Improving three aspects of a content based image retrieval system

Feng Wang
Improving three aspects of a content based image retrieval system

by

Feng Wang

in partial fulfillment of the requirements for the degree of

Master of Science
in Computer Science

at the Delft University of Technology,
to be defended publicly on Thursday March 17, 2016 at 13:00 PM.

Supervisor: Dr. D.M.J. Tax, TU Delft
Thesis committee: Prof. dr. ir. M.J.T. Reinders, TU Delft
Prof. dr. Alan Hanjalic, TU Delft
Dr. J.C. van Gemert, TU Delft

This thesis is confidential and cannot be made public until March 17, 2016.

An electronic version of this thesis is available at http://repository.tudelft.nl/.
Preface

This thesis is submitted as a part of the requirements for the degree of Master of Science in Computer Science at the Delft University of Technology. I spend about a year working on the thesis project at the Pattern Recognition Lab, and the details of my work are presented in the following report.

I would like to give my special gratitude and thanks to my supervisor: dr. David Tax. It is an honour to work with him. During my thesis, the most exciting parts were the inspiring discussions with him. I have really learned a lot about how to perform a scientific research from his help, advice, and patience. I would say without him it is not even possible for me to accomplish this master thesis. Also I would like to thank the committee members Prof. M.J.T. Reinders, Prof. Alan Hanjalic, and Dr. J.C. van Gemert.

Besides, this thesis is supported in part by NWO/SurfSara project SH-318-15. Thanks for their Cartesius cluster and the high performance GPUs.

At last I would like to thank my family and friends for their help during my master studies.

Feng Wang
Delft, March 2016
A content-based image retrieval system directly accepts image(s) as input, and returns similar images to the user. The key step in the searching process for a content-based image retrieval system is calculating the similarities between the query image and the images in the database, followed by generating a ranked list based on the similarities.

In this thesis work, we give a framework to solve the above problem, which contains three parts: image representation, feature combination, and ranking. In detail, firstly, some representations are extracted from the query image and the images in the database. Then, some similarities are calculated for a query-image pair. At last, the ranking algorithm is used to get a final rank from the similarities vector.

For each part of the framework, we propose some candidate methods, and their characteristics are studied in detail. By applying the framework, a simple way of combining multiple image representations is introduced, and its performance is evaluated in the experiment. The results indicate that there is a notable improvement by combining multiple image representations than using them individually.

Besides, this work also studies how to treat the ranking problem as a classification problem. Several asymmetric classification losses are proposed and compared. The comparison results give some instructions on how to design a good classification loss for the ranking problem.

At last, we build an end-to-end trainable content-based image retrieval system based on the convolutional neural network (CNN). The experimental results for the end-to-end system are not satisfactory, where the reasons are analyzed and some potential solutions are proposed.
## Contents

### Abstract

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2 Background of the content based image retrieval</td>
<td>5</td>
</tr>
<tr>
<td>3 Convolutional neural network</td>
<td>15</td>
</tr>
<tr>
<td>4 Feature combination</td>
<td>25</td>
</tr>
<tr>
<td>5 Learning to rank</td>
<td>29</td>
</tr>
<tr>
<td>6 Evaluation criteria</td>
<td>33</td>
</tr>
<tr>
<td>7 Objective</td>
<td>36</td>
</tr>
</tbody>
</table>
Improving three aspects of a content based image retrieval system

Summary and future work

General training settings

Visual word/CNN + unsupervised similarity measurements + sorting

Learning to rank

Datasets

Comparison with the state-of-the-art

Experiments of different image representations

Oxford5k

Training both fc10 and fc6 with a large regularization term

Visual retrieval examples of the powered asymmetric loss and the logistic loss

Experiments of the asymmetric losses

ukbench

Distance metric in Equation 4.12

Experiments of an end-to-end trainable retrieval system

Full ConvNet structures

Summary and future work

Appendix

Bibliography
Introduction

The rapid growth of network hardware and search engines has spurred the development of many multimedia applications, in which the image retrieval plays an important role to make the search engines more “intelligent” and powerful. In general, image retrieval is the technique to retrieve images from a large database, based on a query from the user. The query can be provided in the form of some texts, a query image, or both. All the inputs, except the basic image pixel information, can be regarded as some metadata of the image.

Traditional image retrieval systems use the metadata of the image, especially the texts, as input. This kind of system has advantages when the user only wants to search some general objects, e.g., a cellphone, a chair, a cake, and so on. Or when the user has the explicit information of the thing to be found, like the TUDelft library, or the Eiffel Tower, the text input image retrieval system also shows good performance. However, there are some circumstances that the text input image retrieval cannot handle. For example, if one does not know the name of the building in Figure 1.1a, he will probably describe it by text as “tower” or “tower with a footbridge”. Submitting these query texts to a real world search engine, like Google Images\(^1\), the top returned images are showed in Figure 1.1b, which are not satisfactory because none of the buildings in Figure 1.1b are even close to the object that the user intended to find. In the case when one wants to find some specific objects or scenes without an explicit description, the system that accepts image pixel information usually is able to give better results. For example, Google Images can also accept image as input, and submitting Figure 1.1a as query, the top returned images are shown in Figure 1.1c, which are obviously better compared to Figure 1.1b. This kind of image retrieval system is called the content based image retrieval system. The term “content based” indicates that the system analyzes the content of the image instead of some metadata, and obviously, the content can be referred to gray level, intensity, color, shape, texture, and so on. Some techniques in the content based image retrieval allow multiple images or a clip of video as inputs, while this kind of techniques is not included in this paper. We only consider the one-image-input content based image retrieval.

1.1. Problem description and our solution

A real content based image retrieval system should firstly maintain a big image database. Then the user will input an image query which contains the objects he desired to search. The retrieval system compares the input image with all images in the database, and returns a list of images to the user which are ranked by their similarities to the query. At last, during the search, the user can give some implicit feedbacks to the retrieval system, e.g., the clicks, mouse hovers, and so on.

In this paper, we focus on calculating the similarities between the query and the images in the database, and generating a ranked list based on the similarities. The traditional content based image retrieval system is based on the bag-of-visual-words method, which firstly extracts a vector of the SIFT \(^1\) based visual words to represent an image. Then a weighted cosine distance is calculated between the representations of a query-image pair. At last a simple sorting is used to rank the images in the

\(^1\)https://www.google.com/imghp
1. Introduction

(a) A sample query image (Kapellbrücke in Switzerland).
(b) Top returned images using "tower with a footbridge" as input from Google Images.
(c) Top returned images using Figure 1.1a as input from Google Images.

Figure 1.1: A sample query and the searching results.

Database based on their distances to the query image. There are some problems for this traditional method:

1. It is hard to be extended. For instance, it is not trivial to use multiple image representations at the same time. To do so, one has to invent some new feature fusion methods, e.g., [2], [3].

2. The method is unsupervised, and all parts are carefully designed and tested by the experts. The performances for different datasets may be not always optimal. Consequently, some manual tunings are required to use the same techniques in different datasets.

To make the system extendable, we analyze the content based image retrieval problem from a system perspective, and propose a decoupled framework, where there are three main parts: image representation, feature combination, and ranking. In detail, because the query image and the images in the database usually are of different size, format, some universal representations needs to be obtained from them. Then some similarities can be calculated from a pair of image representations, and this step is the feature combination. At last, the ranking algorithm is applied to get a final rank from the similarities vector.

Some candidates are given for each step of the framework. Specifically, two image representations are proposed, which are the SIFT based visual words, and the image descriptor extracted from a pre-trained convolutional network (ConvNet). Besides, we treat the ranking problem as a binary classification problem, i.e., each query-document pair has a label: relevant or irrelevant. Especially, the ranking problem is differen from the classification problem where the top positions in a ranked list are much important than the bottom positions. In order to better fit the characteristics of the ranking problem, we propose and investigate some adjusted loss functions. At last, when applying a supervised ranking method, it is possible to build an end-to-end trainable system. This thesis work also explores how to build an end-to-end trainable system for the retrieval problem, and evaluates its performance.

1.2. Research questions

The main questions to be answered in this paper are as follows.

1. For the two image representation candidates, i.e., the SIFT based visual words, and the image descriptor extracted from a pre-trained ConvNet, which is better for the content based image retrieval problem?
1.3. Thesis outline

The outline of this thesis report is as follows. In Chapter 2, we give a detailed description of the content based image retrieval, followed by a brief introduction of the evaluation criteria of the retrieval problem. Then the traditional bag-of-visual-words based method are reviewed and analyzed. The notations used in this paper are also presented in this chapter.

Chapter 3 introduces the convolutional neural network, and illustrates how to use a pre-trained convolutional network to extract a descriptor from an input image.

In Chapter 4, two feature combination methods are proposed. The first one is called the traditional feature combination method in text retrieval, which directly borrows the techniques from the text retrieval. The second one is the distance metric learning, where we propose two simple distance metrics.

The supervised ranking algorithms, learning to rank, is introduced in Chapter 5. There are three main approaches of learning to rank, i.e., the pointwise approach, the pairwise approach, and the listwise approach. In this thesis, we only evaluate the pointwise learning to rank, where the ranking problem is treated as a binary classification problem. So almost all classification losses can be used to guide the training. Most of the losses designed for classification are symmetric, i.e., they treat the positive and negative objects equally. However, for the ranking problem, the top positions in a ranking list are crucial, which suggests that the false negative objects should be avoided to the greatest extent. So we propose several asymmetric classification losses to ensure the images in the top positions are classified correctly.

Chapter 6 shows all candidate methods proposed in the previous three chapters, and illustrate how to embed them into our framework. Besides, we also gives an overview of what methods will be evaluated in the experiments in this chapter.

The following three chapters are experimental chapters. In Chapter 7 we firstly introduce the datasets to be used, and give some general settings for all experiments. Then different image representation methods are compared, as well as a general comparisons between the supervised and unsupervised ranking methods. At last, we show how to combine multiple image representations using our framework and give their experimental results.

In Chapter 8, we make a thorough comparison between the symmetric loss and the asymmetric losses when treating the ranking problem as a classification problem. Besides, by comparing different asymmetric losses, we also reveal the key of designing a successful asymmetric loss for ranking. At last, we check if our supervised method has enough generalization power by the experiments of transfer learning, i.e., we train the model on one dataset and test it on another one.

In Chapter 9, we build and test an end-to-end trainable retrieval system based on the CNN.

Finally, Chapter 10 summarizes the whole thesis, gives an overall discussion of the experimental results, and proposes some future research directions.
In this chapter, we introduce how a common content based image retrieval system works, where the step “searching” is analyzed in details. The notation is introduced, together with some commonly used evaluation criteria. Then the traditional bag-of-visual-words method is described. At last, some related works of improving the traditional bag-of-visual-words method are presented.

2.1. Framework

A real-world content based image retrieval system has four steps, which are shown in Figure 2.1. Firstly, the retrieval system maintains a big image database in which the images are usually crawled from the Internet or uploaded by the users. Then the user can input an image which contains the objects he desired to find to the system. Then the retrieval system compares the input image with all images in the database, and returns a ranked list to the user. At last, the user can give some feedbacks to the retrieval system, like the clicks, mouse hovers, and so on. In this paper, we only focus on the “searching” step which is the key component in a content based image retrieval system.

