: procedure AGGREGATION METHOD
   \( Q \): Query image
   \( I \): Image from the database
2: \textbf{Input:} RGB image
3: extracting deep features (VGG16)
4: \textbf{Output:} \( X \in \mathbb{R}^{(K \times W \times H)} \)
   \( 3D \) tensor of feature maps \( \triangleright \) extracted from last CNN layer
5: \( X' \in \mathbb{R}^{(K \times W' \times H')} = \text{maxpool}(X, w, h, s_w, s_h) \)
   \( w, h \) are the width and height of the max-pooling window
   \( s \) max-pooling stride
6: Global average pooling
   \( \psi = \frac{1}{MN} \sum_{x=1}^{W} \sum_{y=1}^{H} f(x, y) \)
7: L2 normalization = \( \psi / \| \psi \| \)
8: Euclidean distance between \( Q \) and \( I \) \( \triangleright \) similarity measure
9: end procedure

3.1.2 Selecting the Convolutional layer for Image Features

Convolutional neural network consists of multiple convolutional layers, as explained earlier. These convolution-based filters are trained to learn and detect visual patterns. As mentioned earlier, these filters can be seen as trainable feature extractors which can detect a simple pattern (edges, corners) or more complex pattern (a car for example). Razavian et al [37] gave an interpretation of the convolutional filters and their functionality. They showed that the first convolutional layer detected simple patterns like edges and textures. As we move forwards through out the network, the deeper layers detect more complex patterns. Zelier and Fergus [38] visualized the learned convolutional filters in order to get an insight about their functionality. They used their ZF CNN model [38], which consists of 8 layers, 5 convolutional layers, 2 fully connected layers, and a softmax layer. They showed that layer 2 responds to corners and some edges/color conjunction. The third layer detects similar textures and has filters which are more complex and are more invariant to different image nuisances. Layer 4 presents filters that respond more to class-specific attributes. Layers 5 shows the whole object with significant pose variations. Given this analysis, one can conclude then that CNNs learn abstractions. In particular, the level of abstraction increases as
we go forward in the network.

While the previous analysis is informative about CNNs, it shed light onto why CNNs work well for visual recognition tasks. Nevertheless, in image retrieval, the high level task is to obtain similar images from a database. This problem can be cast as that of searching for images that have similar objects. For retrieval, we need to consider features that capture low-level features (e.g., edges or texture), and that are robust to object variations in scale, illumination, and translation, among others. Therefore, in order to get image features that are robust to several variations, we chose to get those image features from the latest convolutional layers. Also, it has been found empirically [28] that using the last convolutional layer as a features extractor outperforms using the fully connected layers.

### 3.1.3 Deep features

This work proposes a compact image representation obtained as the aggregation of deep features extracted from the last convolutional layer. Deep features extracted from the last convolutional layer are shown to be powerful for image retrieval [28, 36, 40]. This is because they generalize well when CNNs are trained on a large dataset such as ImageNet. In this work, the network is considered to be fully convolutional. This is because this work is interested in using the features computed by the convolutional layers of a CNN.

In this work, we used RGB images as inputs to a CNN. For an RGB image, multiple filters are convolved with the image channels. The output is a feature map associated with a convolutional filter as a response of the convolution filter. For an RGB image of size \((W_i \times H_i \times 3)\), a filter of size \((F \times F)\) is convolved with the given image with convolution stride \(z\). The output feature map associated with this convolution has a size of \((W_i - F + 2P)/z + 1\) where \(P\) is the input image padding. For multiple filters of the same parameters (convolutional filter size and stride), multiple feature maps are generated with the same spatial dimension and the depth of the feature maps equal to the number of filters, so feature maps are a 3D tensor. Figure 2.2 shows the feature maps generated from a VGG16 pre-trained model. The number of channels (depth of the feature maps) equals to the number of the convolutional filters in the last convolutional layer.

For a given image \(I\), the output of the pre-trained model is a 3D tensor \(X \in \mathbb{R}^{(K \times W \times H)}\) where \(X\) encodes the features extracted from the last convolutional layer, \(K\) is the number of features maps (channels), and \(W\) and \(H\) are the width and height of the spatial feature
Figure 3.2: Example of a max-pooling operation. The max-pooling window size is $(2 \times 2)$ and the stride is $(2 \times 2)$. With this configuration, the max-pooler only analyzes the four windows shown in the left matrix (i.e., blue, green, pink, and purple). Out of those colored regions, the max-pooling operation returns the maximum value in the colored patch, which is of size $(2 \times 2)$; matrix on the right.

3.1.4 Brief review of Max-pooling layers

A pooling layer in a CNN architecture is in charge of decreasing the dimensionality of the output of a convolutional layer. Pooling layers achieve spatial invariance through reducing the feature maps’ resolution [47]. Consequently, this reduction decreases the memory footprint of the computed feature maps.

This work uses frequently max-pooling layers. A max-pooler can be viewed as a filter whose goal is to analyze one area of the feature map, and select the maximum value of that area. Max-pooling is superior in capturing invariances in contrast with other pooling operations [47]. It provides shift and distortion invariance [48]. Max-pooling is computed by taking the maximum value within a neighborhood in a patch of the spatial feature maps. This patch is called max-pooling window and is represented as a matrix of size $n \times m$. For two consecutive max-pooling windows, the spaces in between them is the stride $s$ of max-pooling. Note that with a certain stride parameters, max-pooling windows can overlap. For a square max-pooling window of size $w \times w$, and stride $s$, overlapping max-pooling is produced when $s < w$. In case of $s = w$, non-overlapping max-pooling is applied. Figure 3.2 shows an
Figure 3.3: General processing pipeline for deep-learning-base aggregation for image retrieval. Extract and aggregate features has three stages: 1) Using pre-trained model as a features extractor; 2) aggregation method; and 3) post processing steps. The output is a compact vector with discriminative information called an image representation.

example of max-pooling. That figure illustrates the following case: window size is $(2 \times 2)$, and the stride size is $s = 2$. Note that the max-pooling operation subsamples a feature map. The input feature map is $(4 \times 4)$ and after max-pooling it becomes $(2 \times 2)$.

The max-pooling operation can achieve rotation invariance. This is because the max-pooling result of two images depicting the same content but one is slightly rotated will produce the same output\[48\].

### 3.2 Proposed Aggregation Method

As discussed earlier, the features computed from deep CNNs are general enough and can work for many visual recognition tasks. Starting from this premise, we use off-the-self pre-trained model to extract features. However, the main task is to aggregate these features so that they become informative features for image retrieval. Figure 3.3 shows the general proposed outline for deep feature aggregation for image retrieval.
3.2.1 Max-Pooling Aggregation

An image contains one or more objects of interest, the remaining content in the image may be considered as clutter. Extracting deep features maximizes the contribution of the objects of interest and minimizes the contribution of clutter. From this perspective, this thesis proposes deep features aggregation by taking the maximum value of a patch of the feature maps associated with an object of interest. In other words, this thesis proposes to aggregate information from the feature maps by using max-pooling operations, since they reduce the dimensions of the feature maps while still giving invariance benefits. Consequently, the reduced feature maps will still preserve rich information for image retrieval. Figures 3.4 illustrates that applying overlapping max-pooling on a 3D feature maps, outputs another 3D tensor with lower spatial dimensionality while still maintaining discriminative information. Instead of dividing the input image into a grid, extracting deep features of each region using a pre-trained CNN model, and then aggregating regional feature maps to obtain a final image representation as in [37]; this work proposes passing the entire image into the pre-trained model, extracting deep features from the last conv. layer, and then dividing spatial features into \( l \) overlapping patches. This thesis denotes each 3D tensor associated with each patch by \( X_l \). The patch is a rectangular region, and has two parameters: height and width. To aggregate deep features using maximum value in a patch, this method uses a max-pooling with window size of the same patch height and width. The patch parameters (height and widths) are parameters of the feature maps spatial resolution. For a 3D tensor of feature maps \( X \in \mathbb{R}^{(K \times W \times H)} \) where \( K \) is the number of spatial features maps (channels). \( W, H \) are the width and height of the spatial feature maps respectively, max-pooling is applied over a patch \( X_l \) with the following parameters:

