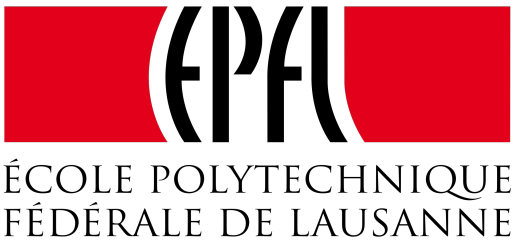

École polytechnique fédérale de Lausanne

IVRL

Semester Project - Final Report

Content Based Web Image Retrieval

Author:
Yuliang Zheng
yuliang.zheng@epfl.ch

Supervisor
Bin Jin

2016 Spring

Yuliang Zheng
yuliang.zheng@epfl.ch

Abstract

With the prevalence of images online, finding the related or intended images becomes more and more important for users. For image retrieval, normally we can do it in two ways, using a few keywords or instantly a sample picture as the query. In other words, there are two frameworks: text-based and content-based.

The goal of this semester project is to accomplish the similar effects as Google's image-based engine, that is to say, implementing Content-based Image Retriecal. Utilizing a few different methods, and even implemented some tricks such as binary encoding, the author tried to find the one offering the best accuracy.

To fulfil the CBIR(Content-based Image Retrieval) implemention, the basic idea is to build the feature matrix for all the images, and then use the query image as the baseline to rank the rest images in the dataset. Based on the ranking result, the top one will be the best match.

Contents

1 Introduction **3**
1.1 Two Basic Image Retrieval Types . . . 3
1.2 Goals of the Project . . . 3
1.3 Overview of the Report . . . 4

2 Background **4**
2.1 Algorithm Pipelines . . . 4
2.2 Descriptors . . . 5
2.2.1 SIFT . . . 5
2.3 PCA . . . 6
2.4 Encoding . . . 6
2.4.1 BOV . . . 6
2.4.2 Fisher Vector . . . 7
2.4.3 Vector Normalization . . . 8
2.4.4 Binary Encoding . . . 8
2.4.5 CNNs . . . 9
2.5 Ranking . . . 10
2.5.1 Euclidean Distance . . . 10
2.5.2 Hamming Distance . . . 10

3 Experiments **11**
3.1 Datasets and Experimental Set-up . . . 11
3.1.1 Datasets . . . 11
3.1.2 Experimental Set-up . . . 11
3.2 Evaluation Methods . . . 12
3.3 Experiments with BOV . . . 13
3.4 Experiments with Fisher Vector . . . 14
3.4.1 A Simple Implementation . . . 14
3.4.2 Fisher Vector Normalization . . . 14
3.4.3 Fisher Vector Binarization . . . 15
3.5 Experiments with CNNs . . . 16

4 Conclusions **17**

Yuliang Zheng
yuliang.zheng@epfl.ch

1 Introduction

1.1 Two Basic Image Retrieval Types

With the development of Internet, and the availability of image capturing devices such as the digital cameras, the size of digital image collection is increasing rapidally. Efficient image browsing or retrieval are widely used in various domains, including entertainment, fashion, medicine, etc. There are basically two types of image retrieval: TBIR(Text-based Image Retrieval) and CBIR(Content-based Image Retrieval).

Utilizing a keyword to retrieve the target images on a search engine such as Google, is the traditional way. As the text-based approach can be tracked back to 1970s. On the Internet, each image is manually annotated by text descriptors(i.e. each image has a few keyword labels to describe it), which are then used by the database management system(DBMS) to perform image retrieval.

However, the TBIR has some deficiencies. The first is that a considerable level of human labour is required for manual annotation. The second is the annotation inaccuracy due to the subjectivity of human perception [1]. While the image itself contains a lot of useful information, using the method of CBIR, where images are indexed by their visual content, such as color, texture, shapes, will offer better accuracy compared to TBIR.

1.2 Goals of the Project

The goal of this semester project is to accomplish the similar effects as Google's image-based engine, and explore a few CBIR technologies to find out the one with best accuracy. Figure 1 is an example of CBIR using the Google search engine. In this example, the author input a sample picture names "*300.jpg*" as the query, the search engine analized this picture as type *shore*, and offered a bunch of similar images. To fulfil the CBIR, we need basically two steps: extracting the proper feature matrix for images, and doing a nice ranking for the images based on the feature matrix. What's more, a reliable evaluation is needed to help select the best plan.

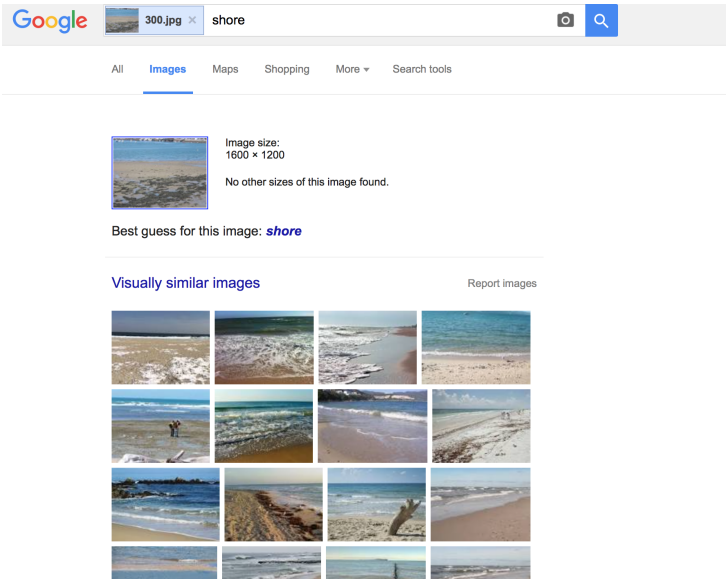

Figure 1: Google's image-based engine

1.3 Overview of the Report

The rest of this report is organized as follows. Section 2 briefly introduces all the theories or technologies utilized in the semester project, and the algorithm pipeline is explained as well. Section 3 introduces all the related datasets and the adopted performance measurement as well as the experimental results, while some analyses are provided. Section 4 concludes this semester project, or this report.