The “searching” step can be divided into four main parts shown in Figure 2.2. Each step is explained as follows. The first step is the image representation. The content based image retrieval system accepts image pixel information as input. However, the raw image pixel cannot be directly employed to compare two images, and predict the relevance level between them. On one hand, both the input and potential output images may be of different sizes, formats, so it is almost impossible to perform calculations directly on pixels. On the other hand, usually one image contains hundreds of thousands of pixels, which make the calculation very slow. So before comparing two images, some image representations should be extracted from them, such that two feature vectors for a query-image pair with the same size can be used in further processing.

The second step is the feature combination. Different from the traditional image classification task, where each image has a label, in the retrieval problem, usually a bag of two or more images share a label. For example, in some image retrieval tasks, a query image with an image in the database will have a binary label relevant or irrelevant, or a real value label to indicate the relevance level. Besides, associated with the same query, two images in the database can have label 1 or -1, where 1 means the first image is more relevant than the second one, and -1 is the other way around. Even associated with the same query, the relevance degrees, or a ranked list (permutation), of all images in the database can be regarded as the label. As a result, for the retrieval problem, two objects should be combined and a new feature vector will be extracted. And in the case of the content based image retrieval, “two objects” means two image representations, where one is the query image representation, and the other one is the representation for an image in the database.

The third step is the ranking. After the feature combination, different type of feature vectors can be obtained. If the feature is just a scalar value, then a simple sorting can be applied to get the final ranked list. Otherwise if the feature is a vector of value, some loss functions as well as different
2. Background of the content based image retrieval

Figure 2.1: How a content based image retrieval system works.

Figure 2.2: The framework of the "searching" in a content based image retrieval system.
optimization methods can be used to learn an appropriate model. Then the model can be used to rank all images in the database corresponding to a query.

The last step is re-ranking. Because in the real applications, the image database is usually very large, the methods from the “ranking” step usually should be very efficient, which means they probably will lose some details. So the ranked list from last step may not be the final result. An optional re-ranking step can be employed by using some complex methods on top images of the ranked list to refine the result. In this report, we will not discuss re-ranking.

In summary, there are four main steps in the searching framework, which are image representation, feature combination, ranking, and re-ranking. We only focus on the top three steps.

### 2.2. Notations

In this paper, the following notations are used. 

\[ D = \{ d_i | i \in 1, \ldots, |D| \} \]

is the document set contains all the images to be retrieved, and obviously there are \(|D|\) images at all. 

\[ Q = \{ q_i | i \in 1, \ldots, |Q| \} \]

represents the query set which contains \(|Q|\) queries. For each query \( q \) in the query set, the document set can be divided into two parts: relevant document set \( D_r(q) \) and irrelevant document set \( D_{ir}(q) \). Similarly, 

\[ D_r(q) = \{ d_i | i \in 1, \ldots, |D_r(q)| \}, \text{ and } D_{ir}(q) = \{ d_i | i \in 1, \ldots, |D_{ir}(q)| \}. \]

For an input query \( q \), the retrieval system will give a returned set \( R(q) = \{ d_i | i \in 1, \ldots, |R(q)| \} \), which is usually set as a permutation of all images in the database in a real world retrieval system. Then let \( t \) denote the image representation method, and \( comb \) denote the feature combination method. For a query \( q \) and an image \( d \), a feature vector \( v_{q,d} \) is extracted by:

\[
v_{q,d} = comb(t(q), I(d)) \quad (2.1)
\]

At last, a ranking model \( M \) is applied on all \( v \)s to get the ranking of all images in the database.

### 2.3. Evaluation Criteria

For the content based image retrieval system, some general evaluation criteria are used to assess the performance.

1. **Precision-recall curve (PR curve).** From Dumais [4], “recall is the proportion of all relevant documents in the collection that are retrieved by the system; precision is the proportion of relevant documents in the set returned to the user.” For a query \( q \), we can have:

\[
\text{Precision} = \frac{|R(q) \cap D_r(q)|}{|R(q)|} \quad (2.2)
\]

\[
\text{Recall} = \frac{|R(q) \cap D_r(q)|}{|D_r(q)|} \quad (2.3)
\]

By changing the size of \( R \), a PR curve can be obtained, like Figure 2.3. Although to get high precision and recall at the same time is very difficult, it can be seen as an ultimate goal for the retrieval system. The biggest shortcoming for the PR curve is that it is hard to be used to for direct comparison of the performances. So other evaluation methods are needed.

2. **Precision at position \( p \) (P@p).** Now if we set \( y_i \in \{-1, 1\} \) denotes if the \( i \)-th document is relevant \((y_i = 1)\) or irrelevant \((y_i = -1)\), with the same definition of the precision mentioned above, the precision at position \( p \) is:

\[
P@p = \frac{\sum_{j=1}^{p} (y_j + 1)}{2p} \quad (2.4)
\]

Generally speaking, the precision at position \( p \) shows the precision when only returning top \( p \) results, which is useful in measuring the top \( p \) results.

3. **Mean average precision (mAP).** The precision is a single-value criterion, but it does not consider the order in the returned list, which is an important property in a retrieval system. The precision at position \( p \) considers order information, but only shows how good of the top \( p \) results.
2. Background of the content based image retrieval

The average precision (AP) considers the order of all returned documents by averaging the \( P@p \) for all positions. The definition of AP is

\[
AP = \frac{1}{|D_r(q)|} \sum_i \frac{|R(q)|}{4|i|} \sum_{j=1}^i (y_j + 1)
\]  

(2.5)

The AP represents the area under the PR curve. Note that AP is a position based evaluation criterion. If we make the \( D_r(q) = 1 \) and \( D_r(q) = 10 \), and move the position of the relevant document in the returned list gradually from the top to the bottom, the APs are shown in Figure 2.4. From Figure 2.4, it is clear that the top positions have dominating influence on AP. When more irrelevant documents have higher ranking than the relevant documents, the AP decreases greatly. And comparing Figure 2.4a and Figure 2.4b, the slope of the curve in Figure 2.4b is not as steep as in Figure 2.4a. So for a query, when there are more relevant documents, the influence of the wrongly ranked irrelevant documents decreases.

The mean Average Precision (mAP) is calculated by averaging the APs over all input queries. So in average, the top positions still have larger influences on the mAP. Currently the mAP is the most popular evaluation criterion for the retrieval problem, and we also use it as the main evaluation criterion in this work.

4. Running time. In most of the scenarios, the image retrieval system needs to return the results in real-time, which makes the running time an important evaluation criterion. Considering the user interaction, the goal of the running time is to be as short as possible. But the running time is to a large degree determined by implementation/hardware, so we will not discuss this criterion in this report.

5. Memory. With the development of the storage hardware, the memory is usually not critical for an image retrieval system. But in some special situations, it is still significant and is usually the key restriction of the whole system, e.g., the mobile visual search. Because memory requirements is highly influenced by the usage scenarios, we will also excluded this criterion.
2.4. Bag-of-visual-words based method

The bag-of-visual-words based method is the traditional and most widely used content based image retrieval algorithm, originally proposed in [7]. The basic idea of the bag-of-words based method is to firstly represent each image using a set of visual words from a visual dictionary, which is similar to the representation of text documents in a natural language. To perform the ranking, a similarity between the query and each image in the database is calculated. We will elaborate on that in the coming sections.

2.4.1. Image representation

Because the tradition bag-of-visual-words based method is an unsupervised method, the image representation part is carefully designed by experts. It can be divided to three sub-processes:

Words extraction and dictionary construction

From [7], there are three main steps in words extraction and dictionary construction.

1. Feature points detection.
2. Feature points description.

For the feature points detection, the Hessian affine detector [8] is used. For a common image, hundreds to thousands key points can be detected.

Then for each key point, the classical SIFT descriptor is used, and a 128 dimension descriptor is obtained. SIFT descriptor is invariant to scale, translation, and rotation, which is helpful in comparing if two images are similar or not. But as a local descriptor, SIFT ignores the global information of the image. It also ignores the color information, which is an important property of images. Besides, the descriptor has 128 dimensions, which is too large to be used in a real-time system, and this is the reason why a dictionary needs to be obtained. All these disadvantages make people pursue in the path of finding a better representation of the images.

The feature point detection and description is applied to a large collection of images to generate a big dataset that contains thousands to millions of SIFT descriptors. Then the approximate k-means method is used to cluster this dataset, and the clusters centers are treated as the visual words in the dictionary. Let the visual words set we get be \( t = \{ t_i \mid i \in |t| \} \), and the number of visual words \( |t| \) is usually much larger than the number of key points detected in an image.

Visual words of the images

After generating a visual words dictionary, for each image \( d \), the same key points detection and description methods is applied to get the SIFT descriptors. Then the approximate nearest neighbor (ANN)
algorithm is used to assign each descriptor to a visual word. Now the input image \( d \) is represented by a vector of visual words, i.e.,

\[
v_d = (c(t_1, d), ..., c(t_i, d), ..., c(t_{|t|}, d))
\]  

(2.6)

where \( c(t_i, d) \) means the number of occurrences of the visual \( t_i \) in image \( d \). Since the size of the dictionary usually is much larger than the number of visual words an image may contain, the visual words vector is usually very sparse. Now the similarity between two images can be calculated based on two visual words vectors, for example, cosine similarity, Euclidean similarity, and so on. To make the calculation efficient, the inverted index is used.

**Inverted index**

After the previous steps, every image can be represented by a vector of visual words. In order to calculate the similarity efficiently, the inverted index is used for the image database. In a normal index, the entry is the documents list which contains all words appearing in the documents. The inverted index on the other hand has an entry for each word and stores all documents containing this word. In the case of content based image retrieval, an entry of the inverted index is a visual word, and it stores the number of occurrences of the images containing this visual word. Since the visual words vector of an image usually is very sparse, the inverted index for the image database makes the calculation of similarity and retrieval very fast.

For example, in Figure 2.5, there are three images \( d_1, d_2, d_3 \) and the size of dictionary is 4 (4 visual words \( t_1, t_2, t_3, t_4 \)). The left part is the normal index, in which image \( d_1 \) has 3 words (1 \( t_1 \), and 2 \( t_4 \)), image \( d_2 \) has 3 words, and image \( d_3 \) has 1 word. The middle and left parts are the inverted indices. The right part shows the sparsity of the inverted index, and usually in the implementation, some corresponding sparse functions or modules are used.

**2.4.2. Feature combination**

After we have found the visual words of all images in the database and the query, the tf-idf (term frequency-inverse document frequency) weighted cosine distance is used to combine two visual words vectors.