- max-pooling window size:
  
  \begin{align*}
  w &= W/3 \quad (3.1) \\
  h &= H/2 \quad (3.2)
  \end{align*}

- max-pooling stride
  
  \begin{align*}
  s_h &= h/2 \quad (3.3) \\
  s_w &= w/9 \quad (3.4)
  \end{align*}

\( w \) and \( h \) are the width and height of the max-pooling window, respectively; and \( s_h \) and \( s_w \) are max-pooling stride along the y and x axes, respectively.
Figure 3.4: Proposed max-pooling aggregation method. Applying overlapping max-pooling on a 3D feature maps, outputs another 3D tensor with lower spatial dimensionality while still maintaining discriminative information.

### 3.2.2 Discussion

Max-pooling is a common pooling layer in CNN. It is usually applied with a small window size that include a part of an object in the image. Most of the CNN models include max-pooling with a small window size and a small stride size. In case of AlexNet $w = 3, s = 2$ [11], and for VGG16 $w = 2, s = 2$ [1]. Krizhevsky et al. [11] also observed that using overlapping max-pooling increased the performance. The pooling method proposed in this work also uses overlapping max-pooling windows. What differentiate this work from [11, 1] is that max-pooling window and stride size are large and involves about 35% of the feature map size.

Let $X_k$ be the set of all $W \times H$ spatial location of a feature maps $k \in \{1, .., k\}$. For a spatial location $(x, y)$ in feature maps of dimension $(W \times H)$, overlapping max-pooling of size $(w \times h)$ with stride $s_h, s_w$ are applied. The output of max-pooling is another 3D tensor with the same number of channels but has a different spatial dimension of size: $W' = (W - w)/s_w + 1$, $H' = (H - h)/s_h + 1$. The output 3D tensor has a lower spatial resolution but it still keeps most of the discriminative information. In sum, one can write the following mathematical expression:

$$X' \in \mathbb{R}^{(K \times W' \times H')} = \text{maxpool}(X, w, h, s_w, s_h)$$  \hspace{1cm} (3.5)

where $W'$ and $H'$ are the new spatial dimensions after performing max-pooling.

### 3.2.3 Multi-Resolution Max-Pooling

The objects of interest (e.g., building) in the query image and the relevant images may appears at different location and scales. The approach described in the previous section
describes a simple yet powerful method to reduce the resolution of the 3D map. However, the approach does not consider the variations in scale. This is because a single max-pooling operation is applied to the feature maps by using $N$ overlapping max-pooling windows with different stride and window size on each spatial location $(x, y)$. This results in $N$ different 3D tensors $X'_1, X'_2, \ldots, X'_N$ of different spatial dimension. In sum, the following mathematical expression represents the proposed approach:

$$\{X'_1, X'_2, \ldots, X'_N\} = \text{max-pool}(X_N),$$  \hspace{1cm} (3.6)

where $X'_i$ represent the outputs of each max-pooling operation at the $i$-th resolution. After computing all the max-pooling operations, global average pooling is applied on each of these 3D tensors to get a single vector for every tensor $X'_i$. The overview of multi-scale max-pooling aggregation is illustrated in Figure 3.5. In that figure, the method applies different max-pooling operation on the 3D tensor. For every output of each max-pooling, the system computes a single vector representing the output image. Lastly, the method combines all the obtained vectors to generate a single representation.

### 3.2.4 Global Average Pooling

Global average pooling outputs the spatial average of a feature map after performing overlapping max-pooling of window size $(w \times h)$ and stride $s_h, s_w$ for a given spatial location.
Figure 3.6: Global Average Pooling. Global average pooling takes the average of all the values at a spatial location and outputs a scalar value. For example, a spatial feature (green) turns into a scalar value (green).

Let \( X' \in \mathbb{R}^{(K \times W' \times H')} \) the 3D features tensor of discriminative features. Global average pooling is:

\[
\psi_l = \frac{1}{M} \sum_{x'=1}^{W'} \sum_{y'=1}^{H'} f(x', y'),
\]

where \( f(x', y') \) is the feature map for a given pixel after applying max-pooling; and \( M \) is the number of pixels in a feature map. Figure 3.6 illustrates the global average pooling operation. It takes the average of all the values in a spatial location. The outputs of spatial features (green-left) is a scalar value (green-right).

Global average pooling produces the \( k \)-dimensional vector which is the same dimensionality of the image representation. In sum, the final representation is the concatenation of all the global average poolings:

\[
\psi = \{ \psi_1, \psi_2, ..., \psi_l \}.
\]

Figure 3.7 shows the role of global average pooling in the transferring of the 3D tensor to a low-dimensional vector preserving the discriminative information.

Global average pooling makes the network fully convolutional so an input image of any size can be passed into the network. This is effective because there is no need to resize the image prior to passing it into the network. Using an image of any size preserves the aspect ratio and protects from distortions caused by resizing the image to fit the network input layer requirements. Global average pooling results in a compact image representation of \( k \)-dimension regardless of the input image size. Also, because global average pooling produces
Figure 3.7: Performing global average pooling over the 3D tensor of discriminative feature acquired from max-pooling aggregation results in a vector of dimensionality equals the number of channels in the feature maps.

It is expected that an image has multiple objects or multiple visual patterns. Each max-pooling operation captures information related to a one visual pattern. Consequently, overlapping max-pooling capture information of multiple visual pattern. Pooling the contribution Performing max-pooling over a spatial location returns the maximum response of each feature map at each region. Applying a global max-pooling on a spatial location when the image contains multiple objects results in a maximum response for only one object and loss of the information of the other objects and so reduce the performance. Including the influences of all the object in the image leads to higher performance. To do that, using global average pooling over the maximum response associated with multiple objects (regions) leverages the discriminative information for all the objects in the image. This aligns well with the conclusions in [49]. In addition, Zhou et al. [50] concluded that using a global max pooling allows a system to identify only one part of the image, while global average pooling can find all discriminative parts of the image.

In case of multi-scale max-pooling, each patch is aggregated using max-pooling aggregation, and then global average pooling. The output is a $k$-dimensional vector. For $N$ max-pooling window with different parameters, an $N$ $k$-dimensional vectors $\psi_1, \psi_2, ..., \psi_N$ are produced. Each of these vectors are l-2 normalized, then summed element-size:

$$\psi = \text{sum}\{\psi_1, \psi_2, ..., \psi_l\}. \quad (3.9)$$
Figure 3.5 shows the multi-scale max-pooling aggregation. Using global average pooling gives the same dimensional vector regardless to the max-pooling parameters. In this work, there are $N$ different max-pooling, this gives $N$ different 3D tensors $X'_N$. Applying global average pooling on $X'_N$ gives a vector $ψ_N$. 
Chapter 4

Experiments and Results

This thesis proposes max-pooling aggregation for image retrieval. Max-pooling is effective in capturing discriminative information from the feature maps for image retrieval. Selecting the optimal parameters of max-pooling is essential to have discriminative information which is useful for image retrieval. This chapter studies the effect of max-pooling window size and stride size, as well as measures the retrieval performance using standard benchmarks. For all the presented experiments in this chapter, this work used VGG16 [1] as the pre-trained model to extract features from its last convolutional layer. Since this aggregation method uses max-pooling as an aggregation method and VGG16 model has max-pooling layer after the last convolutional layer, this work also studies the effect of last max-pooling of VGG16 by truncating this max-pooling layer and extracting deep features, then comparing the performance when keeping max-pooling after the last convolutional layer.