2 Background

In this section, the project's algorithm is explained. And a brief description to all the related theories or technologies used are given, in order to help the readers to have a simple but overall understanding of this semester project.

2.1 Algorithm Pipelines

The basic idea for doing Content-based Image Retrieval in this semester project, is to fetch a bunch of images first, build the reasonable feature matrix, and rank the images based on the query image. To select the best plan, reliable evaluation methods are necessary. Figure 2 shows the general structure of the project's implementation.

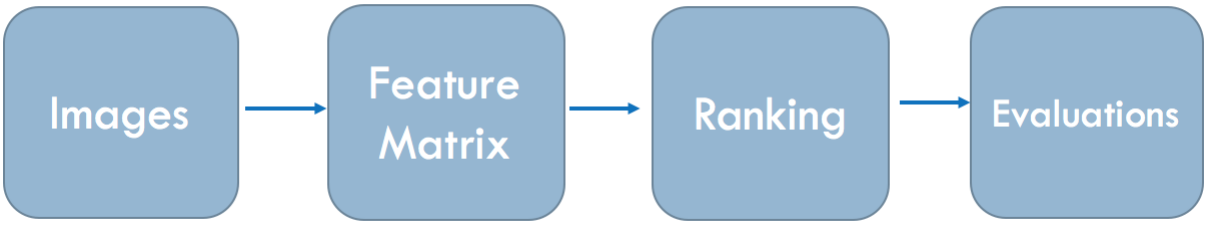

Figure 2: General Structure

To build the feature matrix, the author utilized three methods: BOV(*bag-of-visual-words*), Fisher Vector, and CNNs(*Convolutional Neural Networks*). Before doing BOV or Fisher Vector encoding, we need to extract descriptors from each image. SIFT descriptors were chosen to build the descriptor matrix of each image. So, each image was represented by a SIFT descriptor matrix of size 128 $\times$ n, where n is the number of interest points we chose. However, CNNs only needs the images to be the input, as the author used *VGG16* model, each image was represented by a vector of dimension 4096.

The extracted SIFT descriptor matrices are in huge size, so it's impossible to do rankings based on them. The idea of the project is to use a column vector instead of a big matrix to represent one image, which makes the ranking step easier. BOV can encode the descriptor matrix based on a codebook to do k-means, while Fisher Vector can do so by anthor clustering method, GMM(Gaussian Mixture Model).

As the SIFT descriptor matrix is 128D, the author did a PCA to reduce its dimension from 128D to 64D. However, the results showed PCA only improved Fisher Vector's accuracy, but not for BOV. So, the author did a PCA before doing the Fisher Vector encoding, which is shown in Figure 3.

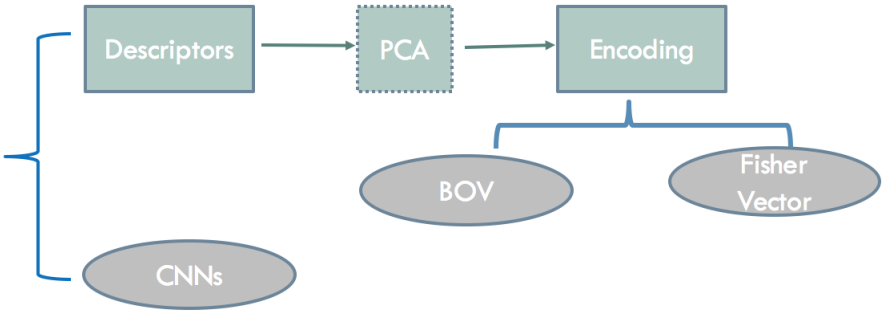

Figure 3: Feature Matrix Extraction

For the part of Fisher Vector, besides a simple implementation, the author also did "Fisher Vector Normalization" to increase accuracy. As Fisher Vectors are always highly-dimensional and dense, computing the Euclidean distances to a bunch of images takes a long time. In order to increase speed, the author tried with two binarization methods of Fisher Vector. The "$\alpha = 0$ normalization" and a more complex SH encoding, the details of the two methods can be found in the rest of this section. It is noteworthy that the author used the normalized Fisher vectors with the best α to do the SH encoding.

The essense of ranking is to do a similarity measure. Here, the author chose the simple Euclidean distance to compute the similarity, while Hamming distance for the binarized Fisher Vectors. Three reliable evaluation methods were used to select the best plan, and their implementation details are contained in the next section.

The rest of this section will provide the details of the theories or technologies mentioned above.

2.2 Descriptors

In computer vision, visual descriptors or image descriptors are descriptions of the visual features of the contents in images, videos, or algorithms or applications that produce such descriptions. They describe elementary characteristics such as the shape, the color, the texture or the motion, among others. [2]

2.2.1 SIFT

Scale-invariant feature transform (or **SIFT**) is an algorithm in computer vision to detect and describe local features in images. SIFT was presented in 1999 by David Lowe and includes both a keypoint detector and descriptor.

SIFT descriptors are computed as follows [5]:

1. Detect keypoints using the SIFT detector, which also detects scale and orientation of the keypoint.
2. For a given keypoint, warp the region around it to canonical orientation and scale and resize the region to 16×16 pixels.
3. Compute the gradients for each pixels (orientation and magnitude).
4. Divide the pixels into 16, 4×4 pixels squares.
5. For each square, compute gradient direction histogram over 8 directions.
6. concatenate the histograms to obtain a 128 (16*8) dimensional feature vector.