**Tf-idf**

Instead of directly using visual words occurrences in similarity calculation, usually the so-called tf-idf weights is firstly assigned to the inverted index, which is an analogy to the classical text analysis and retrieval. The the tf-idf weight of each word \( t_i \) in the document \( d_j \) is calculated by:

\[
\text{tf-idf}_{t_i,d_j} = tf_{t_i,d_j} \cdot idf_{t_i}
\]  

(2.7)

where the \( tf_{t_i,d_j} \) is the term frequency:

\[
tf_{t_i,d_j} = \frac{c(t_i, d_j)}{|d|} = \frac{c(t_i, d_j)}{\sum_{k} c(t_k, d)}
\]  

(2.8)

And \( idf_{t_i} \) is the logarithm of the inverse document frequency. The document frequency of word \( t_i \) \((df_{t_i})\) represents the number of documents which contain word \( t_i \). So we have:
2.4. Bag-of-visual-words based method

\[ df_{t_i} = \frac{|\{d | t_i \in d\}|}{|D|} \]  

where \(|\{d | t_i \in d\}|\) means the number of documents which contain the word \(t_i\), and then the idf_{t_i} is defined by:

\[ idf_{t_i} = \begin{cases} 
\log \frac{|D|}{|\{d | t_i \in d\}|} & \text{if } |\{d | t_i \in d\}| \neq 0 \\
0 & \text{if } |\{d | t_i \in d\}| = 0 
\end{cases} \]  

Then the visual words vectors are updated by the tf-idf weights as shown in Equation 2.11. The new visual words vector will be used in the next section to calculate the similarity.

\[
v_d = (v_{d,1}, v_{d,2}, ..., v_{d,|D|}) 
= (\text{tf-idf}_{t_1,d} \cdot c(t_1, d), ..., \text{tf-idf}_{t_{|D|},d} \cdot c(t_{|D|}, d)) \]  

For example, in Figure 2.5, the idfs for all words are \((\log \frac{2}{3}, \log \frac{2}{3}, 0, \log \frac{2}{3})\), and tf for image \(d_4\) is \((\frac{1}{3}, 0, 0, \frac{2}{3})\). So the tf-idf weighted index for image \(d_4\) is \((\frac{1}{3} \log \frac{2}{3}, 0, 0, \frac{2}{3} \log \frac{2}{3}) = (0.059, 0, 0, 0.117)\), and the whole example inverted index with the tf-idf weight is shown in Figure 2.6.

Similarity calculation

Based on the inverted index, there are different ways to calculate the similarity between two images, for example, the cosine distance, the Euclidean distance, or the Hellinger similarity [10]. The most widely used similarity measurement is the cosine distance. For two images \(q, d\), they can be firstly expressed by two tf-idf weighted visual words vectors. Then the cosine distance is calculated by:

\[
similarity(q, d) = \cos(v_q, v_d) = \frac{v_q \cdot v_d}{||v_q|| \cdot ||v_d||} = \frac{1}{||v_q|| \cdot ||v_d||} \sum_{k=1}^{||D||} v_{q,k} v_{d,k} 
= \sum_{k=1}^{||D||} v_{q,k} v_{d,k} \quad \text{(if } v_q \text{ and } v_d \text{ are normalized)} \]  

In the implementation, after the normalization of the inverted index, this step is only a simple matrix multiplication, which is efficient enough for the large scale image retrieval. Now the similarities between a query and all images in the database are calculated, which are then used to generate a ranked list.

2.4.3. Ranking

The traditional bag-of-visual-words based method is unsupervised, and after we get the tf-idf weighted cosine similarities between the query and all images in the database, a simple ascend sorting can return the ranked list.
2.4.4. Shortcomings
The traditional bag-of-visual-words based method has good performance when evaluated by mean average precision. But there are still some shortcomings.

First of all, from a system perspective, this method is unsupervised and ad hoc, which means all parts are carefully designed and tested by the experts. Firstly, this results in that the bag-of-visual-words method is hard to be extended. For example, if one wants to use multiple image representations, he has to design some feature fusion method. Secondly, since the method is unsupervised, when applied on different datasets, it has to be manually tuned. A reasonable solution is to use supervised algorithms in ranking to learn the best features as well as policies.

Secondly, because the visual words are built on SIFT descriptors, the bag-of-visual-words method also shares the properties of SIFT. Since SIFT is locally invariant in scale, rotation, and translation, the bag-of-visual-words method is robust to corresponding image deformation to some extend. But SIFT does not consider global spatial information of the image, which makes the retrieval system needs additional spatial verification in the re-ranking step. And the low efficiency of the spatial verification usually limits the algorithm only applied to the top ranked results.

At last, the bag-of-visual-words method is a k Nearest Neighbor (kNN) algorithm. But the high dimensionality of SIFT, i.e., the 128 dimension descriptor for each key point, requires the reduction of the dimensionality of the descriptors, which is why a visual dictionary needs to be generated. Besides, the visual words are used to calculate the similarity instead of the raw SIFT descriptors. In the end, the whole procedure becomes more complex. A straightforward solution is to directly extract some global representations from the images.

2.5. Related work
As mentioned above, for the sake of high efficiency, the dimensionality of the descriptors is reduced, while it suffers from the sacrifice on accuracy. Some improving methods are proposed trying to find the best balance between them. For example, Hamming Embedding [11] reduces the dimensionality by binarization, so for each SIFT descriptor, not only a visual word will be obtained, but also a binary code, which shows the distance between the descriptor to the cluster center. Experiments show that Hamming Embedding can improve the performance greatly compared to traditional visual words image representation, and with a good implementation, the loss of efficiency is still acceptable. Although Hamming Embedding has a better balance between the accuracy and the efficiency, it is still based on the SIFT descriptors, which means the color and spatial information are still missing.

Many works are devoted in importing color information into content based image representation. A straightforward solution is to use an image representation which is color related, e.g. CSIFT builds the SIFT descriptors in a color space [12]. Another way is to fuse the color descriptors with the SIFT descriptors. For example, coupled multi-index [13] proposes a new indexing scheme, the coupled multi-index, to use the color information of an image. Basically, each key point is not only described by a SIFT descriptor, but also an 11 dimension “color names” descriptor, and then the multi-index are used to fuse them.

For the spatial information, some improvements of the SIFT are proposed to embed geometric information into descriptors, for example, the weak geometric consistency [11], Geometry-Preserving Visual Phrases [14]. But usually geometric verification methods are computational expensive, only very simple spatial information can be embedded into descriptors. Actually, the spatial information usually is considered in the re-ranking step. In the re-ranking, only top images in the returned list are processed to verify the spatial information, so some complex but slow geometric verification methods, e.g., the random sample consensus (RANSAC), can be applied to them.

Compared to the local descriptors, the global descriptors have some naturally advantages. For example, the calculation of similarities using global descriptors is easier. So some global descriptors are tried for image retrieval problem, e.g., Gist [15], VLAD [16]. Recently, with the development of CNN, a pre-trained convolutional network (ConvNet) can also be used as a global descriptor extractor. Some CNN based global descriptors [17] are proposed and show competitive performances in content based image retrieval.

All the methods mentioned above are unsupervised, which is reasonable in real application: we should not assume the database contains the images similar to user input. So a supervised method may result in overfitting. But the supervised method still has some advantages. Firstly with a su-
Related work

A supervised method, one can probably get better parameters automatically instead of manfully designing and choosing some hyperparameters. Secondly, more different features can be combined with the supervised method. At last, when the database becomes large enough, it makes sense to assume the database will cover most of the types of the user input images.

Currently not so many attentions have been paid to using supervised method in content based image retrieval. [18] combines bag-of-visual-words with the supervised ranking algorithm, so the shortcomings of the SIFT descriptors still exist. [19] learns a distance metric based on the CNN features, while not all loss functions and learning schema are tried.

In this paper, we improve the traditional content based image retrieval system from the following aspects:

1. Using the CNN based image representation.
2. Proposing new feature combination methods.
3. Using learning to rank as a supervised learning method, and proposing several new ranking losses.
3 Convolutional neural network

As mentioned above, a deep CNN can be used as a global feature extractor. In this chapter, we will firstly illustrate some attributes of the CNN and compare it with the bag-of-visual-words method. Then we will briefly introduce how to build and train a deep ConvNet. At last, we will show how to use a pre-trained ConvNet to extract a representation from an input image.

3.1. Introduction

Recently, with the breakthrough and success [20] of using the CNN on image classification, the CNN become popular, and many work has been devoted to improving the architectures and the learning algorithms to make the CNN fit for other tasks than simple image classification. One important reason for the popularity is that, when it is trained on a very large image classification dataset, the ConvNet can be easily transferred to other computer vision tasks, and often shows satisfactory or even better performance than the traditional methods. We will also use pre-trained ConvNets as the image representation methods in the content based image retrieval, in two different ways:

1. Using the pre-trained ConvNet as a fixed feature extractor. The output of any layer of a pre-trained ConvNet can be regarded as a descriptor of the input image. In Chapter 7, we compare the performance for the outputs of different layers by experiments.

2. Using the pre-trained ConvNet as an initialization and retrain (part of) it. Instead of using the ConvNet as a fixed feature extractor, all or a part of the ConvNet can also be tuned for better fitting the specific task. In our situation, we combine the ConvNet with some ranking losses, which is introduced in Chapter 9, to tune it on target datasets.

Essentially, the CNN image descriptors can be seen as a global image representation, which encodes color and spatial information naturally. But on the other hand, if too much spatial information is encoded, one may intuitively doubt that it is not robust enough to some deformations. So in general for retrieval problem, the balance between global and detail of the image representation needs to be made.

3.2. Neural Network

Convolutional neural network is a type of neural network. In this section, we introduce the basic components of a neural network, the structure of the fully-connected neural network, and how to train a neural network.

3.2.1. The neuron

The neuron is the basic component of all neural networks. The name “neuron” already indicates that it is firstly inspired by the human neuron cell. Ignoring the biological implication, a neuron in the computational model accepts one or more inputs, reweighs the inputs, sums them, and finally gives one output by passing the sum through an activation function. Assume the inputs $x = (x_1, x_2, x_3, ..., x_n)$,
the corresponding weights are \( \mathbf{w} = (w_1, w_2, w_3, \ldots, w_n) \), the activation function is \( f \), and the output is \( \text{out} \), then for a neuron, we have

\[
\text{out} = f \left( \sum_{i=1}^{n} (w_i x_i) + w_0 \right)
\]

(3.1)

where \( w_0 \) is the bias parameter. If we add \( x_0 = 1 \) into \( x \) and rewrite the previous equation in vector form, then

\[
\text{out} = f \left( \sum_{i=0}^{n} (w_i x_i) \right) = f (\mathbf{w}^T \mathbf{x})
\]

(3.2)

So a neuron just applies an activation function on the dot product between the inputs and weights, whose structure is shown in Figure 3.1.

The reason to use an activation function is to introduce a non-linearity into the model. It takes one single input \( x \) and perform a non-linear operation on the input. Many different activation functions are proposed.

1. **Sigmoid function.** It computes \( f(x) = \frac{1}{1 + e^{-x}} \) which is shown in Figure 3.2a. Basically it transforms the input to a value between 0 and 1. The sigmoid function is the most widely used activation function. However it gradually loses the interests of the researchers. Theoretically, when the input has very large magnitude, the output approaches to 0 or 1, and its gradient with respect to the input is close to 0. In Section 3.2.4, we introduce how to learn the parameters of a ConvNet, where the 0 gradient makes the learning become difficult. In practice, it performs worse than some other activation functions like the ReLU which will be introduced later.

2. **Tanh function.** For input \( x \), it computes \( f(x) = \tanh x \) which is shown in Figure 3.2b. It transforms the input into to a value between -1 and 1. As the sigmoid function, when the input is very large or very small, the gradient is close to 0. So the tanh function is also not a preferred activation function.

3. **ReLU function.** ReLU is the abbreviation of Rectified Linear Unit. Its mathematician form is \( f(x) = \max(0, x) \). In practice, the ReLU converges faster in learning than the sigmoid or tanh function. And at the same time the calculation of a simple max operation is more efficient than the sigmoid and tanh. So currently ReLU is the most popular activation function. Figure 3.2c shows the ReLU function.
3.2. Neural Network

(a) The Sigmoid activation function.

(b) The Tanh activation function.

(c) The ReLU activation function.

(d) The PReLU activation function.

Figure 3.2: Some activation functions.

4. **PReLU function** [21]. PReLU represents the Parametric ReLU, and it computes \( f(x) = \max(0, x) + a \min(0, x) \), where \( a \) usually is positive and much smaller than 1. If it is set to a constant number, then it changes to LReLU (Leaky ReLU) [22]. Figure 3.2d shows an example of the PReLU function.