4.1 Comparison with the state of art

This thesis evaluates the two general approaches of aggregation methods for image retrieval (i.e., local feature based and deep learning based). For local feature based or SIFT based image retrieval, this thesis implements Fisher [32] and VLAD [35] vectors as a baseline for image retrieval using VLfeat [51] package. To implement the baseline, this thesis clustered SIFT descriptors into 16 clusters using Gaussian Mixture model (GMM). To learn the GMM parameters, Flickr100k distractors [42] are used. In the case of VLAD, the number of clusters is 64. For deep learning based, we compare our proposed aggregation method with three state-of-art methods: NetVLAD [27], sum pooling aggregation
(SPoC) [28], and Cross dimensional weighting (CroW) [36]. For NetVLAD, we used the author’s best pre-trained model, and their implementation code \(^1\) to calculate the image representation. As recommended by the authors, for the Oxford and Paris datasets which are described below, NetVLAD crops all the query images and calculate the image representation as 512-dimensional vector. This work chose this dimensionality in order to match the same dimensionality of the proposed image representation dimensionality. For SPoC aggregation, we implemented the paper and extracted deep features from the last convolutional layer before max-pooling layer (layer conv5-3) using MatCaffe \(^2\) package. All images were resized to 586 × 586 and used full query images for retrieval. For CroW aggregation, we used the authors’ implementation code \(^2\) to evaluate the three datasets; see next section for a full description of this datasets. To evaluate Oxford and Holidays dataset, PCA and whitening were trained on Paris dataset. We trained PCA and whitening parameters for Paris dataset using Oxford dataset. Due to computational constraints, PCA for Holidays dataset was trained on Paris dataset instead of Oxford100K, as it was originally proposed by Kalantidis et al. [36]. All deep features are extracted using VGG16 trained by Simonyan and Zisserman [1]. In this thesis, all features are extracted using MatConvnet \(^3\) package unless mentioned otherwise. Max-pooling aggregation with large window and stride was implemented using MatConvnet \(^3\) and Matlab. All these experiments were implemented using an Nvidia Titan X GPU with memory of 12GB; an Intel Core i7-5820K CPU 3.30GHz and 12 cores; and 32GB of RAM. In this section, the terms positive and database are used to express the number of retrieved images. Positive means that the number of retrieved images equals to the number of relevant images to a query image for a given dataset. Database means when the system retrieves all the images in the database and ranks them according to their similarity to the query image.

4.1.1 Datasets

This thesis evaluates the performance of the proposed max-pooling and other related aggregation methods for image retrieval using three standard datasets, which are considered as the benchmark for image retrieval. Oxford Building dataset [42], contains 5062 images of Oxford landmarks, 55 queries of 11 different landmarks, so each landmark has 5 different

\(^1\)NetVLAD: We got the code and downloaded their best model from: http://www.di.ens.fr/willow/research/netvlad/

\(^2\)CroW: We got the code from: https://github.com/yahoo/crow
queries. The ground-truth for each query is divided into good (relevant images), okay images (relevant images), and junk images (non-relevant images). The performance is evaluated with the mean average precision (mAP); see next section for a description of this measure. Oxford dataset suggests a bounding box for each query, so the query should be cropped before searching for relevant image.

Paris [43] dataset contains 6412 images collected from Flickr\(^3\) by searching for particular Paris landmarks. It also provides 55 queries of 11 different landmarks. The ground-truth for each is divided into good, okay and junk. Each query has a bounding box so that the system can crop the object of interest as in Oxford dataset.

The third dataset used in this work is the Holidays [34] dataset. It contains 1491 images of 500 different groups. Each group has a query image and one or more relevant images. Some images have wrong orientations. This work fixes these images by manually rotating them. To evaluate the performance in mAP, the Holidays dataset has an evaluation code so the query is removed from the dataset when retrieving images, unlike Oxford and Paris datasets where query images are not removed when retrieve images.

Table 4.1 shows a summary of the three datasets. Each row corresponds to a dataset. The columns show information about number of images, number of landmarks and query images in a dataset. Also, the fifth column shows the evaluation protocol recommended for each dataset and the evaluation metric which is mAP for all the three datasets.

---

\(^3\)https://www.flickr.com/
4.1.2 Evaluation metric

For image retrieval experiments on Oxford, Paris and Holidays, we evaluate the performance using mean average precision (mAP). mAP measures retrieval quality and evaluates ranking. We calculated mAP for Oxford and Paris dataset using the evaluation code\(^4\) provided by oxford dataset. For Holidays, mAP is measured using Holidays’ dataset evaluation code\(^5\).

mAP stands for the mean average precision. The mAP measure can be calculated from Eq.4.1

\[
mAP = \frac{1}{N} \sum_{N=1}^{N} AP, \tag{4.1}
\]

where \(N\) is number of queries. The AP (average precision) can be calculated by the combination of Precision (P) and Recall (R):

\[
P = \frac{\{\text{Relevant Samples}\} \cap \{\text{Retrieved Samples}\}}{\{\text{Relevant Samples}\}}, \tag{4.2}
\]

\[
R = \frac{\{\text{Relevant Samples}\} \cap \{\text{Retrieved Samples}\}}{\{\text{Retrieved Samples}\}}, \tag{4.3}
\]

\[
AP = \frac{1}{k} \sum_{k=1}^{k} P(k) \Delta R(k), \tag{4.4}
\]

where \(k\) is the number of retrieved images, \(\Delta R(k)\) is the change in recall.

4.2 Experiments

In this section we present a series of experiments that illustrate the effect of max-pooling parameters and evaluate the retrieval performance. In section 4.2.1, we study the effect of max-pooling window size on image retrieval performance. Section 4.2.2 studies the effect of max-pooling stride size. Also, the experiments in section 4.2.1 and section 4.2.2 are repeated when truncating max-pooling after the last conv. layer in section 4.2.3. In section 4.2.4, we study the effect of multi-scale max-pooling which detect objects of any potential size.

\(^4\)http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/
\(^5\)http://lear.inrialpes.fr/~jegou/data.php
4.2.1 Max-pooling window size

We evaluate the effect of max-pooling window size on image retrieval performance. In this experiment, we fix the stride size to \( s_h = s_w = h/4 \) and change max-pooling window size. Max-pooling window size has two parameters: width \( w \) and height \( h \). To tune window size, we calculate \( w \) and \( h \) as a parameter of the width \( W \) and height \( H \) of the a spatial feature map correspond to an input image.

\[
\begin{align*}
    w &= W/a \quad (4.5) \\
    h &= H/b \quad (4.6)
\end{align*}
\]

In all the experiments, we consider a square window so \( a = b \). We started with the maximum window size possible (i.e., \( a = 1 \) and \( b = 1 \) as in [37]) and then decreased the window size by increasing \( a \) and \( b \) by 1.