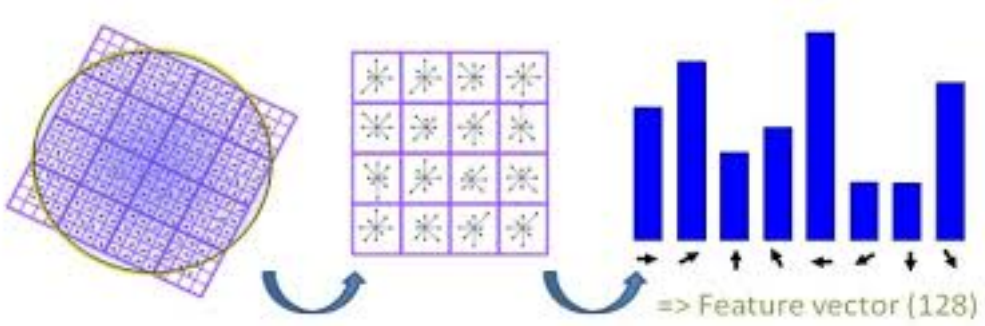

Figure 4: SIFT Descriptor Illustration [5]

2.3 PCA

PCA, or **Principle Component Analysis** is a widely used method for dimesionality reduction. If doing SVD(Singular Value Decomposition) to an m × n matrix **X**, we have:

$$X = USV^T \tag{1}$$

Where U is an m × n matrix, S is an n × n matrix, and V^T is also an n × n matrix. And V is a matrix of eigenvectors (each column is an eigenvector), S is a diagonal matrix with eigenvalues λ_i in the decreasing order on the diagonal. So, the Principal components are given by:

$$XV = USV^TV = US \tag{2}$$

If we want to reduce the matrix dimensionality from m × n to p × n (p < m), we only need to select the first p rows of XV.

As the author chose SIFT to compute the initial feature matrix(descriptor matrix), each image will be represented by a matrix with dimensionality 128 × n, where n is depending on how many useful prixes are selected in that image. If the extracting image contains a lot of information(e.g. a high-definition picture with many different faces), the resulting descriptor matrix must be in a big size. Doing a dimensionality reduction might be helpful to compress the matrix without losing important contents of the image. So, PCA definitely is one of the best choices to fulfil the dimensionality reduction of the descriptor matrix.

2.4 Encoding

In this semester project, after successfully extracting the descriptor matrix to represent one image, the image was further encoded into a vector. Here, the author tried a few methods, in order to select the best one providing the highest accuracy.

2.4.1 BOV

One important method used in BOV is clustering(e.g. K-means). The basic steps to implement BOV are as follows. "Interest points are detected in the image and local invariant descriptors are extracted. Each descriptor is assigned to its closest visual word in a 'visual vocabulary': a codebook obtained offline by clustering a large set of descriptors with K-means. This results

in a typically highly-dimensional sparse histogram representation." [7] And then represent the image by a frequency vector, which is the final BOV description of that image.

Figure 5 offers an intuitive explanation of how BOV works. There are three different images, one contains a woman, while the other two are a bike and a violin. Besides, a codebook with plenty of "visual words" is already existing. So, after extracting the corresponding descriptors of each image, the codebook is used to compute the hitstogram representation for the images. And after sorting, each image can be represented by a frequency vector.

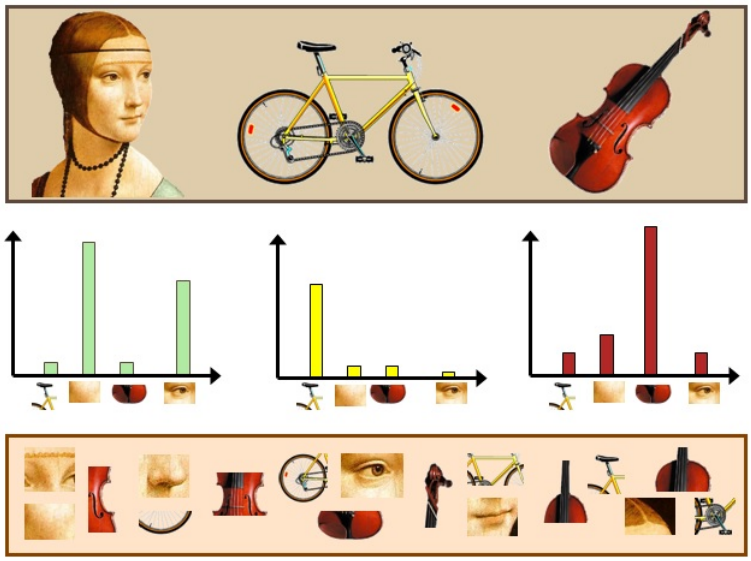

Figure 5: BOV: *representing object as histograms of words occurrences* [6]

2.4.2 Fisher Vector

The Fisher Vector is an image representation obtained by pooling local image features. It is frequently used as a global image descriptor in visual classification. While the FV can be derived as a special, approximate, and improved case of the general Fisher Kernel framework, it is easy to describe directly. [8]

X is the descriptor matrix(here, SIFT descriptors) of one image, where $X = (x_1, \cdots, x_N)$, a set of D dimensional feature vectors. Let $\Theta = (\mu_k, \Sigma_k, \pi_k : k = 1, \cdots, K)$ be the parameters of a **Gaussian Mixture Model(GMM)** fitting the distribution of descriptors. The GMM associates each vector x_i to a mode k in the mixture with a strength given by the posterior probability. [8]

$$q_{ik} = \frac{exp[-\frac{1}{2}(x_i - \mu_k)^T \Sigma_k^{-1}(x_i - \mu_k)]}{\sum_{t=1}^{K} exp[-\frac{1}{2}(x_i - \mu_t)^T \Sigma_k^{-1}(x_i - \mu_t)]} \tag{3}$$

For each mode k, the mean and covariance deviation vectors are as follows:

$$\mu_{tk} = \frac{1}{N\sqrt{\pi_k}} \sum_{i=1}^{N} q_{ik} \frac{x_{ti} - \mu_{tk}}{\sigma_{tk}} \tag{4}$$

$$\nu_{tk} = \frac{1}{N\sqrt{2\pi_k}} \sum_{i=1}^{N} q_{ik} \left[\left(\frac{x_{ti} - \mu_{tk}}{\sigma_{tk}} \right)^2 - 1 \right] \tag{5}$$

where $t = 1, \cdots, D$ spans the vector dimensions.