3.2.2. **Fully-connected Neural Network**

Usually the neurons in a neural network are organized in layers. A widely used layer type is the fully-connected (fc) layer. If a neural network only consists of fc layers, it is unidirectional, and the input of a layer is the output of the adjacent previous layer. For a \( K \) layer fully-connected neural network, it has one input layer, \( K - 1 \) hidden layers, and one output layer. The neurons in the output layer usually do not have the activation functions, because it will be attached to a loss function (loss layer). An example of the fully-connected neural network with depth 2 is shown in Figure 3.3.

Let the input of a fully-connected neural network be \( \mathbf{x} \), the weights of all layers be \( W = (\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_K) \), and the activation functions of each layer be \( f_1, f_2, ..., f_K \). The whole fully-connected neural network can be seen as a large score function \( s \), which can be written as:

\[
\begin{align*}
    s(\mathbf{x}, W) &= \text{out}_K = f_K(\mathbf{w}_K \cdot \text{out}_{K-1}) \\
    \text{out}_{K-1} &= f_{K-1}(\mathbf{w}_{K-1} \cdot \text{out}_{K-2}) \\
    \vdots \\
    \text{out}_2 &= f_2(\mathbf{w}_2 \cdot \text{out}_1) \\
    \text{out}_1 &= f_1(\mathbf{w}_1 \cdot \mathbf{x}^T)
\end{align*}
\]  

(3.3)
3.2.3. Objective

Overall, as in Equation 3.3, a neural network can be seen as a score function, and the outputs of the output layer are the scores. In the supervised learning, an objective is needed to guide the learning process. Generally, there are two parts in the objective function: a data dependent loss, and a regularization term.

Loss

In general, the loss is used to measure the differences between the prediction and the ground truth, or we can say, measure the correctness of the score function. Depending on the types of the problems to be solved, different losses are used, e.g., the cross-entropy loss and the (squared) hinge loss are two popular classification losses. Assume the training objects are $X = (x_1, x_2, ..., x_N)$, where $N$ is the number of training samples, and $Y = (y_1, y_2, ..., y_N)$ are corresponding labels. Then the cross-entropy loss is:

$$
Loss(X, W) = \frac{1}{N} \sum_{i=1}^{N} \left( -s(x_i, W)_{y_i} + \log \sum_{j \neq y_i} \exp(s(x_i, W)_j) \right)
$$

(3.4)

where $s(x_i, W)_{y_i}$ means the score of the correct class, and $s(x_i, W)_j$ means the score of the $j$-th class. And the hinge loss is:

$$
Loss(X, W) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \left( \max(0, s(x_i, W)_j) - s(x_i, W)_{y_i} + \Delta \right)
$$

(3.5)

where $\Delta$ is a hyperparameter indicating the margin, which usually is set as 1. Similarly, the squared hinge loss is:

$$
Loss(X, W) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \left( \max(0, s(x_i, W)_j) - s(x_i, W)_{y_i} + \Delta \right)^2
$$

(3.6)

The loss function is data dependent, i.e., it measures in what degree the model fits the training data.

Regularization term

The regularization term is used to enhance the generalization power and prevent the overfitting. The two most widely used regularization terms are the $L2$ regularization and the $L1$ regularization. The $L2$
regularization is calculated by \( \text{Reg}(W) = \frac{1}{2} \lambda \| W \|^2 \), where \( \lambda \) is a trade-off hyperparameter controlling the regularization effect. With the increasing of \( \lambda \), the penalty on the loss function will also increase. Roughly the \( L2 \) regularization tends to give big penalties to the big weights, small penalties to small weights.

The \( L1 \) regularization term is defined as \( \text{Reg}(W) = \lambda \| W \|_1 \). Still, \( \lambda \) controls the regularization effect. Because \( \lambda \| W \|_1 \) has a kink when \( W = 0 \), it tends to leave big weights and make small weights to 0. This will result in a more sparse weights selection than the \( L2 \) regularization, which is less appealing in an image related problem because most of the pixels in an image contain some information.

One should note that adding a regularization term in the objective is not the only possible approach to regularize a neural network. Alternatives are to use Maxnorm [24], Dropout [24], or Batch Normalization [25].

**Objective function**

By combining the loss and the regularization term, we can have the final objective function:

\[
\text{Obj}(X, W) = \text{Loss}(X, W) + \text{Reg}(W)
\]

And it can be used to guide the learning process.

**3.2.4. Gradient decent**

The goal of training is to get the best weights that minimize the objective, i.e., the goal of training is to find \( W \) that

\[
\min_W \text{Obj} = \min_W (\text{Loss}(X, W) + \text{Reg}(W))
\]

To find the global minimum, we need to set the first order derivative of the objective to zero:

\[
\frac{\partial \text{Obj}}{\partial W} = 0
\]

and solve the equation. But usually Equation 3.8 cannot be solved analytically, and the gradient decent is used to do the optimization. Gradient decent optimizes the objective step by step, and in each step, it gives a small update to \( W \). The vanilla version of the gradient decent is:

\[
W \leftarrow W - \eta \frac{\partial \text{Obj}}{\partial W} \tag{3.10}
\]

where \( \eta \) is a hyperparameter controlling the step size of each update. In practice, instead of the vanilla gradient decent, people also use Momentum [26], Nesterov Momentum [27], Adagrad [28], RMSprop [29], or Adam [30] to update the weights, which are out of the scope of this report. Besides, when the training data is very large, it is very slow to use all data in calculating the gradients. A practical solution is to use only a part of the data in updating, which is called the mini-batch gradient decent.

By analogy, gradient decent can be compared to a blindfolded man slopes down from the top of a hill. He just needs to step down from the sheerest direction. But because he is blindfolded, when he stops, it is hard to know if this is the bottom. Similarly, using gradient decent usually leads to a local minimum, so properly initializing the weights is very important in practice.

In particular, because neural networks have a layer organization, backpropagation is used to calculate the gradients. For more details about the backpropagation, one can refer to [31].

**3.3. The architecture of a CNN**

**3.3.1. Overview**

The convolutional Neural Network is a specific type of neural network which is designed specifically for image related problems. So it also consists of neurons with weights, biases, and activation functions. The network still has a layer structure, and an objective function combined by the loss and the regularization term is used to guide the learning process. The batch gradient decent is used to optimize the objective function.
The CNN makes a clear assumption that the inputs of a CNN are images instead of some common data. One of the most important natural properties of the image is that it usually has local characteristics, i.e., the pixels in the image are correlated with their neighbors. To catch and learn the local characteristics of the images, the convolutional layer, which is a specific type of layer structure, is designed and used in the CNN. Basically the convolutional layer only connects to some local regions of input, which is different with the fully connection. The local connectivity is the main reason of the CNN’s success in image related problems. In the following sections we will introduce some widely used layer structures in the CNN.

### 3.3.2. Convolutional layer

The name "convolutional layer" comes from that it just computes a convolution of the input, and then an activation function is followed. The weights of a convolutional layer is usually called as a filter or kernel. For example, an input and a filter are shown in Figure 3.4a, where the left top corner of the input is convolved with the filter and the output is in Figure 3.4b. By sliding the filter across all positions of the input, the final output is presented in Figure 3.4c.

![Convolutional layer example](image)

(a) A sample input and the filter.  
(b) The convolutional result of the operation in Figure 3.4a.  
(c) The whole convolutional results.

Figure 3.4: An example of convolution on 2-D data.

The sample input in Figure 3.4a only has two dimensions (height, width), while an image input usually has three dimensions (height, width, depth). For example, an RGB image has three channels, R, G, and B, so its depth is 3. The filter of the convolutional layer works identically on each depth channel to keep the consistency between the channels.

### 3.3.3. Pooling layer

A modern CNN usually contains many layers, e.g., 7, 16, 32, or even more, which makes the whole network have millions of parameters. To train such a network is very slow and difficult, which is the reason why the CNN just becomes popular recently: with the development of high speed computation hardware like the graphics processing unit (GPU), training a big CNN is doable. But the number of parameters still needs to be reduced, so the pooling layer is used.

As the convolutional layer, the pooling layer also operates on a small region of inputs, slides across all positions in the width and height dimensions, and keeps the operations for the depth channels identical. In general, the pooling can be seen as a down-sampling process. Different operations can be used for the down-sampling, like the Max, Sum, Average, and so on. Currently the most widely used pooling is the max pooling. Figure 3.5 gives a visual example how the max pooling works on 2-D data. It is clear that the dimension of data is reduced.

### 3.3.4. Normalization layer

Different normalization schemas can be used as an analogy to lateral inhibition\(^1\) in human brains. One commonly used normalization layer is the local response normalization which is firstly proposed in [32]. However, in some recent experiments, the normalization layers do not have notable contributions to the performance. So they gradually lose the interests of the researchers. As a result, we also do not give a detailed description here.

---

3.3.5. Fully-connected layer
The fully-connected layer in the CNN is identical as in the common neural network which is illustrated in Section 3.2.2. It simply uses all outputs of the previous layer as inputs of the current layer, therefore the outputs of a fully-connected layer can be computed by a matrix multiplication.

3.3.6. AlexNet
In 2012, Alex Krizhevsky et. al submitted the AlexNet [20] to the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), and it beat other submissions at least 10% in classification error. This improvement drew the attention and spurred the research about the CNN. AlexNet has 8 fully-connected/convolutional layers followed by a cross-entropy loss. Its structure is given in Figure 3.6. We use it in our later experiments by the transfer learning which will be introduced in next section.

3.4. Transfer learning
Intuitively, by changing the objective function, the CNN can be used in different tasks. But when the dataset of a task is small, e.g., less than thousands of images, people usually do not directly train a ConvNet from scratch. Because

- If the ConvNet is large and deep, it is very easy to overfit on small dataset.
- If the ConvNet is small and shallow, its performance usually is not very competitive.

In this case, transfer learning can be used to employ pre-trained ConvNets to other tasks. Firstly, a deep ConvNet is trained and tested on a large dataset, for example, the ImageNet dataset which contains 1.2 million images of 1000 categories. Then the pre-trained ConvNet can be transferred to other tasks by two ways:

1. Using the pre-trained ConvNet as a fixed feature extractor. This is the easiest way of using the pre-trained ConvNets in other tasks: ignore the top $k$ ($k$ is smaller than the total number of layers) layers of a ConvNet, and then the rest part can be seen as a fixed feature extractor. For an input image, the output of a truncated pre-trained ConvNet is a feature vector, which can be used in other tasks.

2. Re-train (part) of the ConvNet. The second way not only use the pre-trained ConvNet as a fixed feature extractor, but also tune (part of) the ConvNet. Besides only removing some top layers of a ConvNet, the task specific loss function can be added to the ConvNet. Then based on the task specific loss, (part of) the weights of the pre-trained ConvNet can be fine tuned. Actually, all the weights can be tuned, but usually the low layers’ weights are remained fixed, and only the top layers are tuned. The reason is that in a deep ConvNet, usually the lower layers contain more low level features, e.g., the edge, corner, color, and so on, which are universal for all kinds of images. And the higher layers contain more high level task specific features. So keeping the well-learned low level features fixed and tuning the task specific features is a good choice.
Figure 3.6: The structure of AlexNet.
3.5. Summary

In this chapter, we firstly illustrate the structure of the general neural network, and how to train it. Then we introduce some characteristics of the CNN, and one widely used ConvNet, AlexNet. At last the transfer learning is introduced to show how to use a pre-trained net in different tasks. With the transfer learning, a pre-trained ConvNet can be treated as an image representation extractor, and compared to the traditional SIFT based image representation introduced in Section 2.4.1, it is global, and contains spatial and color information.