![Figure 4.1: Effect of max-pooling window size when keeping max-pooling layer after the last convolutional layer. Blue curve is the performance for database retrieval. The performance decreases by decreasing max-pooling window size.](image)

Table 4.2 shows the retrieval evaluation on Oxford dataset when changing the window size. We noticed that performance decreases by decreasing max-pooling window size. In Table 4.2 first column is max-pooling window size. We tuned this parameter while we fixing the stride size to \( h/4 \). Second column in the same Table is the max-pooling stride size. The
Table 4.2: Effect on the window size of the max pooler when keeping the last max-pooling layer. Performance decreases by decreasing max-pooling window size. Also, overlapping between consecutive max-pooling is important. About quarter of the max-pooling windows are overlapped. Stride is fixed to tune the window size.

<table>
<thead>
<tr>
<th>window size</th>
<th>stride</th>
<th>positive</th>
<th>database</th>
</tr>
</thead>
<tbody>
<tr>
<td>h=H/1, w=W/1</td>
<td>h/4</td>
<td>0.5062</td>
<td>0.5796</td>
</tr>
<tr>
<td>h=H/2, w=W/2</td>
<td>h/4</td>
<td>0.5756</td>
<td>0.6383</td>
</tr>
<tr>
<td>h=H/3, w=W/3</td>
<td>h/4</td>
<td>0.569</td>
<td>0.6315</td>
</tr>
<tr>
<td>h=H/4, w=W/4</td>
<td>h/4</td>
<td>0.5543</td>
<td>0.6217</td>
</tr>
<tr>
<td>h=H/5, w=W/5</td>
<td>h/4</td>
<td>0.55543</td>
<td>0.6217</td>
</tr>
<tr>
<td>h=H/6, w=W/6</td>
<td>h/4</td>
<td>0.5381</td>
<td>0.613</td>
</tr>
<tr>
<td>h=H/7, w=W/7</td>
<td>h/4</td>
<td>0.55375</td>
<td>0.6133</td>
</tr>
<tr>
<td>h=H/8, w=W/8</td>
<td>h/4</td>
<td>0.5313</td>
<td>0.6056</td>
</tr>
<tr>
<td>h=H/9, w=W/9</td>
<td>h/4</td>
<td>0.5312</td>
<td>0.6057</td>
</tr>
</tbody>
</table>

Positive column shows the retrieval performance when the number of retrieved images equals to the number of relevant images to a query image for a given dataset. In a similar fashion, the database column presents the retrieval performance when the system retrieves all the images in the database and ranks them according to their similarity to the query image. The effect of max-pooling window size on image retrieval performance on Oxford dataset is shown in Figure 4.1. The Figure 4.1 shows that changing max-pooling window size affects the retrieval performance. The performance decreases by decreasing max-pooling window. At the same time global max-pooling over the entire spatial max-pooling decreases the performance since global max-pooling returns the response for one visual pattern associated with the maximum value in the spatial feature map.

4.2.2 Max-pooling stride size

To evaluate max-pooling stride size parameters, we used the best results obtained from tuning max-pooling window size \((w = W/2, h = H/2)\) in table 4.2, and then changed the
Figure 4.2: Effect of max-pooling stride size when keeping max-pooling layer after the last convolutional layer. The stride achieves a maximum performance at $h/3$ and then the performance stays flat.

Max-pooling stride has two parameters $s_h, s_w$ stride on x-axis and y-axis, respectively. We set $s_h, s_w$ as parameters of the max-pooling window in the following manner:

$$s_h = h/c$$

$$s_w = w/d.$$  \hfill (4.7) \hfill (4.8)

In all the experiments, we chose $s_h = s_w$ and start from the maximum stride possible which satisfy $c = d = 1$, and then decreased the stride by factor of 1. Table 4.3 shows the stride effect of retrieval performance on Oxford dataset. First column is the max-pooling window size which is set to $h = H/2$, $w = H/2$. Second column contains different stride settings. This Table also compares the retrieval performance between positive and database in columns 3 and 4. Figure 4.2 shows the performance of max-pooling aggregation when changing the stride size. The performance increases by decreasing the stride size but remains the same at $s_h = h/3$, $s_w = w/3$. Note that when using the entire database to evaluate performance, the performance measure increases significantly in comparison when we use only the positive set for each query; compare the positive and database column in Table 4.3.
Table 4.3: Effect on the stride of the max-pooler when keeping max-pooling at the last convolutional layer.

<table>
<thead>
<tr>
<th>window size</th>
<th>stride</th>
<th>positive</th>
<th>database</th>
</tr>
</thead>
<tbody>
<tr>
<td>h=H/2, w=W/2</td>
<td>h/1</td>
<td>0.4812</td>
<td>0.5502</td>
</tr>
<tr>
<td>h=H/2, w=W/2</td>
<td>h/2</td>
<td>0.5469</td>
<td>0.6128</td>
</tr>
<tr>
<td>h=H/2, w=W/2</td>
<td>h/3</td>
<td>0.5767</td>
<td>0.6545</td>
</tr>
<tr>
<td>h=H/2, w=W/2</td>
<td>h/4</td>
<td>0.5756</td>
<td>0.6383</td>
</tr>
<tr>
<td>h=H/2, w=W/2</td>
<td>h/5</td>
<td>0.5754</td>
<td>0.6381</td>
</tr>
<tr>
<td>h=H/2, w=W/2</td>
<td>h/6</td>
<td>0.5769</td>
<td>0.6421</td>
</tr>
<tr>
<td>h=H/2, w=W/2</td>
<td>h/7</td>
<td>0.577</td>
<td>0.6421</td>
</tr>
<tr>
<td>h=H/2, w=W/2</td>
<td>h/8,h/9,h/10</td>
<td>0.5745</td>
<td>0.6373</td>
</tr>
<tr>
<td>h=H/2, w=W/2</td>
<td>h/11</td>
<td>0.5737</td>
<td>0.6368</td>
</tr>
<tr>
<td>h=H/2, w=W/2</td>
<td>h/12</td>
<td>0.5737</td>
<td>0.6354</td>
</tr>
</tbody>
</table>

4.2.3 Truncate max-pooling layer of the last convolutional layer of VGG16

Since VGG model has a max-pooling layer after the last convolutional layer and the proposed aggregation method uses max-pooling for aggregation, we studied the effect of original max-pooling after the last convolutional layer on retrieval performance. Earlier in sections 4.2.1 and 4.2.2, deep features are extracted when keeping the last max-pooling layer (layer pool5) as in [36, 40]. In this section, we present the result of studying the effect of only max-pooling aggregation by truncating max-pooling layer in VGG architecture and extract deep features from layer (conv5-2) like in [28]. We repeated the previous experiments to see max-pooling window size effect and stride size on aggregation. Since max-pooling performs a spatial subsampling, the spatial feature dimensional has a higher dimensionality. Specifically, twice the height and width of the spatial dimensionality when extracting features at layer (pool15). Table 4.4 shows the results of varying the dimensionality of the max-pooling window while keeping the stride parameter, which was set to h/4. Features for this
Table 4.4: Effect on the window size of the max pooler when truncating the last max-pooling layer