To do the **Fisher Vector Encoding** to the image of descriptor matrix X, we only need to extract all the mean vectors and covariance deviation vectors from the GMM results, that is:

$$\Phi(X) = [\cdots \mu_k \cdots \nu_k \cdots]^T \tag{6}$$

Here, $\Phi(X)$ is the Fisher Vector of the image. Since the image's descriptor matrix is of size D $\times$ N, while the GMM's cluster number is K, the resulting Fisher Vector is of size 2DK $\times$ 1. Obviously, **Fisher Vector Encoding** contributes a lot in the compression of feature matrix.

2.4.3 Vector Normalization

Using the Fisher Vectors to do the rankings, is definitely feasible. However, if discounting the influence of large values in the vector before the L_2 normalization, we can getter higher accuracy. One simple way to achieve this is by raising the Fisher Vector's each dimension to the power of a α, where $\alpha \in [0, 1]$.

If $Y = (y_1, \cdots, y_M)$ is a Fisher Vector of dimension M $\times$ 1. The basic algorithm to do the normalization is shown below:

1. For $i = (1, \cdots, M)$ apply power normalization

$$y_i = sign(y_i) \sqrt[\alpha]{|y_i|}$$

2. Apply the l_2 - normalization

$$Y = Y / \sqrt{YY^T}$$

2.4.4 Binary Encoding

Since the size of Fisher Vector is mainly depended on the dimension of the descriptors **D** as well as the visual vocabulary size **K** (i.e. the cluster number in GMM), only if one of the two numbers becomes large, the Fisher Vector will be high-dimensional. As a result, doing further compression, such as binary enoding, is a useful method to ameliorate this deficiency.

Here, the author utilized two methods to binarize the Fisher Vector.

$\alpha = 0$ binarization Recalling the previous section of **Vector Normalization**, if α goes to zero, y_i will converge to -1 if $y_i < 0$, 0 if $y_i = 0$, 1 if $y_i > 0$. That is to say:

When $\alpha \to 0$,

$$y_i = \begin{cases} -1 & \text{if } y_i < 0 \\ 0 & \text{if } y_i = 0 \\ 1 & \text{if } y_i > 0 \end{cases} \tag{7}$$

Hence, the $\alpha = 0$ **binarization** is one kind of encoding methods, which makes the Fisher Vector to converge to a ternary representation.

Spectral Hashing(SH) Instead of using a ternary representation, the SH algorithm finds another binary encoding method. SH is a binary encoding way such that points which are far apart in the original Eclidean space are also far apart in the Hamming space and vice-versa [7].

Assuming that the datapoints $x_i \in R^d$ are samples from a probability distribution $p(x)$. If $p(x)$ is seperable, and similarity between datapoints is defined as $e^{-||x_i - x_j||^2/\varepsilon^2}$,then the eigenfunctions of the continuous weighted Laplacian, L_p have an outer product form [9]. Here, $\Phi_i(x)$ is an eigenfunction of the weighted Laplacian defined on R^1 with eigenvalue λ_i.

All the eigenfunctions are sinusoidal functions and can be divided into "single-dimension" eigenfunctions and "outer-product" eigenfunctions [10]. All single-dimension eigenfunctions are of the form:

$$\Phi_k(x(i)) = \sin(\pi/2 + \frac{k\pi}{b_i - a_i} x(i)) \tag{8}$$

$$\lambda_k = e^{\frac{-\sigma^2}{2}\left|\frac{k\pi}{b_i - a_i}\right|^2} \tag{9}$$

Here, $x(i)$ refers to the ith coordinate of x and $x(i)$ is uniformly distributed in $[a_i, b_i]$.

So, if we are given the training set $\{x_i\}$ and the desired encoding bit k, the **SH** algorithm can be implemented as follows: [9]

1. Finding the principal components of the data using PCA.
2. Calculating the k smallest single-dimension analytical eigenfunctions of L_p using a rectangular approximation along every PCA direction. This is done by evaluating the k smallest eigenvalues for each direction using (equation 8), thus creating a list of dk eigenvalues, and then sorting this list to find the k smallest eigenvalues.
3.Thresholding the analytical eigenfunctions at zero, to obtain binary codes.

2.4.5 CNNs

Learning effective feature representations and similarity measures are crucial to the retrieval performance of a content-based image retrieval (CBIR) system [14]. Besides to design sophisticated low-level feature extraction algorithms, the well-known "semantic gap" issue wins more and more concerns around the world. So, reducing the "semantic gap" between low-level image pixels captured by machines and high-level semantic concepts perceived by human, is important.

In this semester project, the author examined a state-of-the-art deep learning method **CNNs (Convolutional Neural Networks)**, hoping to increase accuracy and reducing the "semantic gap" as well.

Unlike conventional machine learning methods that are often using "shallow" architectures, deep learning mimics the human brain that is organized in a deep architecture and processes information through multiple stages of transformation and representation [14]. Here, the author ulitized the VGG16 model, which its detailed information can be found at [12].

The CNNs architecture is as follows. The image is passed through a stack of convolutional (conv.) layers, where we use filters with a very small receptive field: 3×3 (which is the smallest size to capture the notion of left/right, up/down, center). A stack of convolutional layers (which has a different depth in different architectures) is followed by three Fully-Connected (FC) layers. [13]

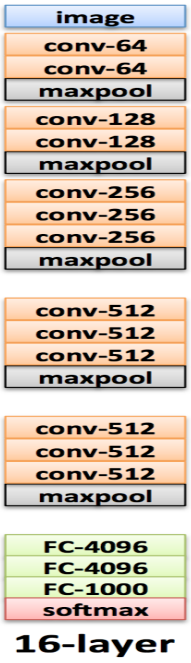

Figure 6: VGG16: Layer Configuration [12]

Figure 6 is the layer configuration of the VGG 16 model. It has 16 weight layers. In this project, instead of using the final layer's output, the author used the penultimate layer's output. The *FC-4096* layer gives 4096 outputs. So, each inputting image now is represented by a vector of dimension 4096.

2.5 Ranking

The nature of ranking is to compute the similarity between two vectors. For similarity comparison of two vectors, we can use sereval metrics.

2.5.1 Euclidean Distance

The essence of **Euclidean Distance** is to compute the l_2 norm of the difference of two vectors.