As mentioned above, we improve the three aspects of the content based image retrieval: image representation, feature combination, and ranking. In the next chapter, we will introduce two feature combination methods.
As mentioned above, for a typical supervised learning problem, like image classification, each object has a label. However, for the retrieval problem, the label is assigned to a bag of objects, e.g., a query-document pair has a label relevant or irrelevant. So different from classification, there is an additional problem in retrieval: how to combine a pair of objects into a new feature vector? In this chapter, we will introduce two feature combination methods.

4.1. Traditional feature combination methods in text retrieval

As mentioned in Section 2.4.1, an image can be expressed by a vector of visual words, which is analogous to the text retrieval where a document is described by a vector of natural words. As a result, many feature combination methods in text retrieval can be applied to the content based image retrieval. There are two main types of the text based feature combination models: the query dependent, and the query independent models. These models will be described in the coming sections.

4.1.1. Query dependent model

As its name implies, a query dependent model considers both the query and the document when calculating the similarity. Since a document is represented by a vector of words, a straightforward way to combine a query-document pair is to calculate the distance between them. Different types of distance can be used here, like the Euclidean distance, the cosine distance, and the tf-idf weighted cosine distance. Probabilistic models also draw much attention, for example, BM25 [33] or the language model [34].

There are many models proposed in the information retrieval literature, and their principles are not focal points in this report. [35] gives a thorough summarize of the query dependent models, as well as an analysis for each of them. We directly adopt the settings of the models in [35]. In total, 26 features are extracted from two visual word vectors. Their definitions are shown in Table 4.1. Here are some explanations of Table 4.1:

1. \(v_q = (c(t_1, q), c(t_2, q), \ldots, c(t_{|q|}, q))\) is a query vector. \(v_d = (c(t_1, d), c(t_2, d), \ldots, c(t_{|q|}, d))\) is a document vector. \(t_i\) is the \(i\)-th word.

2. \(c(t_i, d)\) represents the number of occurrences of word \(t_i\) in document \(d\).

3. \(|q|\) represents the number of words in \(q\), and \(|d|\) is the the number of words in \(d\).

4. \(t_i \in q \cap d\) means that the word \(t_i\) exists in both \(d\) and \(q\), so both \(c(t_i, d)\) and \(c(t_i, q)\) are not zero.

5. \(D\) is the document database, and \(|D|\) is the total number of documents in the database.

6. \(df\) is the abbreviation of the document frequency, which represents the number of documents that contain the word \(t_i\). Its definition is shown in Equation 2.10.
### Table 4.1: Extracted features.

<table>
<thead>
<tr>
<th>ID</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\sum_{t \in \text{eq}d} c(t_i, d)$</td>
</tr>
<tr>
<td>2</td>
<td>$\sum_{t \in \text{eq}d} (c(t_i, d) \cdot idf_1(t_i))$</td>
</tr>
<tr>
<td>3</td>
<td>$\sum_{t \in \text{eq}d} (c(t_i, d) \cdot idf_2(t_i))$</td>
</tr>
<tr>
<td>4</td>
<td>$\sum_{t \in \text{eq}d} \log (c(t_i, d) + 1)$</td>
</tr>
<tr>
<td>5</td>
<td>$\sum_{t \in \text{eq}d} \log \left( \frac{c(t_i, d)}{</td>
</tr>
<tr>
<td>6</td>
<td>$\sum_{t \in \text{eq}d} \log \left( \frac{c(t_i, d) + 1}{</td>
</tr>
<tr>
<td>7</td>
<td>$\sum_{t \in \text{eq}d} \log \left( \frac{c(t_i, d) \cdot</td>
</tr>
<tr>
<td>8</td>
<td>$\sum_{t \in \text{eq}d} \log \left( \frac{c(t_i, d) \cdot \text{idf}(t_i)}{</td>
</tr>
<tr>
<td>9</td>
<td>$\sum_{t \in \text{eq}d} \log \left( \frac{c(t_i, d) \cdot \text{idf}(t_i)}{</td>
</tr>
<tr>
<td>10</td>
<td>$BM25_1(q, d)$</td>
</tr>
<tr>
<td>11</td>
<td>$BM25_2(q, d)$</td>
</tr>
<tr>
<td>12</td>
<td>$BM25_3(q, d)$</td>
</tr>
<tr>
<td>13</td>
<td>$BM25_4(q, d)$</td>
</tr>
<tr>
<td>14</td>
<td>$\log (BM25_1(q, d))$</td>
</tr>
<tr>
<td>15</td>
<td>$\log (BM25_2(q, d))$</td>
</tr>
<tr>
<td>16</td>
<td>$\log (BM25_3(q, d))$</td>
</tr>
<tr>
<td>17</td>
<td>$\log (BM25_4(q, d))$</td>
</tr>
<tr>
<td>18</td>
<td>$\sum_{t \in \text{eq}d} (c(t_i, q) \cdot c(t_i, d))$</td>
</tr>
<tr>
<td>19</td>
<td>$\sum_{t \in \text{eq}d} \left( \frac{c(t_i, q) \cdot c(t_i, d)}{</td>
</tr>
<tr>
<td>20</td>
<td>$\cos \langle \mathbf{v}_q, \mathbf{v}_d \rangle$</td>
</tr>
<tr>
<td>21</td>
<td>$\sum_{t \in \text{eq}d} \left( c(t_i, q) \cdot idf_1(t_i) \cdot c(t_i, d) \cdot idf_1(t_i) \right)$</td>
</tr>
<tr>
<td>22</td>
<td>$\sum_{t \in \text{eq}d} \left( c(t_i, q) \cdot idf_1(t_i) \cdot c(t_i, d) \cdot idf_2(t_i) \right)$</td>
</tr>
<tr>
<td>23</td>
<td>$\sum_{t \in \text{eq}d} \left( c(t_i, q) \cdot idf_1(t_i) \cdot c(t_i, d) \cdot idf_2(t_i) \right)$</td>
</tr>
<tr>
<td>24</td>
<td>$\sum_{t \in \text{eq}d} \left( c(t_i, q) \cdot idf_2(t_i) \cdot c(t_i, d) \cdot idf_1(t_i) \right)$</td>
</tr>
<tr>
<td>25</td>
<td>$\cos \langle \mathbf{v}_q \circ idf_1, \mathbf{v}_d \circ idf_2 \rangle$</td>
</tr>
<tr>
<td>26</td>
<td>$\cos \langle \mathbf{v}_q \circ idf_2, \mathbf{v}_d \circ idf_2 \rangle$</td>
</tr>
</tbody>
</table>

7. *idf* is the inverse document frequency. *idf*\(_1\) and *idf*\(_2\) are two commonly used definitions:

\[
idf_1(t_i) = \log \left( \frac{|D|}{df(t_i)} \right) \tag{4.1}
\]

\[
idf_2(t_i) = \log \left( \frac{|D| - df(t_i) + 0.5}{df(t_i) + 0.5} \right) \tag{4.2}
\]

8. *BM25 [33]* is a commonly used relevance evaluation method:

\[
BM25(q, d) = \sum_{t \in \text{eq}d} \frac{idf_1(t_i) \cdot c(t_i, d) \cdot (k_1 + 1)}{c(t_i, d) + k_1 \left( 1 - b + b \frac{|d|}{\text{avg}dl} \right) \cdot \frac{k_3 + 1 \cdot c(t_i, d)}{k_3 + c(t_i, d)}} \tag{4.3}
\]

where \(\text{avg}dl = \frac{1}{|D|} \sum_{d \in D} |d|\) is the average length of all documents. \(k_1, k_3,\) and \(b\) are three hyperparameters. Four settings of the hyperparameters are applied:

- For *BM25\(_1\)*, \(k_1 = 2.5, k_3 = 0, b = 0.8\), and *idf*\(_1\) is used.
- For *BM25\(_2\)*, \(k_1 = 2.5, k_3 = 0, b = 0.8\), and *idf*\(_2\) is used.
- For *BM25\(_3\)*, \(k_1 = 1.2, k_3 = 7, b = 0.75\), and *idf*\(_1\) is used.
- For *BM25\(_4\)*, \(k_1 = 1.2, k_3 = 7, b = 0.75\), and *idf*\(_2\) is used.
9. \( \mathbf{v}_q \circ \text{idf} \) is the Hadamard product of two vectors with the same size, i.e.:
\[
\mathbf{v}_q \circ \text{idf} = \left( c(t_1, q)\text{idf}(t_1), c(t_2, q)\text{idf}(t_2), \ldots, c(t_{|\mathcal{T}|}, q)\text{idf}(t_{|\mathcal{T}|}) \right)
\]
(4.4)

10. \( \cos \langle \mathbf{v}_q, \mathbf{v}_d \rangle \) represents the cosine distance of two vectors \( \mathbf{v}_q \) and \( \mathbf{v}_d \).

### 4.1.2. Query independent model

Query independent model only considers the importance of the documents themselves, which is independent with the query. Namely the query is not used in calculating the query independent features. The query independent model intends to give a global measurement to each document in the database. In the text retrieval, the query independent model usually is used to re-rank the top results of the returned list by filtering out some documents which are not important. One of the most famous models is the PageRank [36], which can measure the importance of the webpages by the hyperlink structure.

Query independent models are important in the text retrieval, especially in the webpage search, where the amount of the useful information contained in the documents (webpages) have high variances. Users tend to be only interested in the documents with valuable information. However, for the image retrieval, the variance of the amount of information contained in natural images usually is much lower than in text documents, which means it is hard to determine the importance of the images. As a result, we will not use query independent models in this report.

### 4.2. Distance metric learning

Distance metric learning is also widely used in image retrieval. The literature of “distance metric learning” usually contains the following parts: the definition of the distance metric, the objective function, and the optimization method. Here we only focus on the distance metric because the objective functions are thoroughly studied in the learning to rank literature which will be presented in Chapter 5. For the optimization method, only the stochastic/batch gradient decent is used for simplicity.