<table>
<thead>
<tr>
<th>window size</th>
<th>stride</th>
<th>positive</th>
<th>database</th>
</tr>
</thead>
<tbody>
<tr>
<td>h=H/1, w=W/1</td>
<td>h/4</td>
<td>0.5062</td>
<td>0.5796</td>
</tr>
<tr>
<td>h=H/2, w=W/2</td>
<td>h/4</td>
<td>0.5756</td>
<td>0.6375</td>
</tr>
<tr>
<td>h=H/3, w=W/3</td>
<td>h/4</td>
<td>0.5692</td>
<td>0.6322</td>
</tr>
<tr>
<td>h=H/4, w=W/4</td>
<td>h/4</td>
<td>0.5505</td>
<td>0.6223</td>
</tr>
<tr>
<td>h=H/5, w=W/5</td>
<td>h/4</td>
<td>0.544</td>
<td>0.6199</td>
</tr>
<tr>
<td>h=H/6, w=W/6</td>
<td>h/4</td>
<td>0.5449</td>
<td>0.6177</td>
</tr>
<tr>
<td>h=H/7, w=W/7</td>
<td>h/4</td>
<td>0.5332</td>
<td>0.6098</td>
</tr>
<tr>
<td>h=H/8, w=W/8</td>
<td>h/4</td>
<td>0.5324</td>
<td>0.6075</td>
</tr>
<tr>
<td>h=H/9, w=W/9</td>
<td>h/4</td>
<td>0.5296</td>
<td>0.605</td>
</tr>
<tr>
<td>h=H/10, w=W/10</td>
<td>h/4</td>
<td>0.527</td>
<td>0.603</td>
</tr>
<tr>
<td>h=H/11, w=W/11</td>
<td>h/4</td>
<td>0.5206</td>
<td>0.5982</td>
</tr>
<tr>
<td>h=H/12, w=W/12</td>
<td>h/4</td>
<td>0.523</td>
<td>0.5987</td>
</tr>
</tbody>
</table>

Experiments were extracted from the last convolutional layer ($\text{conv5-2}$). We evaluated image retrieval on Oxford in terms of the mAP when retrieving positive and database. Also, Figure 4.3 shows the effect of window size on Oxford dataset image retrieval when truncating max-pooling. Note that the performance decreased by decreasing the window size. Moreover, the comparison between positive column (red curve) and database column (blue curve) shows that the database always outperforms the positive retrieval.

The results of the stride size effect experiment are shown in Table 4.5. The features for this experiment were extracted from layer ($\text{conv5-2}$) of VGG16. We set the window size to the best result obtained from window-size-effect experiment, and tuned the stride size. Retrieval evaluation is performed on Oxford dataset in terms of the mAP when retrieving positive and database. Figure 4.4 shows that the performance increases by decreasing the stride size but remains the same after $h/3$. Tables 4.4 and 4.5, and Figures 4.3 and 4.4 show
that truncating original max-pooling from VGG architecture does not make any improvement with extra cost since truncating max-pooling increases the dimensionality of the spatial feature maps than keeping it.

Figure 4.3: Effect of max-pooling window size when truncating max-pooling layer after the last convolutional layer.

Figure 4.4: Effect of max-pooling stride size when truncating max-pooling layer after the last convolutional layer.
Table 4.5: Truncate max-pooling from the last CNN layer. Stride size effect

<table>
<thead>
<tr>
<th>window size</th>
<th>stride</th>
<th>positive</th>
<th>database</th>
</tr>
</thead>
<tbody>
<tr>
<td>h=H/2, w=W/2</td>
<td>h/1</td>
<td>0.4575</td>
<td>0.5276</td>
</tr>
<tr>
<td>h=H/2, w=W/2</td>
<td>h/2</td>
<td>0.5407</td>
<td>0.609</td>
</tr>
<tr>
<td>h=H/2, w=W/2</td>
<td>h/3</td>
<td>0.5799</td>
<td>0.6346</td>
</tr>
<tr>
<td>h=H/2, w=W/2</td>
<td>h/4</td>
<td>0.5756</td>
<td>0.6375</td>
</tr>
<tr>
<td>h=H/2, w=W/2</td>
<td>h/5</td>
<td>0.5797</td>
<td>0.6432</td>
</tr>
<tr>
<td>h=H/2, w=W/2</td>
<td>h/6</td>
<td>0.5775</td>
<td>0.642</td>
</tr>
<tr>
<td>h=H/2, w=W/2</td>
<td>h/7</td>
<td>0.5688</td>
<td>0.6358</td>
</tr>
<tr>
<td>h=H/2, w=W/2</td>
<td>h/8</td>
<td>0.572</td>
<td>0.6346</td>
</tr>
<tr>
<td>h=H/2, w=W/2</td>
<td>h/9</td>
<td>0.5719</td>
<td>0.6345</td>
</tr>
<tr>
<td>h=H/2, w=W/2</td>
<td>h/10</td>
<td>0.5719</td>
<td>0.6345</td>
</tr>
<tr>
<td>h=H/2, w=W/2</td>
<td>h/11</td>
<td>0.5751</td>
<td>0.6402</td>
</tr>
<tr>
<td>h=H/2, w=W/2</td>
<td>h/12</td>
<td>0.5753</td>
<td>0.6399</td>
</tr>
</tbody>
</table>

4.2.4 Multi-scale max-pooling

We previously studied the effect of a max-pooling with a large window size and stride. Instead of using one max-pooling with a large window size, we test using multiple max-pooling aggregations with different window scales, i.e., different window sizes. To do that, we fixed the stride to $s_h = s_w = h/4$, and then tried different sets of max-pooling with different window’s scales. Similar to section 4.2.1, we considered a square window and started with a maximum window size. Then, we gradually decreased the window size and applied different scales of max-pooling windows. Table 4.8 shows the different settings of max-pooling window size used in this experiment. Each row represents an experiment with the number of different max-pooling used, and the parameters of each max-pooling window. We tested 9 different max-poolings. We enumerate each experiment in the range of 1 and 12. Performing a max-pooling on a 3D tensor results in another 3D tensor. To get a vector
Table 4.6: Multi-scale max-pooling aggregation. Window size effect when summing the image representation associated with each max-pooling aggregation vector. Oxford dataset

<table>
<thead>
<tr>
<th>window size</th>
<th>stride</th>
<th>positive</th>
<th>database</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>h/4</td>
<td>0.5676</td>
<td>0.6346</td>
</tr>
<tr>
<td>2</td>
<td>h/4</td>
<td>0.5536</td>
<td>0.6252</td>
</tr>
<tr>
<td>3</td>
<td>h/4</td>
<td>0.5713</td>
<td>0.6373</td>
</tr>
<tr>
<td>4</td>
<td>h/4</td>
<td>0.5755</td>
<td>0.6395</td>
</tr>
<tr>
<td>5</td>
<td>h/4</td>
<td>0.5717</td>
<td>0.6399</td>
</tr>
<tr>
<td>6</td>
<td>h/4</td>
<td>0.5727</td>
<td>0.64</td>
</tr>
<tr>
<td>7</td>
<td>h/4</td>
<td>0.5701</td>
<td>0.6397</td>
</tr>
<tr>
<td>8</td>
<td>h/4</td>
<td>0.5709</td>
<td>0.6395</td>
</tr>
<tr>
<td>9</td>
<td>h/4</td>
<td>0.5658</td>
<td>0.6328</td>
</tr>
<tr>
<td>10</td>
<td>h/4</td>
<td>0.5708</td>
<td>0.6368</td>
</tr>
<tr>
<td>11</td>
<td>h/4</td>
<td>0.5429</td>
<td>0.6157</td>
</tr>
<tr>
<td>12</td>
<td>h/4</td>
<td>0.5381</td>
<td>0.6142</td>
</tr>
</tbody>
</table>

out of 3D tensor, global average pooling is used. For \( N \) different max-poolings, applying global average pooling results in \( N \) vectors.