Given two vectors $X = (x_1, x_2, ..., x_n)$ and $Y = (y_1, y_2, ..., y_n)$, the distance is computed as:

$$d(X, Y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2} \tag{10}$$

2.5.2 Hamming Distance

Assuming $X = (x_1, x_2, ..., x_n)$ and $Y = (y_1, y_2, ..., y_n)$ are two vectors, the **Hamming Distance** *d(X, Y)* is the number of different coefficients in which the two verctors differ. The following is an example.

$$d(00011, 11001) = 3 \tag{11}$$

3 Experiments

Here, instead of using a web crawler to fetch a bunch of images from the Internet, the author used the publicly available dataset as well as publicly available feature descriptors to do the experiments. And in this section, the author will give a detailed exhibition of all the experiments step by step as well as the corresponding results.

3.1 Datasets and Experimental Set-up

3.1.1 Datasets

The different datasets used in this semester project are presented in the following.

Holidays *(1491 images, 4.456M descriptors, 500 queries)* This is a dataset which mainly contains personal holiday photos. The dataset includes a very large variety of scene types(natural, water, etc). In this dataset, there are 500 image groups. The dataset is available at [11].

Flickr60k *(67714 images, 140M descriptors)* This is a dataset with 67714 arbitrarily-retrieved images from Flickr. As the size of *Holidays* dataset is small, using another large-sized image dataset with various types will offer better codebooks. In this semester project, Flickr60k is used to learn PCA, the cluster centers of k-means as well as GMM. The dataset is available at [11].

3.1.2 Experimental Set-up

Extraction of descriptors To begin with, we need to transform the visual image into a mathematical matrix to do the experiments. Here, the author extracted the SIFT descriptors of each image in the *Holidays* dataset to do the analysis. Each image in the *Holidays* dataset is then represented by a descriptor matrix of size $128 \times n$, where n is the number of interest points selected. These descriptor matrices are further used in BOV and Fisher vector to do further encoding.

PCA BOV and Fisher Vector are two methods using clustering technologies. However, one scalability problem clustering methods have to deal with is dimensionality: in general high-dimensional data is too sparsely distributed in its high-dimensional space in order to identify meaningful clusters. Therefore the data is typically projected into a lower dimensional space.

So, in this semester project, the author also reduced dimensionality of each descriptor matrix from 128D to 64D, in order to increase accuracy. And the PCA was done on the base of vocabularies trained in Fluckr60k.

Ranking Methods *Euclidean distance* was used to compute the distance between the query image and the rest in the Holidays dataset if no binarization method was implemented. And then the author sorted all the images based on the resulting Euclidean distance. However, in the case of using SH to do binarization, the author replaced the *Hamming distance* with the *Euclidean distance*.

3.2 Evaluation Methods

Three evaluation standards are used to compute the **MAP** *Mean Average Precision* to compare different methods, in order to select the one with best accuracy. As the Holidays dataset [11] already classifies the 1491 images into 500 groups, we can use the first image of each group as the query to do the retrieval. And each group's entries except the query one, are called the "*Perfect Match*" of that query.

Method 1 The first one is a special method to compute each query's **AP** *Average Precision*, and then to get the **MAP**(mean of all the AP). Its corresponding algorithm is shown below:

```
sum_of_ap = 0
For # groups:
    np = # Perfect Match of this group
    nr = # retrieved items of this group
    ap = 0
    For # retrieved items in this group:
        ns = # Perfect Match so far
        r = rank of this retrieved item
        if this item ∈ Perfect Match:
            if r == 0: precision_0 = 1
            else: precision_0 = ns/nr
            precision_1 = (ns + 1)/(nr + 1)
            ap += (precision_0 + precision_1)/2
    sum_of_ap = ap
MAP = sum_of_ap/ # groups
```

Here, symbol "#" means "the number of", "*ap*" means "average precision", "*MAP*" means "mean average precision".

Attention: Being different from the previous one, in the following two evaluation methods, each group's *Perfect Match* includes the query itself.

Method 2: The Last Position This is an intuitive way to compute the **MAP**, and it works in the following way:

```
sum_of_ap = 0
For # groups:
    np = # Perfect Match of this group
    ap = 0
    For # retrieved items in this group:
        m is the item ∈ Perfect Match, with the largest ranking number(i.e. the last Perfect
Match entry being retrieved)
        position = rank of m
        ap += np/position
    sum_of_ap = ap
MAP = sum_of_ap/ # groups
```

Method 3: TOP 10 This is also a simple and intuitive way to compute the **MAP**, and it works in the following way:

```
sum_of_ap = 0
For # groups:
    ap = 0
    For # retrieved items in this group:
        num is the number of Perfect Match in the first 10 retrieved items
        ap += num/10
    sum_of_ap = ap
MAP = sum_of_ap/ # groups
```

3.3 Experiments with BOV

Here, the author built two codebooks to do the k-means clustering. One is using the Holidays dataset itself to compute the vocabularies, the other one is using the Flickr60k which has a much larger size of images. What's more, different cluster numbers($\mathbf{K}$ = 8, 64, 256) were tried, and PCA was added as well.