Let \( \mathbf{v}_q = (v_{q,1}, v_{q,2}, \ldots, v_{q,|\mathcal{T}|}) \) be the query vector, where \( v_{q,i} \) represents the \( i \)-th element in the query vector \( \mathbf{v}_q \). Similarly, the document vector is \( \mathbf{v}_d = (v_{d,1}, v_{d,2}, \ldots, v_{d,|\mathcal{T}|}) \). Then the distance between the query \( q \) and the document \( d \) is defined by:
\[
\text{dist}(q, d)^2 = \| \mathbf{v}_q - \mathbf{v}_d \|^2 = (\mathbf{v}_q - \mathbf{v}_d)^T A (\mathbf{v}_q - \mathbf{v}_d)
\]
(4.5)

where \( A \) is a \( |\mathcal{T}| \times |\mathcal{T}| \) distance metric. Note that \( A \) should be positive and semi-positive to satisfy the non-negative and triangle inequality requirements that hold for a metric matrix. Then the standard metric learning problem can be expressed by the optimization problem shown in Equation 4.6:
\[
\min_{A \in \mathbb{R}^{|\mathcal{T}| \times |\mathcal{T}|}} \sum_{q \in \mathcal{Q}} \sum_{d \in D_{\mathcal{R}}(q)} \| \mathbf{v}_q - \mathbf{v}_d \|^2_A \\
\text{s.t.} \quad A \succeq 0, \sum_{q \in \mathcal{Q}} \sum_{d \in D_{\mathcal{R}}(q)} \| \mathbf{v}_q - \mathbf{v}_d \|^2_A \geq 1
\]
(4.6)

Now if we let the metric matrix \( A \) be:
\[
A = \begin{bmatrix}
  a_{11} & a_{12} & a_{13} & \ldots & a_{1|\mathcal{T}|} \\
  a_{21} & a_{22} & a_{23} & \ldots & a_{2|\mathcal{T}|} \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  a_{|\mathcal{T}|1} & a_{|\mathcal{T}|2} & a_{|\mathcal{T}|3} & \ldots & a_{|\mathcal{T}||\mathcal{T}|}
\end{bmatrix}
\]

Then we can rewrite Equation 4.5 to:
\[
\text{dist}^2 = a_{11}(v_{q,1} - v_{d,1})^2 + a_{22}(v_{q,2} - v_{d,2})^2 + \ldots + a_{|\mathcal{T}||\mathcal{T}|}(v_{q,|\mathcal{T}|} - v_{d,|\mathcal{T}|})^2 + \\
2a_{12}(v_{q,1} - v_{d,1})(v_{q,2} - v_{d,2}) + 2a_{13}(v_{q,1} - v_{d,1})(v_{q,3} - v_{d,3}) + \ldots + 2a_{|\mathcal{T}||\mathcal{T}|}(v_{q,|\mathcal{T}|} - v_{d,|\mathcal{T}|})(v_{q,|\mathcal{T}|} - v_{d,|\mathcal{T}|}) + \\
w_{1}(v_{q,1} - v_{d,1})^2 + w_{2}(v_{q,2} - v_{d,2})^2 + w_{|\mathcal{T}|+1}(v_{q,1} - v_{d,1})(v_{q,2} - v_{d,2}) + w_{|\mathcal{T}|+2}(v_{q,1} - v_{d,1})(v_{q,3} - v_{d,3}) + \ldots + w_{|\mathcal{T}|+|\mathcal{T}|}(v_{q,|\mathcal{T}|} - v_{d,|\mathcal{T}|})(v_{q,|\mathcal{T}|} - v_{d,|\mathcal{T}|})
\]
(4.7)
where \( \mathbf{w} = (w_1, w_2, ..., w_{|\mathbf{d}|(|\mathbf{d}|-1)}) \) represents the weights vector to be learned. From Equation 4.7 we can see essentially, in distance metric learning, for two objects \( \mathbf{v}_q = (v_{q,1}, v_{q,2}, ..., v_{q,|\mathbf{q}|}) \), and \( \mathbf{v}_d = (v_{d,1}, v_{d,2}, ..., v_{d,|\mathbf{d}|}) \), the combined new object is

\[
\mathbf{v}_{qd} = ((v_{q,1} - v_{d,1})^2, (v_{q,2} - v_{d,2})^2, ..., (v_{q,|\mathbf{q}|} - v_{d,|\mathbf{d}|})^2, \\
(v_{q,1} - v_{d,1})(v_{q,2} - v_{d,2}), (v_{q,1} - v_{d,1})(v_{q,3} - v_{d,3}), ..., (v_{q,|\mathbf{q}|} - v_{d,|\mathbf{d}|})(v_{q,|\mathbf{q}|-1} - v_{d,|\mathbf{d}|-1}))
\]  

(4.8)

Now it is clear that metric distance learning combines feature vectors in the following two ways:

\[
(v_{q,i} - v_{d,i})(v_{q,j} - v_{d,j}) = (v_{q,i} - v_{d,i})^2 \text{ where } i, j \in \{1, ..., |\mathbf{t}|\} \text{ and } i = j
\]  

(4.9)

and

\[
(v_{q,i} - v_{d,i})(v_{q,j} - v_{d,j}) \text{ where } i, j \in \{1, ..., |\mathbf{t}|\} \text{ and } i \neq j
\]  

(4.10)

When the size of the feature space is \( m \), the size of \( \mathbf{v}_{qd} \) becomes \( |\mathbf{q}||\mathbf{d}|-1 \), so the training is very easy to be overfitting. A practical solution is to ignore some items in \( \mathbf{v}_{qd} \). The most common scheme is to only use the items in Equation 4.9, which is also the selection of this paper:

\[
\mathbf{v}_{qd} = ((v_{q,1} - v_{d,1})^2, (v_{q,2} - v_{d,2})^2, ..., (v_{q,|\mathbf{q}|} - v_{d,|\mathbf{d}|})^2)
\]  

(4.11)

As shown in Equation 4.7, the commonly used distance metric just defines a way to construct a new feature vector based on two inputs. Inspired by this insights, and in order to give a contradistinction, we propose another simpler distance metric for two input vectors \( \mathbf{v}_q \) and \( \mathbf{v}_d \) by concatenation:

\[
\mathbf{v}_{qd} = (v_{q,1}, v_{q,2}, ..., v_{q,|\mathbf{q}|}, v_{d,1}, v_{d,2}, ..., v_{d,|\mathbf{d}|})
\]  

(4.12)

In summary, the distance metrics in Equation 4.11 and Equation 4.12 are evaluated in the experiments.

### 4.3. Summary

By using the feature combination methods, two image representations are combined to form a new feature vector. If the feature vector only contains one value, then a simple sorting is enough to generate a ranked list. Otherwise if the feature vector contains more than one value, a supervised method can be used to find the best way of predicting the ranked list, and this can be referred to learning to rank. In next chapter, we will introduce the learning to rank and propose some new ranking losses.
As mentioned above, a query-document pair \( q, d \) can be represented by a feature vector \( v_{q,d} \). If the combined feature vector \( v_{q,d} \) is a single value, a simple sorting can be used to do the ranking. When \( v_{q,d} \) gives a feature vector with multiple elements, some ways to do the ranking has to be found. This is called "learning to rank". In this chapter, we will firstly introduce some key components of machine learning framework, followed by three commonly used approaches in learning to rank. And we also propose some new loss functions for the ranking problem.

### 5.1. Framework of the general learning to rank

Learning to rank algorithms try to solve the ranking problem by discriminative training using the query-document pair inputs. For a query and a document, a vector of features is extracted, and learning to rank algorithms find the optimal way to use them to get a ranking score. There are many components in a general machine learning framework, but in learning to rank we focus on four of them:

- **The input space.** The input space contains the objects to be used by the score function. For example, for image classification problem, the input space is the image pixels or a feature vector extracted from the image.

- **The output space.** The output space usually contains the labels corresponding to the input objects. Depending on the problem type, there are different kinds of output spaces. For example, for a classification problem, the output space is a set of classes, and for a regression problem, the output space is the set of real numbers.

- **The score function.** Essentially, the score function is a function which maps the input objects to some scores, and the type of scores is determined by the output space. For a classification problem, usually the score function calculates the posterior probabilities of the input object for each class. For a regression problem, the score function directly calculates the scores in the output space. For a ranking problem, the score function usually gives a ranking score for the document corresponding to a query.

- **The loss function.** Usually, the training data contains the input objects and the ground truth. The loss function accepts the output of the score function and the ground truth as input, and calculates in what degree the output value of the score function in accordance with the ground truth, i.e., it measures the difference between the predicted value and the ground truth. The loss function is used in the objective to guide the learning process to find the optimal score function.

The learning to rank models can be embedded in the general machine learning framework. Depending on which type of ranking information is available, different approaches can be applied. Generally there are three main approaches: pointwise, pairwise, and listwise.
5.2. Pointwise

For the pointwise approach, the input space of the learning algorithm is the feature vectors $v_{q,d}$ extracted from the query-document pairs. The output space is the relevance level between the corresponding query and document, which can be relevant or irrelevant, or a real value number showing the relevance degree. The score function $s$ accepts a feature vector as input and calculates a score $s(v_{q,d})$ from it.

When the ground truth is relevant or irrelevant, we can assign the label $y_{q,d}$ as:

$$y_{q,d} = \begin{cases} 1, & \text{if document } d \text{ is relevant given query } q \\ -1, & \text{if document } d \text{ is irrelevant given query } q \end{cases} \quad (5.1)$$

Now the ranking problems becomes a classification problem. In this thesis work, the ranking If the ground truth is given by some scalar numbers, the label $y_{q,d}$ can be assigned as:

$$y_{q,d} = \text{the relevance degree between } q \text{ and } d \quad (5.2)$$

And the ranking problem becomes a standard regression problem. Now almost all existing classification/regression algorithms as well as all different classification/regression losses can be easily embedded into this approach.

However, there are some problems for the pointwise approach:

1. No matter what kind of ground truth information is provided, one can always extract the binary label relevant/irrelevant, and treat the ranking problem as a classification problem. But under this condition, sometimes not all information is considered in the algorithm. The relative preferences between documents corresponding to the same query may be ignored. For example, when documents $d_1$ and $d_2$ are both relevant to query $q$, and $d_1$ is more relevant than $d_2$, if pointwise approach is applied, the information that "$d_1$ is more relevant than $d_2$" is lost. This problem is partly solved by the following pairwise learning to rank, and listwise learning to rank.

2. When treating the ranking problem as a regression problem, the relative preferences between documents corresponding to the same query are considered naturally. But in real application, it is much harder to get the real value relevance degree, compared to the relevant/irrelevant binary label. People usually make a trade off and divide the documents into some categories, like "good", "ok", "bad". But in this case, it is not a regression problem yet, and all disadvantages mentioned in the last paragraph still exist.

3. The common loss functions for the classification or regression problem are usually symmetric, i.e., they use the same equation to calculate the loss for each object, where no explicit position information is considered. It is reasonable to doubt some information is lost when using these loss functions for the ranking problem. For example, as mentioned in Section 2.3, the top positions in a ranked list dominate the mAP.

5.2.1. Asymmetric classification loss

As mentioned above, the top positions in a ranked list needs to be focused on in a ranking problem. In detail, as shown in Figure 2.4, when an irrelevant document is classified as relevant and ranks at top position above the relevant documents, the AP decreases greatly. When we treat the ranking problem as a binary classification problem, a straightforward solution is to give the positive and negative objects different weights to prevent the negative objects being misclassified, i.e., prevent the false positive objects.

Here we invent some new asymmetric losses which are based on the symmetric logistic loss. The reason to choose logistic loss is that the later experimental results in Section 7.4 show that the logistic loss is better than other commonly used classification losses. For simplicity, we assume in a binary classification problem, the training object is $x$, the score function is $s$, and the label $y \in -1, 1$. The logistic loss is:

$$Loss_{\text{logistic}} = \frac{1}{\log 2} \log (1 + \exp(-ys(x))) \quad (5.3)$$

The asymmetric losses which will be introduced in this section are listed below.
1. The weighted symmetric loss.
2. The normalized weighted symmetric loss.
3. The powered symmetric loss.
4. The weighted asymmetric loss.
5. The normalized weighted asymmetric loss.
6. The powered asymmetric loss.
7. The weighted nonconvex loss.
8. The normalized weighted nonconvex loss.
9. The powered nonconvex loss.