To get the final image representation as one vector, we tested two approaches: 1) summation of all vectors element-wise, and 2) concatenating all vectors. We compared these two approaches when testing max-pooling window size. Table 4.6 shows the retrieval performance on Oxford positive and database when summation approach is used. The first column in Table 4.6 shows the experiments with window size parameters which can be found in Table 4.8. The second column is the stride size, and the third and fourth columns are the results in terms of the mAP on Oxford dataset using positive and database sets, as in previous experiments. Testing multiple parameters shows that multi-scale max-pooling aggregation is more robust to tuning than a single max-pooling aggregation and does no require tuning.
Table 4.7: Multi-scale max-pooling aggregation. Window size effect when concatenation the image representation associated with each max-pooling aggregation vector. Oxford dataset

<table>
<thead>
<tr>
<th>window size</th>
<th>stride</th>
<th>positive</th>
<th>database</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>h/4</td>
<td>0.5417</td>
<td>0.6346</td>
</tr>
<tr>
<td>2</td>
<td>h/4</td>
<td>0.5509</td>
<td>0.6208</td>
</tr>
<tr>
<td>3</td>
<td>h/4</td>
<td>0.5568</td>
<td>0.6373</td>
</tr>
<tr>
<td>4</td>
<td>h/4</td>
<td>0.556</td>
<td>0.6395</td>
</tr>
<tr>
<td>5</td>
<td>h/4</td>
<td>0.5558</td>
<td>0.6399</td>
</tr>
<tr>
<td>6</td>
<td>h/4</td>
<td>0.5577</td>
<td>0.64</td>
</tr>
<tr>
<td>7</td>
<td>h/4</td>
<td>0.5584</td>
<td>0.6397</td>
</tr>
<tr>
<td>8</td>
<td>h/4</td>
<td>0.5591</td>
<td>0.6395</td>
</tr>
<tr>
<td>9</td>
<td>h/4</td>
<td>0.542</td>
<td>0.6328</td>
</tr>
<tr>
<td>10</td>
<td>h/4</td>
<td>0.5745</td>
<td>0.6368</td>
</tr>
<tr>
<td>11</td>
<td>h/4</td>
<td>0.522</td>
<td>0.6157</td>
</tr>
<tr>
<td>12</td>
<td>h/4</td>
<td>0.5386</td>
<td>0.6142</td>
</tr>
</tbody>
</table>

steps. This is because the mAP does not vary much as we change the parameters.

Also, Table 4.7 shows the concatenation effect for multi-scale max-pooling when Oxford positive and database are evaluated using mAP. Figures 4.5 and 4.6 compare summation and concatenation on Oxford dataset when retrieving database and positive consequently. We can see in those figures that summation outperforms concatenation and tends to be more robust to tuning.
Figure 4.5: Comparing between multi-scale max-pooling aggregation when summing and concatenating all \( N \) vectors produced by \( N \) max-pooling. Evaluation when retrieving all the images in the database. Summation is more robust and outperforms concatenation in case of database retrieval.

Figure 4.6: Comparing between multi-scale max-pooling aggregation when summing and concatenating all \( N \) vectors produced by \( N \) max-pooling. Evaluation when retrieving positive images for queries. Summation is more robust and outperforms concatenation in case of positive retrieval.
Table 4.8: Multi-scale max-pooling aggregation. Window size effect when concatenation the image representation associated with each max-pooling aggregation vector. Oxford dataset

<table>
<thead>
<tr>
<th>Exp.</th>
<th>max-pooling 1</th>
<th>max-pooling 2</th>
<th>max-pooling 3</th>
<th>max-pooling 4</th>
<th>max-pooling 5</th>
<th>max-pooling 6</th>
<th>max-pooling 7</th>
<th>max-pooling 8</th>
<th>max-pooling 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>H/1, W/1</td>
<td>H/2, W/2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>H/1, W/1</td>
<td>H/2, W/2</td>
<td>H/3, W/3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>H/1, W/1</td>
<td>H/2, W/2</td>
<td>H/3, W/3</td>
<td>H/4, W/4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>H/1, W/1</td>
<td>H/2, W/2</td>
<td>H/3, W/3</td>
<td>H/4, W/4</td>
<td>H/5, W/5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>H/1, W/1</td>
<td>H/2, W/2</td>
<td>H/3, W/3</td>
<td>H/4, W/4</td>
<td>H/5, W/5</td>
<td>H/6, W/6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>H/1, W/1</td>
<td>H/2, W/2</td>
<td>H/3, W/3</td>
<td>H/4, W/4</td>
<td>H/5, W/5</td>
<td>H/6, W/6</td>
<td>H/7, W/7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>H/1, W/1</td>
<td>H/2, W/2</td>
<td>H/3, W/3</td>
<td>H/4, W/4</td>
<td>H/5, W/5</td>
<td>H/6, W/6</td>
<td>H/7, W/7</td>
<td>H/8, W/8</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>H/1, W/1</td>
<td>H/2, W/2</td>
<td>H/3, W/3</td>
<td>H/4, W/4</td>
<td>H/5, W/5</td>
<td>H/6, W/6</td>
<td>H/7, W/7</td>
<td>H/8, W/8</td>
<td>H/9, W/9</td>
</tr>
<tr>
<td>9</td>
<td>H/1, W/1</td>
<td></td>
<td>H/3, W/3</td>
<td></td>
<td>H/5, W/5</td>
<td></td>
<td>H/7, W/7</td>
<td></td>
<td>H/9, W/9</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td>H/4, W/4</td>
<td></td>
<td>H/6, W/6</td>
<td></td>
<td>H/8, W/8</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>H/1, W/1</td>
<td></td>
<td></td>
<td></td>
<td>H/5, W/5</td>
<td></td>
<td></td>
<td></td>
<td>H/9, W/9</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>H/5, W/5</td>
<td>H/6, W/6</td>
<td>H/7, W/7</td>
<td>H/8, W/8</td>
</tr>
</tbody>
</table>
4.3 Comparison with the state-of-the-art

This section presents an evaluation of the performance of the proposed aggregation methods and compares it with that of the state-of-art. The evaluation metric is the mean average precision (mAP). Also, we considered state-of-the-art methods which are similar to the proposed approach. These methods also use off-the-shelf pre-trained networks, extract deep features, aggregate the features, and then apply post-processing techniques to produce a final image representation. All deep learning experiments that we compared with use PCA and whitening as post-processing steps. PCA and whitening is beneficial to have a lower dimensionality and a better performance in some cases (e.g., in case of evaluating Oxford, when PCA is trained on Paris, the performance improves). While PCA shows retrieval improvements for many methods (see below), it is computationally expensive. This is because PCA has to calculate a covariance matrix using a training set with high-dimensional vectors, which can require high-memory footprints and heavy computations. This computational drawback makes PCA not practical for large image databases. Therefore, we avoid any expensive post-processing steps such as PCA. We report all our proposed max-pooling aggregation methods without PCA and whitening.

In this section, we evaluate max-pooling aggregation and multi-scale max-pooling aggregation when keeping max-pooling after the last convolutional layer; and compare it with local feature based image retrieval methods based on Fisher’s vector and Vector of Locally Aggregated Descriptors (VLAD). We considered the following deep-learning-based image retrieval competing methods for this experiment: NetVLAD [27], SPoC [28], and CroW [36].

We repeated all the previous experiments considering that the number of retrieved images equals to the number of relevant images for each query except for Holidays dataset, in which the number of retrieved images is 11 always, regardless of the number of relevant images of a query image. We called this method **Positive**. Retrieving only positive examples gives a better understanding about the system’s ability to retrieve relevant images accurately. For this reason, we calculated the mAP for all the competing methods retrieving only positive examples.