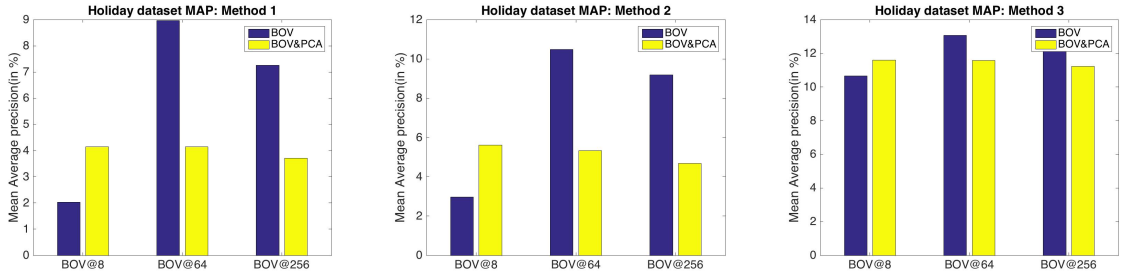

Figure 7: BOV: using the FIickr60k to compute vocabularies

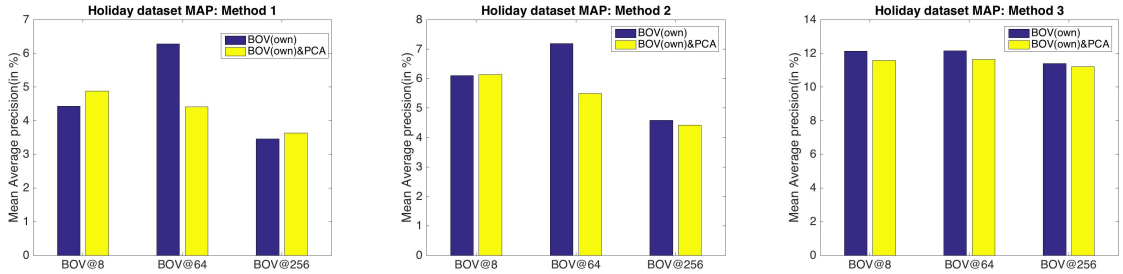

Figure 8: BOV: using the Holidays dataset itself to compute vocabularies

Figure 7 shows the result of using the Flickr60k to build the codebook, while Figure 8 is of using the Holidays dataset itselft to compute the vocabularies. And three ways of evaluation were all utilized. When setting the number of cluster to be 64 and without doing PCA, we got the highest MAP. The best accuracy in the case of using the codebook of the Flickr60k is 8.967%(The Special Method), 10.485%(The Last Position), 13.060%(TOP 10), when the cluster number is 64 and without PCA. The best accuracy in the case of using the codebook of the Holidays dataset is 6.280%(The Special Method), 7.183%(The Last Position), 12.160%(TOP 10), when the cluster number is 64 and without PCA.

Compared to the codebook of Flickr60k, the Holidays dataset one performed worse. The reason might lie in that the the Holidays dataset is more biased because of a smaller image size. The image dataset with bigger size will contain more different elements than the smaller one, and

will provide a more authoritative codebook. What's more, PCA seems to have negative impact here. As the feature matrix of BOV is not highly-dimensioanl, and PCA did a dimensionality reduction to the feature matrix, which definitely lost some part of information, so the worse precisions are acceptable.

3.4 Experiments with Fisher Vector

From the previous section, it's easy to conclude that using the larger sized dataset Flickr60k to compute the vocabularies will provide a better accuracy. So, the author used Flickr60k to train the codebook as well as learn PCA. Five different number of clusters($\mathbf{K}$ = 1, 8, 64, 256, 512) were used, as well as three evaluation methods.

3.4.1 A Simple Implementation

It's clear that PCA has a positive impact on the accuracy, while the green line is always above the red one. So the auuthor implemented PCA in the rest of the project(except for the CNNs). In addition, the result shows that a larger cluster number will offer better accuracy. When the number of cluster is 512, plus implementing PCA, the best case has a MAP of 67.189%(The Special Method), 63.141%(The Last Position), 23.160%(TOP 10).

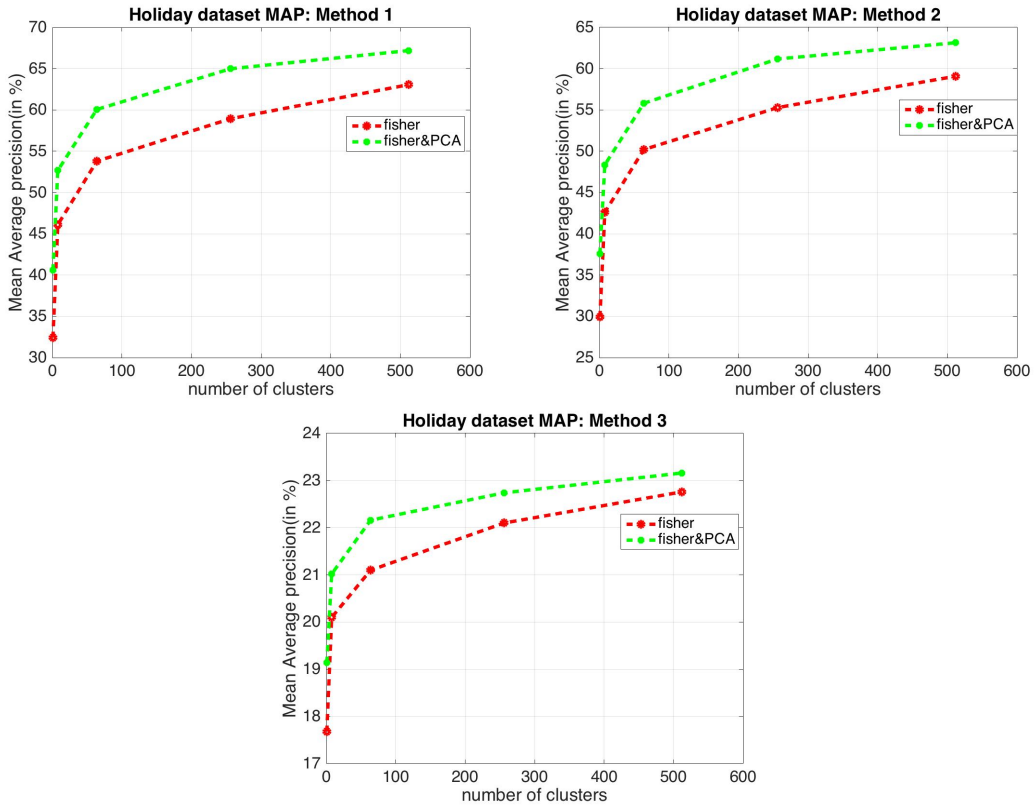

Figure 9: Fisher Vector: a simple implementation

3.4.2 Fisher Vector Normalization

Besides using a simple implementation of the Fisher Vector, the author also did a vector normalization trick. Figure 10 shows the result of three evaluation methods.