The first asymmetric loss is the weighted symmetric loss:

\[
L_{\text{weighted-symmetric}} = a \frac{1 - y}{2} \frac{1}{\log 2} \log (1 + \exp (-ys(x))) + \frac{1 + y}{2} \frac{1}{\log 2} \log (1 + \exp (-ys(x))) \quad (5.4)
\]

where a weight hyperparameter \( a \) is added to the logistic loss to give the negative objects larger penalties when it is larger than 1. Now in Equation 5.4, it is clear that when \( y = 1 \), the loss is identical to the symmetric (logistic) loss, and when \( y = -1 \), the loss is multiplied by a weight hyperparameter \( a \). When \( a > 1 \), the penalties for the negative objects are larger. However, when \( y = 1 \) the loss goes through the point \((0, 1)\), and when \( y = -1 \) and \( a \neq 1 \), the weighted loss does not go through \((0, 1)\), which gives some true negative objects larger penalties than expected, and may harm the retrieval performance. So we also propose a normalized version of the weighted symmetric loss to ensure when \( y = -1 \), the loss also goes through the point \((0, 1)\). Specifically, a new parameter \( c \) is added based on \( a \):

\[
L_{\text{normalized-weighted-symmetric}} = a \frac{1 - y}{2} \frac{1}{\log 2} \log (1 + \exp (-y(s(x) + c))) + \frac{1 + y}{2} \frac{1}{\log 2} \log (1 + \exp (-ys(x))) \quad (5.5)
\]

where

\[
c = \log \left( \exp \left( \frac{\log 2}{a} \right) - 1 \right) \quad (5.6)
\]

Besides a simple multiplication, there are different ways of weighing in a loss. Another practical solution is to raise a power to the negative objects, i.e., the powered symmetric loss:

\[
L_{\text{powered-symmetric}} = \frac{1 - y}{2} \left( \frac{1}{\log 2} \log (1 + \exp (-ys(x))) \right)^{\log a} + \frac{1 + y}{2} \frac{1}{\log 2} \log (1 + \exp (-ys(x))) \quad (5.7)
\]

Compared Equation 5.7 to Equation 5.4, it makes sense to estimate that the powered symmetric loss is better. Because by raising a power larger than 1, the false positive objects have larger penalties, and penalties for the true negative objects become smaller. However, by multiplying a weight larger than 1 as in Equation 5.4, both the false positive and true negative objects have larger penalties, where the larger penalties for the true negative objects are not intended.

Besides, note that the powered symmetric loss always goes through point \((0, 1)\) with arbitrary hyperparameter \( a \) so we do not need to normalize it.

Besides simply adding a hyperparameter to give different weights to different objects, another way of manipulating the loss to prevent the false positive objects is to directly use different losses for different objects. So we import an exponential loss and develop the weighted asymmetric loss:
Learning to rank

\[ \text{Loss}_{\text{weighted-asymmetric}} = \frac{a - y}{2} \exp(-ys(\mathbf{x})) + \frac{1 + y}{2} \frac{1}{\log 2} \log(1 + \exp(-ys(\mathbf{x}))) \]
\[ = \frac{1 - y}{2} \exp(\log a - ys(\mathbf{x})) + \frac{1 + y}{2} \frac{1}{\log 2} \log(1 + \exp(-ys(\mathbf{x}))) \] (5.8)

Note that here in Equation 5.8, the parameter \( a \) is used as a trade-off between exponential part and logistic part. It is clear when \( y = 1 \), the weighted asymmetric loss is identical to the logistic loss, and when \( y = -1 \), the exponential part works. As in Equation 5.5, the normalized version is:

\[ \text{Loss}_{\text{normalized-weighted-asymmetric}} = a \frac{1 - y}{2} \exp(-y(s(\mathbf{x}) + c)) \]
\[ + \frac{1 + y}{2} \frac{1}{\log 2} \log(1 + \exp(-ys(\mathbf{x}))) \] (5.9)

where \( c = -\log a \) ensures the loss goes through the point \((0, 1)\) when \( y = -1 \). As in Equation 5.7, the powered asymmetric loss is:

\[ \text{Loss}_{\text{powered-asymmetric}} = \frac{1 - y}{2} \exp(-\log a \cdot ys(\mathbf{x})) + \frac{1 + y}{2} \frac{1}{\log 2} \log(1 + \exp(-ys(\mathbf{x}))) \]
\[ = \frac{1 - y}{2} (\exp(-ys(\mathbf{x})))^{\log a} + \frac{1 + y}{2} \frac{1}{\log 2} \log(1 + \exp(-ys(\mathbf{x}))) \] (5.10)

where \( a > 1 \) controls the base of the exponential, i.e., \( \exp(-\log a \cdot ys(\mathbf{x})) = (\exp(-ys(\mathbf{x})))^{\log a} = a^{-ys(\mathbf{x})} \).

When hyperparameter \( a > 1 \), all the asymmetric losses described above give larger penalties to the false negative objects. However, the above asymmetric losses cannot handle the cases when there are some noises or outliers in the datasets. Specifically, here the outliers means the mislabeled objects. To reduce the impact of the outliers, we propose the weighted nonconvex loss by importing the sigmoid loss:

\[ \text{Loss}_{\text{weighted-nonconvex}} = a \frac{1 - y}{2} \frac{2}{1 + \exp(ys(\mathbf{x}))} + \frac{1 + y}{2} \frac{1}{\log 2} \log(1 + \exp(-ys(\mathbf{x}))) \] (5.11)

When \( y = 1 \), the weighted nonconvex loss is still identical to the logistic loss, while when \( y = -1 \), it is a weighted sigmoid loss which can reduce the impact of outliers because the sigmoid function restricts the loss smaller than a finite limit \((a)\) when \( y = -1 \). The normalized version of the weighted nonconvex loss is:

\[ \text{Loss}_{\text{normalized-weighted-nonconvex}} = a \frac{1 - y}{2} \frac{2}{1 + \exp(y(s(\mathbf{x}) + c))} \]
\[ + \frac{1 + y}{2} \frac{1}{\log 2} \log(1 + \exp(-ys(\mathbf{x}))) \] (5.12)

where \( c = -\log(2a - 1) \). As in Equation 5.7 and Equation 5.10, the powered nonconvex loss is:

\[ \text{Loss}_{\text{powered-symmetric}} = \frac{1 - y}{2} \left( \frac{2}{1 + \exp(ys(\mathbf{x}))} \right)^{\log a} + \frac{1 + y}{2} \frac{1}{\log 2} \log(1 + \exp(-ys(\mathbf{x}))) \] (5.13)

Due to limited time, in the experiment parts, we focus on the comparison between the powered asymmetric loss (Equation 5.10), and the symmetric loss (Equation 5.3), and only compare all candidates proposed above in Section 8.4. The following figures show how the powered-asymmetric loss works:
5.3. **Pairwise**

For the pairwise approach, the input space is a pair of feature vectors extracted from two query-document pairs $v_{q,d_1}$ and $v_{q,d_2}$, associated with the same query $q$. The output space is the pairwise preference between the documents, which means the label $y_{q,d_1,d_2}$ takes value from $\{1, 0, -1\}$, where

$$y_{q,d_1,d_2} = \begin{cases} 
1, & \text{if document } d_1 \text{ is more relevant than document } d_2 \text{ corresponding to } q \\
0, & \text{if document } d_1 \text{ has the same relevance level as document } d_2 \text{ corresponding to } q \\
-1, & \text{if document } d_1 \text{ is less relevant than document } d_2 \text{ corresponding to } q 
\end{cases}$$

Usually, the score function $s$ is defined similar to the score function in the pointwise approach, which means it only accepts one feature vector $v_{q,d}$ extracted from a query-document pair as input, and calculates a single score as output:

$$\text{score} = s(v_{q,d})$$  

---

**Figure 5.1:** The powered asymmetric loss.

(a) The powered asymmetric loss (when $a = e$).

(b) Influence of the hyperparameter $a$.

From Figure 5.1a, the powered asymmetric loss has bigger penalties on the false positive objects compared to false negative data, while the penalties for correctly classified objects are very close no matter they are negative or positive. As a result, when a negative object is misclassified, there is a large loss. So with the powered asymmetric loss, the classifier prefers the strategy to firstly assign enough objects to negative to ensure all negative objects are classified correctly, and then make the remaining objects be positive.

By applying the powered asymmetric loss in ranking problem, compared to the symmetric loss,

- It tends to classify more objects as negative.

Meanwhile,

- For objects which are classified as positive objects, the powered asymmetric loss has higher probability to classify them correctly than the logistic loss.

- For objects which are classified as negative objects, the powered asymmetric loss is similar to the logistic loss. Besides, as shown in Figure 5.1b, when $a$ is larger than $e$, the loss for the true negative objects are much less than the false negative objects, which matches our goal.

Based on the above analyses, the powered asymmetric loss tends to classify less objects as positive, but they have high probability to be predicted correctly. As a result, when the number of relevant documents for a query is small, the powered asymmetric loss should have notable improvements, while with the increasing of the number of relevant documents, the improvements will gradually decrease.
In prediction, after calculating the $s(v_{q,d_1})$ and $s(v_{q,d_2})$, the predicted label:

$$\hat{y}_{q,d_1,d_2} = \text{Indicator}(f(v_{q,d_1}), f(v_{q,d_2}))$$  \hspace{1cm} (5.16)

where

$$\text{Indicator}(s(v_{q,d_1}), s(v_{q,d_2})) = \begin{cases} 
1, & \text{if } s(v_{q,d_1}) > s(v_{q,d_2}) \\
0, & \text{if } s(v_{q,d_1}) = s(v_{q,d_2}) \\
-1, & \text{if } s(v_{q,d_1}) < s(v_{q,d_2}) 
\end{cases}$$ \hspace{1cm} (5.17)

The loss functions for the pairwise approach can be defined in different ways, and here we will introduce four widely used methods: RankSVM, RankNet, FRank, and RankBoost. Note that for a learning method, not only the loss is important, but also the optimization method. In the original literature of the four methods in pairwise approach, corresponding to different losses, different optimization methods are proposed. However, when assembled with different feature combination methods and image representation methods, we will always choose gradient decent as the optimization method. On one hand, gradient decent can handle different loss functions. On the other hand, other optimization methods usually are designed for some specific loss functions, and it is hard to embed them into our framework.

(a) The hinge loss with $\Delta = 1$.  
(b) The cross entropy loss.  
(c) The fidelity loss.  
(d) The exponential loss.

Figure 5.2: Four pairwise loss functions.

**RankSVM.** RankSVM directly embeds learning to rank into the SVM framework. For two input objects $v_{q,d_1}, v_{q,d_2}$, the hinge loss is

$$Loss_{hinge} = \max(0, \Delta - y_{q,d_1,d_2}(s(v_{q,d_1}) - s(v_{q,d_2})))$$ \hspace{1cm} (5.18)
where $\Delta$ is the margin parameter. The curve of $L_{\text{hinge}}$ to $(s(v_{q,d_i}) - s(v_{q,d_j}))$ is shown in Figure 5.2a. The biggest advantage of RankSVM is that it is embedded into the SVM framework which is well studied. Lots of properties of SVM is directly inherited by RankSVM. For example, RankSVM can be optimized by margin maximization, and the kernel tricks can also be applied.