The table 4.9 shows the results of comparing the retrieval performance of the proposed max-pooling and multi-scale max-pooling with that of the state-of-art, competing methods. The first column in the table shows the considered methods which are grouped based of the aggregation approach (local features based and deep learning based). The second column
(dim.) is the dimensionality of the image representation, and columns \{3, 4\} show the parameters of the proposed methods. Specifically, column 3 shows max-pooling window size where \(h, w\) are the max-pooling height and width, respectively. On the other hand, \(H\) and \(W\) are the spatial feature maps’ height and width, respectively. The column 4 shows the max-pooling stride size. In the same column, \(s_h\) and \(s_w\) are the max-pooling strides over the height and the width, respectively. Columns \{5, 6, 7\} show the evaluation of each aggregation method on mAP for Oxford[42], Paris [43] and Holidays [34] dataset, respectively. Recall that the number of retrieved images equals to the number of positive images for each query.

For local features aggregation based methods, we considered Fisher and VLAD vectors as the baseline. We can see that in the case of the deep learning based methods, SPoC [28] uses a low-dimensional image representation with high performance for Oxford and Holidays dataset but it does not generalize, and does not perform well for Paris dataset. Also, we can see that CroW [36] is the current-state-of-art for off-the-shelf image retrieval. It outperforms all the previous experiments because of their cross-dimensional weighting for the spatial features which encourages response of the features associated with the object of interests. Nevertheless, it uses PCA. It trained PCA carefully to improve the performance. This is seen in case of Oxford dataset, which has been trained in Paris dataset, and it has been observed that training PCA of Oxford dataset on Paris dataset improves the performance. When we evaluated Oxford using CroW aggregation and learned PCA parameters on Oxford dataset the performance was mAP=0.5353 which is about 9% lower than the case of training PCA of Oxford dataset on Paris dataset (mAP=0.6279). Also, empirically, training PCA for Holidays dataset on Oxford105K dataset improves the performance than training PCA on Paris dataset. For NetVLAD [27], the network is expected to perform well since it was fine-tuned for image retrieval rather than using off-the-shelf VGG model. Although fine-tuning was performed for Net-VLAD, other competing off-the-shelf aggregation methods outperformed Net-VLAD.

In Table 4.9, row 7 shows the results of the proposed max-pooling aggregation. This method performs well for all the three datasets. For Oxford dataset it outperformed all the other methods except the CroW aggregation method since they fine-tuned the method using PCA. For the Paris dataset set it slightly outperformed the current-state-of-art without using PCA, but it had a lower performance in the case of Holidays dataset. This is because Holidays images are larger than Oxford and Paris dataset (about two times larger), and the tuning parameters of max-pooling window and stride size are not well-tuned for Holidays’ image
size. We followed the pre-processing stages methods of the state-of-art and resized Holidays dataset images in half. Nevertheless, we did so by, keeping the aspect ratio. The performance increased as shown in row 9 in Table 4.9. The last row in the table shows the results for the multi-scale max-pooling aggregation. The used window size for multiple max-pooling were $h_1 = H/2, w_1 = W/2, h_2 = H/4, w_2 = W/4, h_3 = H/6, w_3 = W/6, h_4 = H/8, w_4 = W/8$. Multi-scale max-pooling has lower performance than a single max-pooling, However, tuning max-pooling window size for multi-scale parameter is expected to perform well and may outperform a single max-pooling aggregation.

Finally, Figures 4.7 and 4.8 compare the proposed methods in this thesis. All the four experiments have the same maximum performance. Multi-scale max-pooling aggregation, when summing the $N$ vectors that produced from $N$ different max-pooling by applying global average pooling, is more robust to spatial resolution changes. Thus, it is more robust to the change in the image size. Figures 4.10 and 4.11 show retrieval examples using the proposed single max-pooling-based retrieval system on Oxford dataset when retrieving top 10 images. First image to the left is the query images. Query images are cropped according to the bounding box defined by the dataset. The red box show the mistakenly retrieved examples. In case of Paris dataset, max-pooling aggregation correctly retrieved top 10 images for all queries.
Figure 4.7: Comparing the four tested methods of max-pooling aggregation on Oxford dataset. Retrieving positive only. Single max-pooling aggregation when keeping max-pooling after last CNN layer (green), truncating max-pooling (purple), mult-scale max-pooling aggregation by summing all vectors of $N$ max-poolings (blue), concatenation $N$ vector (red). While all the four method have the same maximum performance, mult-scale max-pooling when summing $N$ vectors is more robust to spatial resolution changes.
Figure 4.8: Comparing the four tested methods of max-pooling aggregation on Oxford dataset. Retrieving all images in the dataset (i.e., database). Single max-pooling aggregation when keeping max-pooling after last CNN layer (green), truncating max-pooling (purple), multiscale max-pooling aggregation by summing all vectors of $N$ max-poolings (blue), concatenation $N$ vector (red). While all the four method have the same maximum performance, multi-scale max-pooling when summing $N$ vectors is more robust to spatial resolution changes.
Figure 4.9: Visualization of the image representation of a query image, relevant image and non-relevant image which was mistakenly retrieved as a relevant image. Similar visual patterns appears in image representations, which confirms that max-pooling preserving discriminative information. White cell correspond to the maximum value and black cells correspond to minimum values in the image representation.
Figure 4.10: Retrieval of top 10 images for the query associated to a landmark. Query images are cropped in cyan bounding box defined by the Oxford dataset. Red box show false positive images that were mistakenly retrieve as relevant image.

Figure 4.11: Retrieval of top 10 images for the query associated to a landmark. Query images are cropped in cyan bounding box defined by the Oxford dataset. Red box show false positive images that were mistakenly retrieve as relevant image.
Table 4.9: Experiments.

<table>
<thead>
<tr>
<th>method</th>
<th>dim.</th>
<th>window size</th>
<th>stride</th>
<th>Oxford</th>
<th>Paris</th>
<th>Holidays</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local Feature Based</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fisher vector [32]</td>
<td>16384</td>
<td>N/A</td>
<td>N/A</td>
<td>0.1414</td>
<td>0.1016</td>
<td>0.4294</td>
</tr>
<tr>
<td>VLAD [35]</td>
<td>2048</td>
<td>N/A</td>
<td>N/A</td>
<td>0.1616</td>
<td>0.1225</td>
<td>0.39844</td>
</tr>
<tr>
<td>Deep Learning Based</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPoC [28]</td>
<td>256</td>
<td>N/A</td>
<td>N/A</td>
<td>0.5683</td>
<td>0.4797</td>
<td>0.7703</td>
</tr>
<tr>
<td>CroW [36]</td>
<td>512</td>
<td>N/A</td>
<td>N/A</td>
<td>0.6279</td>
<td>0.6642</td>
<td>0.8208</td>
</tr>
<tr>
<td>NetVLAD [27]</td>
<td>256</td>
<td>N/A</td>
<td>N/A</td>
<td>0.5365</td>
<td>0.6113</td>
<td>0.802</td>
</tr>
<tr>
<td>NetVLAD [27]</td>
<td>512</td>
<td>N/A</td>
<td>N/A</td>
<td>0.5919</td>
<td>0.6208</td>
<td>0.807</td>
</tr>
<tr>
<td>Proposed Approach</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max-pooling</td>
<td>512</td>
<td>h=H/2, w=W/3</td>
<td>s_h = h/3, s_w = w/9</td>
<td>0.5962</td>
<td>0.6662</td>
<td>0.7748</td>
</tr>
<tr>
<td>Max-pooling image resized to half size</td>
<td>512</td>
<td>h=H/2, w=W/3</td>
<td>s_h = h/3, s_w = w/9</td>
<td>N/A</td>
<td>N/A</td>
<td>0.8082</td>
</tr>
<tr>
<td>Multi-max-pooling</td>
<td>512</td>
<td>see below</td>
<td>h/4</td>
<td>0.5708</td>
<td>0.6366</td>
<td>0.7843</td>
</tr>
</tbody>
</table>

multi-scale max-pooling window size

max-pooling 1: \( h_1 = H/2, w_1 = W/2 \)
max-pooling 2: \( h_2 = H/4, w_2 = W/4 \)
max-pooling 3: \( h_3 = H/6, w_3 = W/6 \)
max-pooling 4: \( h_4 = H/8, w_4 = W/8 \)
Chapter 5