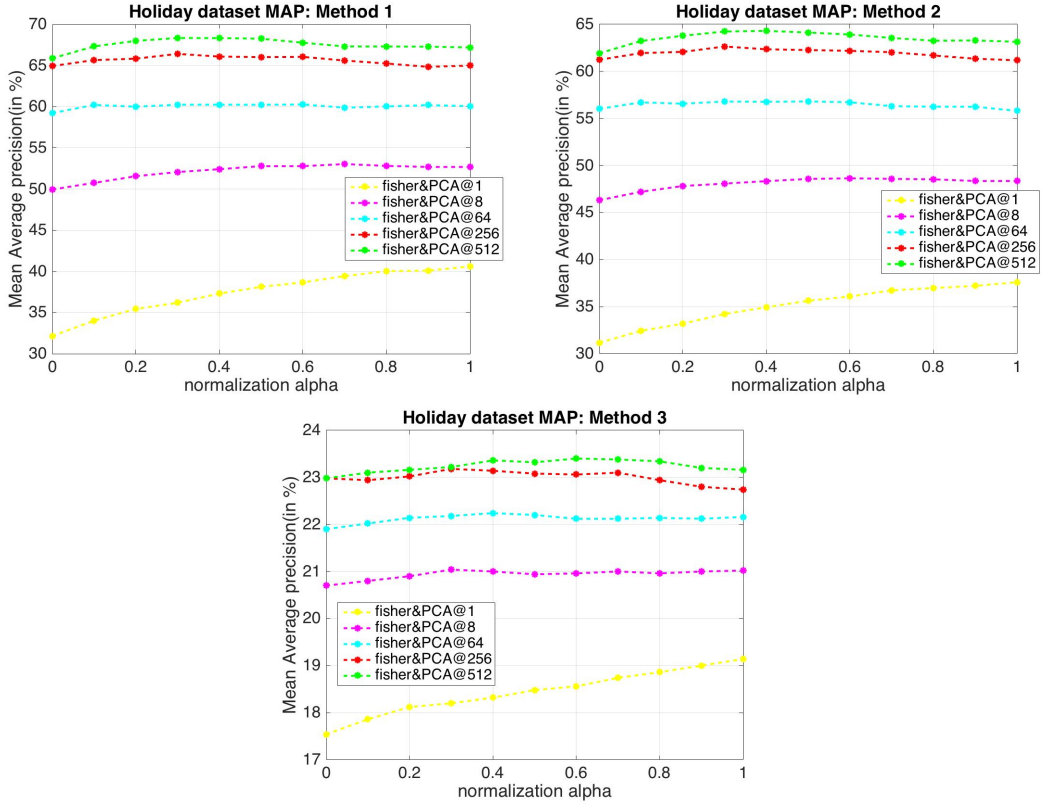

Figure 10: Fisher Vector Normalization

In the first case of using "The Special Method" to compute the MAP, the best accuracy is 68.335%, when the number of cluster is 512 and $\alpha = 0.3$. In the second case of using "The Last Position" to compute the MAP, the best accuracy is 64.299%, when the number of cluster is 512 and $\alpha = 0.4$. In the last case of using "TOP 10" to compute the MAP, the best accuracy is 23.400%, when the number of cluster is 512 and $\alpha = 0.6$.

Though a increase of the number of clusters will improve the accuracy, setting the Fisher Vector to different power numbers make changes as well. As the vocabulary size $\mathbf{K}$(the number of cluster) increases, in order to fetch the best accuracy, the optimal value of α gets closer to zero. For instance, in the first case of using "The Special Method" to compute the MAP, for $\mathbf{K}$(the number of cluster) = 1, $\alpha = 1.0$ yields the best result, while for $\mathbf{K}$(the number of cluster) = 512, $\alpha = 0.3$ is optimal.

3.4.3 Fisher Vector Binarization

The author experimented with two approaches to binarize the Fisher Vectors: one using the vector normalization with $\alpha = 0$, another one is SH with different bit numbers.

Figure 11 offers a comparison of the two proposed binarization schemas on the Holidays dataset for the vocabulary size of $\mathbf{K} = 8$. Surprisingly, the simple "$\alpha = 0$ normalization" binarization scheme works extremely well, even much better than the more complex SH encoding. All the two binarization encodings make no salient improvement on the MAP, even though "$\alpha = 0$ normalization" is better than SH encoding. So, just using the "Fisher Vector Normalization" with the best α is the optimal choice until now.

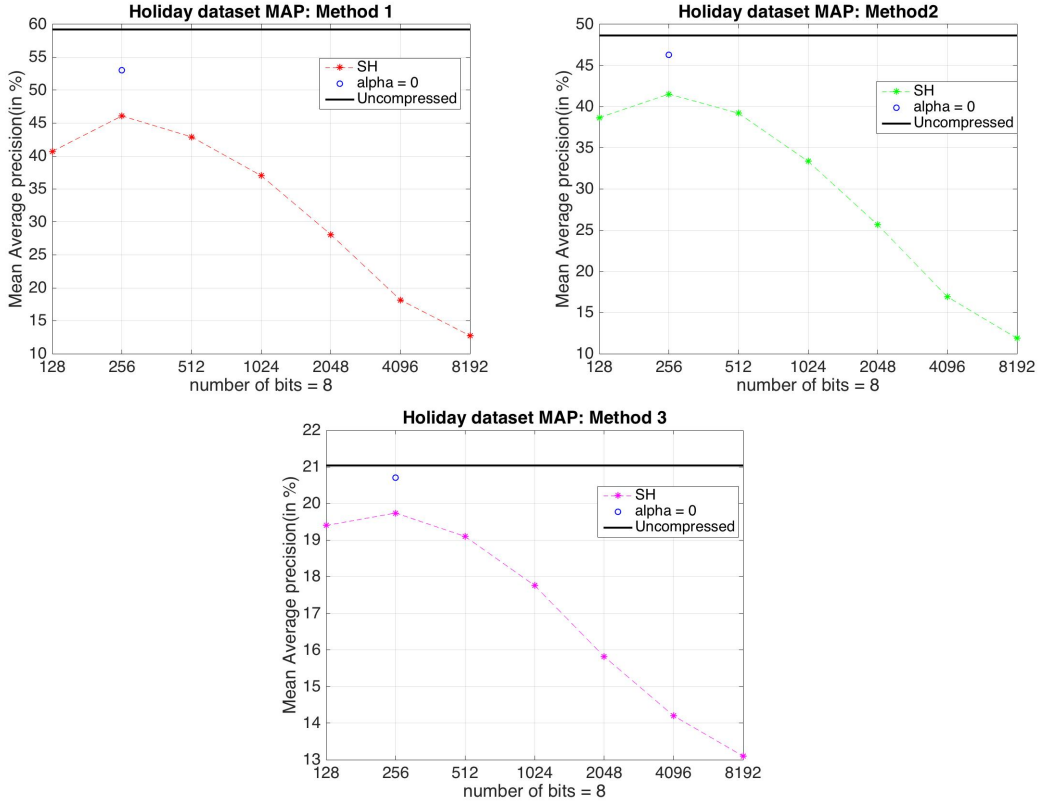

Figure 11: SH: vocabulary size is $\mathbf{K} = 8$, and the number of bits tried are 128, 256, 512, 1024, 2048, 4096, 8192. Uncompressed = nonbinarized accuracy(with the best α).

3.5 Experiments with CNNs

Figure 12 shows the result of CNNs. In the first case of using "The Special Method" to compute the MAP, the accuracy is 71.3230% . In the second case of using "The Last Position" to compute the MAP, the accuracy is 68.7084% . In the last case of using "TOP 10" to compute the MAP, the accuracy is 25.7800%. Compared to the previous experiments result, CNNs offers a the highest accuracy.

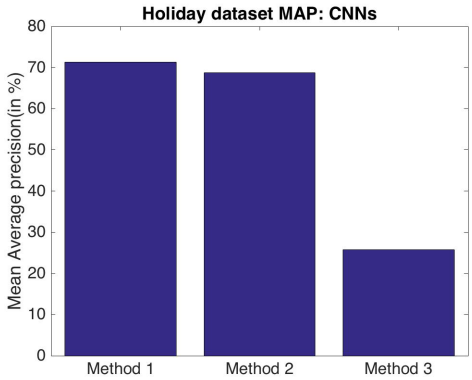

Figure 12: CNNs

4 Conclusions

In this semester project, the author applied three schemes to do the content-based image retrieval. BOV gives the lowest accuracy compared to the other two methods, and doing PCA in BOV even makes the result worse. The reason might be that since its feature matrix's dimensionality is not so high, doing PCA only means a loss of useful information.

Fisher Vector performs better than BOV, and an increase of the number of clusters (the vocabulary size) also means a higher accuracy. As the Fisher Vector is hgh-dimensional and dense, so doing PCA to reduce the feature matrix's dimensionality improves the accuracy. Besides, setting the Fisher Vector to different power numbers(α) makes changes as well. As the vocabulary size(the number of cluster) increases, in order to fetch the best accuracy, the optimal value of α gets closer to zero. Though doing binarization to Fisher Vector might increase the ranking speed. However, the binarization doesn't improve accuracy at all. The complex SH encoding shows a worse effect than the simple "$\alpha = 0$ normalization" binarization, while setting $\alpha = 0$ seldomly provides the best accuracy compared with other α under a specific vocabulary size. So, In this case, doing a "Fisher Vector normalization" with the best α must be the best plan.

What's more, CNNs significantly outperforms than the other two methods. To sum up, CNNs, the one with best accuracy is the author's final choice.

Acknowledgements

This semester project was realized at the IVRL lab at the École polytechnique fédérale de Lausanne (EPFL), under the responsibility of Professor Sabine Süsstrunk and under the supervision of Bin Jin whom the author wish to thank for his helpful comments.

References

[1] Ying Liu, Dengsheng Zhang, Guojun Lu, Wei-Ying Ma. *A survey of content-based image retrieval with high-level semantics.* Monash University, Australia; Microsoft Research Asia, China, 2006. Pattern Recognition 40.1 (2007): 262-282.

[2] "Visual Descriptor" *wikipedia.* `https://en.wikipedia.org/wiki/Visual_descriptor`.

[3] "Pattern Recognition Systems – Lab 5, Histograms Of Oriented Gradients" `http://users.utcluj.ro/ raluca/prs/prs_lab_05e.pdf`.

[4] "Histogram Of Oriented Gradients (HOG) Descriptor | Intel® Software". *Software.intel.com.* `https://software.intel.com/en-us/node/529070`.

[5] "A Short Introduction To Descriptors". *Gil's CV blog.* `https://gilscvblog.com/2013/08/18/a-short-introduction-to-descriptors/`.

[6] "Bag Of Words Models For Visual Categorization". *Gil's CV blog.* `https://gilscvblog.com/2013/08/23/bag-of-words-models-for-visual-categorization/`.

[7] Perronnin, Florent, et al. *Large-scale image retrieval with compressed fisher vectors.*. Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. IEEE, 2010.

[8] "Vlfeat - Documentation > C API". *Vlfeat.org.* `http://www.vlfeat.org/api/fisher-fundamentals.html`.

[9] Y. Weiss, A. Torralba, R. Fergus. *Spectral Hashing.* Advances in Neural Information Processing Systems, 2008.

[10] Y. Weiss, A. Torralba, R. Fergus. *Multidimentional Spectral Hashing.* Advances in Neural Information Processing Systems, 2008.

[11] Jegou, H., Douze, M.: INRIA Holidays dataset `http://lear.inrialpes.fr/people/jegou/data.php`. (2008)

[12] "Visual Geometry Group Home Page" `http://www.robots.ox.ac.uk/ vgg/research/very_deep/`. Robots.ox.ac.uk.

[13] Karen Simonyan, Andrew Zisserman *Very Deep Convolutional Networks For Large-scale Image Recognition.* Visual Geometry Group, Department of Engineering Science, University of Oxford.

[14] Ji Wan, et al. *Deep Learning for Content-Based Image Retrieval: A Comprehensive Study.* 2014 ACM 978-1-4503-3063-3/14/11.