Conclusion

Computing an informative image representation for image retrieval is a challenging task. This thesis not only tackled that problem, but also added the challenge of computing such a representation efficiently. While several previous approaches focused on computing an informative image representation for image retrieval, they left aside the computational complexity of the post-processing stages that help an image retrieval system to improve its performance. This thesis focused on reducing the computational complexity of the post-processing stages while still using an informative image representation.

Unlike the local-based image representations, such as, Fisher [32] and VLAD [35] vectors, modern approaches obtain remarkable retrieval performance using convolutional neural networks (CNN). However, many recent approaches need post-processing techniques to obtain a comparable or better retrieval performance. These post-processing techniques include computing the principal component analysis (PCA), whitening, and normalization, among other procedures. These post-processing techniques help systems to reduce the dimensionality of the image representation, and normalize the representations so that systems can compare them more effectively.

While the post-processing techniques improve the image representation for retrieval purposes, they typically require extra time and computational resources. For instance, PCA requires the computation of a covariance matrix to identify the principal components in a dataset. This computation can be expensive since it depends on the amount of data points as well as the dimensionality of the original representations.

To address the extra computational cost in the image retrieval representation pipelines, this thesis investigated methods based solely on common operations used by CNNs. In
particular, the hypothesis of this work is that max-pooling operations are efficient operations that can be used in a post-processing phase with the goal of reducing the dimensionality of image representations while maintaining rich information in an image representation for image retrieval purposes.

A max-pooler selects a region in a feature map or image and returns the maximum value of such a region. Typically, a system applies a max-pooler to the entire domain of the feature map. As a result of this max-pooling operations, the system computes another feature map with reduced dimensions on both width and height. Thus, max-poolers can reduce the “dimensionality” of the feature maps, which are key pieces to compute an image representation for image retrieval as discussed in Chapter 3. This was the main motivation behind this work. It is worth noticing that many publicly available deep learning libraries provide efficient implementations of max-pooling, which help the proposed method implement an efficient post-processing stage.

This work also assumed that pre-trained CNNs for visual recognition can provide useful information to compute a compact image representation for image retrieval. In particular, this thesis investigated two approaches using max-pooling operations as the main components of a post-processing stage. The first approach investigated the use of a single max-pooler that can be configured to produce a feature map of a certain dimensionality. The second method investigated the use of a bank of max-poolers that can be configured to produce different feature maps that later are combined to produce a single image representation. The output of both approaches are simple feature maps that later are passed to a phase where a single feature map is reduced to a single number.

In sum, both approaches operate as follows. First, a query image is passed through a pre-trained CNN to obtain the set of feature maps, which typically is the output of the last convolutional layer in the network. Then, the set of feature maps is processed by a set of max-poolers; this can be a single max-pooler or multiple ones with different configurations. Subsequently, the system summarizes the output of the max-pooler-stage, which is another set of feature maps with reduced dimensions. To this end, the system applies global average pooling, which computes the average of every feature map in the set. The global average pooling thus maps each feature map to a single number. Consequently, the dimensionality of the proposed image representation equals to the number of feature maps produced by the last convolutional layer.

In Chapter 4, the thesis presents experimental results assessing the impact of the param-
eters and performance compared with state-of-the-art image retrieval systems. In particular, this work implemented the aforementioned approach and devised a series of experiments to understand the effect in retrieval performance of the max-pooler’s parameters. In addition, this work measured the overall retrieval performance, and compared it with the state-of-the-art image retrieval systems (e.g., Net-VLAD [27]). For these experiments, this work used three publicly available and widely adopted for evaluating image retrieval systems. These datasets include Oxford-Buildings [42], Paris [43], and Holidays [34].

The experiment evaluating the impact of the parameters into the final retrieval performance showed the following conclusion: the best max-pooling parameters peak the highest retrieval performance and can be obtained as a function of the dimensions of a feature map (i.e., width and height). The experiments also showed that for a single max-pooling mechanism, the performance decreases when the dimensions of the width and height of the region considered by the max-pooler decreases. Thus, the best point, according to the experiments, indicates that the best region for a max-pooler is of medium size, i.e., not too large and not small with respect of the dimensions of the feature map.

A similar experiment evaluating the impact of the parameters into the final retrieval performance using a multi-max pooling layer showed a similar effect of that of the single max pooling layer. However, the multi-max pooling method tends to be more robust to a non-optimal parameter configuration. This is because the decrease in retrieval performance was minor in comparison with the single max-pooling-based method. Nevertheless, the number of parameters increases given that the multi-max pooling approach uses different max-poolers to operate.

The last experiment measured the image retrieval performance using the standard benchmark, which consists of the aforementioned datasets. In this experiment, the work compared the proposed approach with local-based image representations (e.g., Fisher [32] and VLAD [35] vectors) and competing state-of-the-art deep-learning-based retrieval methods (e.g., Net-VLAD [27], SPoC [28], and CroW [36]. The experiment showed that the proposed single- and multi-max-pooling approach performs comparable than the competing state-of-the-art deep-learning-based retrieval methods. Unlike the competing deep-learning-based methods, the proposed approach does not need an expensive post-processing stage. This experiment thus confirm the hypothesis of the thesis. In other words, expensive post-processing stages can be replaced by simpler and efficient ones based on max-pooling and global average pooling operations. This new post-processing stage leverages the efficient implementations
that several publicly available deep-learning libraries offer, and do not require a secondary training phase in order to compute PCA.

5.1 Future directions

The presented experiments in this thesis demonstrate that max-pooling operations and global-average-pooling can be used to devise an efficient aggregation layer. However, the thesis built on pre-trained CNNs, which were trained specifically for visual recognition tasks. Consequently, it is still possible to boost the retrieval performance considering a training phase of a CNN dedicated to image retrieval tasks.

In addition, it is also possible to optimize automatically for the max-pooling parameters in the proposed aggregation layer within the same training phase of a CNN. Nevertheless, this involves a more detailed and careful implementation and architecture.

Finally, a similarity or distance function can be learned in order to improve the retrieval performance. All the previous future directions can be combined in a training phase of CNN-training phase. In sum, the max-pooling parameters and a similarity or distance function can be learned jointly when training a CNN specifically for image retrieval tasks.
References


REFERENCES


Aggregating Deep Features For Image Retrieval

Mohammad Eiad Kazkaz

Thesis submitted to the
Benjamin M. Statler College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering

Lane Department of Computer Science and Electrical Engineering

APPROVAL OF THE EXAMINING COMMITTEE

__________________________________________
Nasser M. Nasrabadi, Ph.D.

__________________________________________
Xin Li, Ph.D.

__________________________________________
Victor Fragoso, Ph.D., Chair