**RankNet.** For RankNet, with two input objects $v_{q,d_i}, v_{q,d_j}$, a probability is defined by $P_q(d_i, d_j) = \exp(s(v_{q,d_i}) - s(v_{q,d_j}))$, and the target probability is:

$$\hat{P}_q(d_i, d_j) = \begin{cases} 1, & \text{if document } d_i \text{ is more relevant than document } d_j \text{ corresponding to } q \\ 0.5, & \text{if document } d_i \text{ has the same relevance level as document } d_j \text{ corresponding to } q \\ 0, & \text{if document } d_i \text{ is less relevant than document } d_j \text{ corresponding to } q \end{cases}$$

(5.19)

Then the cross entropy loss can be defined by

$$L_{\text{cross-entropy}} = -\hat{P}_q(d_i, d_j) \log P_q(d_i, d_j) - (1 - \hat{P}_q(d_i, d_j)) \log (1 - P_q(d_i, d_j))$$

(5.20)

The curve of $L_{\text{cross-entropy}}$ to $(s(v_{q,d_i}) - s(v_{q,d_j}))$ is shown in Figure 5.2b. From the information theory view, the cross entropy can be used to measure the difference between two probabilistic distributions, and when the two distributions are the same, the cross entropy should be zero. However, it is clear in Figure 5.2b the loss function does not have a zero minimum when two documents have the same relevance level. So the training process may not be accurate. Furthermore, the loss is not bounded in a range, so some outliers and hard pairs may have big influences in training. As a result, the cross entropy loss may not be the best choice.

**FRank.** To improve the shortcomings of the cross entropy loss, FRank is proposed which uses the fidelity loss defined in Equation 5.21.

$$L_{\text{fidelity}} = 1 - \sqrt{\hat{P}_q(d_i, d_j)P_q(d_i, d_j)} - \sqrt{(1 - \hat{P}_q(d_i, d_j))(1 - P_q(d_i, d_j))}$$

(5.21)

The curve of $L_{\text{fidelity}}$ to $(s(v_{q,d_i}) - s(v_{q,d_j}))$ is shown in Figure 5.2c. Compared to 5.2b, the fidelity loss always has a zero minimization, and it is bounded in a range between zero to one. These properties give fidelity loss advantages over the cross entropy loss. However, the fidelity loss is nonconvex so it is harder to be optimized. But when the score function $s$ is also nonconvex, it does not make sense to still require the loss function to be convex.

**RankBoost.** RankBoost directly embeds the pairwise learning to rank into the framework of AdaBoost. Skipping over the details of how to find the optimal weak learners and how to combine them, essentially the RankBoost optimizes the following exponential loss:

$$L_{\text{exponential}} = \exp(-y_{d_i,d_j}(s(v_{q,d_i}) - s(v_{q,d_j})))$$

(5.22)

The curve of $L_{\text{exponential}}$ to $(s(v_{q,d_i}) - s(v_{q,d_j}))$ is shown in Figure 5.2d. Compared to the pointwise approach, the pair level preferences are considered in the pairwise approach. But there are still some shortcomings:

1. The biggest disadvantage is the size of the input space. Assume there are $n = |Q| \cdot |D|$ query-document pairs in total. For the pointwise approach, the size of the input space is just $n$. However, for the pairwise approach, the worse case becomes quadratically larger: $n(n - 1)$.

2. For a retrieval problem, the best and the most accurate ground truth we can get is a ranked list of all documents corresponding to the same query. When this kind of information is provided, the pairwise approach will still ignore some information, because no explicit absolute position of document is used in the loss functions.

### 5.4. Listwise

For the listwise approach, the input space contains the feature vectors extracted from all query-document pairs associated with the query, i.e., $\{v_{q,d_i} | i = 1, 2, ..., |D|, q \in Q\}$. And there are two types of commonly used output spaces.
1. The first one is a list of the relevance degrees of all documents associated with the same query \( q \), which means that the label \( y_q \) consists of all relevance degrees between the documents and the query \( q \). If the ground truth relevance degree between a document \( d \) and a query \( q \) is defined by \( g_t(q,d) \), then we have the label:

\[
y_q = \left( g_t(q, d_1), g_t(q, d_2), ..., g_t(q, d_{|D|}) \right)
\]  

(5.23)

Note that here the relevance degrees can be in the form relevant/irrelevant or in the form of a scalar number.

2. The second output space contains the permutations of the documents associated with the query.

Depending on different types of the output space, the score functions of the listwise approach can also be divided into two categories:

1. If the output space is a list of the relevance degrees, the score function \( s \) is the same as in the pointwise and pairwise approaches, i.e., the score function accepts a feature vector \( v_{q,d} \) of a query-document pair as input, and the output is \( s(v_{q,d}) \). Then the scores of all documents associated with the query can be calculated, and a simple sorting is enough to get a ranked list (a permutation):

\[
perm = \text{sort}\left( s\left(v_{q,d_1}\right), s\left(v_{q,d_2}\right), ..., s\left(v_{q,d_{|D|}}\right) \right)
\]  

(5.24)

2. If the output space is the permutation of all documents associated with the query, the score function \( s \) will firstly calculate the scores for all document-query pair, and then output a permutation \( perm \) by a sorting of the scores:

\[
perm = s\left(v_{q,d_1}, v_{q,d_2}, ..., v_{q,d_{|D|}}\right)
\]  

(5.25)

Still, the loss function can also be categorised to two types depending on the output space:

1. If the output space is the relevance degrees, the loss function directly optimizes the evaluation criteria, like the mAP. However, ranking is a position based problem. Similar to the binary classification problem where the "0-1" error is hard to be directly optimized, most of the criteria in ranking are also non-continuous and non-differentiable, so directly optimizing them are difficult. To solve this problem, the loss functions in the listwise approach usually optimize an upper bound of the original criterion.

2. If the output space is the permutation of the documents, the loss function calculates the difference between the predicted permutation and the ground truth one.

When the listwise approach is compared with the pointwise and pairwise approaches, it has an obvious advantage. The listwise approach directly uses the ranked list of all documents associated with the same query for training, so the position information is considered explicitly, which may have some benefits than other approaches.

Now we can see, essentially the above three approaches all accepts a subset of the feature vectors extracted from all query-document pairs associated with the same query, \( \{v_{q,d_i} | i = 1, 2, ..., |D|, q \in |Q|\} \), as input with different sizes. For the pointwise approach, the subset size is 1, and the size of the input space is \( n \). For the pairwise approach, the subset size is 2, with the input size \( n(n - 1) \). And the subset size for listwise approach is \( |D| \), with the input size \( n! \) in the worst case, which is too complex to be handled for large scale retrieval problem.

5.5. Summary

In this chapter, we introduce the definition of learning to rank. Learning to rank uses supervised learning methods to solve the ranking problem, and it can be generally categorized to three approaches, which are pointwise, pairwise, and listwise. Some details of the three approaches, some representative algorithms, and their properties are analyzed. Besides, we also propose some asymmetric pointwise learning to rank losses. Due to limited time, we only evaluate the pointwise approach in the experiments.

In next chapter, we will illustrate how to embed the methods introduced in this and the previous chapters into our framework.
Improving three aspects of a content based image retrieval system

In content based image retrieval, we categorize the whole framework into four parts which is shown in Figure 2.2. They are the image representation, feature combination, ranking, and re-ranking, while in this thesis only the top three parts are considered. Different methods can be easily embedded into the framework. In Section 2.4, we introduce the traditional content based image retrieval method and analyze its shortcomings in Section 2.4.4. Then in Chapter 3, 4, and 5, other alternative approaches for each part are analyzed. We try to improve the performance of the whole system in the following ways:

• **Image representation.** For the image representation, instead of the SIFT based visual words, the CNN is used as an alternative to extract descriptors from images.

• **Feature combination.** Instead of the tf-idf based cosine distance, more different feature combination methods will be tried, which consists of two main types: traditional feature combination methods in text retrieval, and the distance metric learning.

• **Ranking.** If the combined feature of two image representations is a scalar value, a simple sorting is used to rank the results. Otherwise if the combined feature vector contains more than one value, learning to rank is used to find the optimal way to predict the ranking result.

The overall possible methods can be used in three steps are shown in Figure 6.1. In summary, there are 2 possible methods in image representation, 3 possible methods in feature combination, 2 possible methods in ranking algorithms. So in total at least 12 (=2*3*2) different combinations can be tried. But limited by the time, we cannot try all of them. Instead, we propose the following core questions and try to answer them.

1. Is the CNN based image representation better than SIFT based visual words? If so (or if not), in what degree?
2. How to combine multiple image representations? Is the retrieval performance improved by doing so?
3. Is the powered asymmetric loss indeed better than the logistic loss, and can it improve the retrieval performance? Furthermore, when using pointwise learning to rank, what type of classification loss performs best?
4. In general, is the supervised system comparable with the unsupervised system?
5. How to build an end-to-end trainable retrieval system? Can it improve the performance? If so (or if not), why?

To answer these questions, the methods in Table 6.1 are evaluated. In the following part of this section, we will illustrate in detail how to combine the methods mentioned in Table 6.1. Essentially, the whole process is described by Equation 2.1.
Improving three aspects of a content-based image retrieval system

Figure 6.1: Potential methods in the framework.

<table>
<thead>
<tr>
<th>Method</th>
<th>Image representation</th>
<th>Feature combination</th>
<th>Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Visual words</td>
<td>unsupervised similarity measurements (like the cosine similarity)</td>
<td>sorting</td>
</tr>
<tr>
<td>1</td>
<td>CNN</td>
<td>unsupervised similarity measurements (like the cosine similarity)</td>
<td>sorting</td>
</tr>
<tr>
<td>2</td>
<td>Visual words</td>
<td>traditional feature combination methods in text retrieval</td>
<td>pointwise learning to rank</td>
</tr>
<tr>
<td>3</td>
<td>CNN</td>
<td>traditional feature combination methods in text retrieval</td>
<td>pointwise learning to rank</td>
</tr>
<tr>
<td>4</td>
<td>Visual words and CNN</td>
<td>traditional feature combination methods in text retrieval</td>
<td>pointwise learning to rank</td>
</tr>
<tr>
<td>5</td>
<td>Visual words and CNN</td>
<td>traditional feature combination methods in text retrieval</td>
<td>pointwise learning to rank (powered asymmetric loss)</td>
</tr>
<tr>
<td>6</td>
<td>CNN</td>
<td>metric distance learning</td>
<td>pointwise learning to rank</td>
</tr>
</tbody>
</table>

### 6.1. Visual word/CNN + unsupervised similarity measurements + sorting

Methods 0 and 1 can be categorized as the same type. Method 0, which uses the visual word based image representation, unsupervised similarity measurements, and sorting, is described in detail in Section 2.4.

To avoid the shortcomings of the SIFT based visual words mentioned in Section 2.4.4, method 1 uses the CNN as image representation. When a pre-trained ConvNet is used as a fixed feature extractor as shown in Section 3.4, it is a global feature extractor, and for one input image, a fixed length of feature vector can be obtained. For the feature combination methods, we choose and compare different commonly used similarity measurements, like cosine distance, or even tf-idf based cosine distance. At last for the ranking method, similar to method 0, only a simple sorting is enough.

The experimental results of method 0 and method 1 can partially answer question 1.

### 6.2. Visual words/CNN/Visual words and CNN + traditional feature combination methods in text retrieval + learning to rank

Methods 2, 3, 4, 5 and 6 are in the same type just with differences in the image representations.

Methods 2 uses visual words as the image representation. As mentioned in Section 2.4.1, an input image can be represented by a vector of visual words. Because this representation is analogous to words in text retrieval, other techniques and approaches in text retrieval can also be inherited. As a result, for the feature combination method, the traditional feature combination methods in text retrieval mentioned in Section 4.1 can be easily applied. Specifically, for two image representations $\mathbf{v}_q$ and $\mathbf{v}_d$, the 26 features in Table 4.1 are used. Now with two visual words based image representations, a 26 dimension feature vector is obtained, which cannot be handled by a simple sorting. The learning to rank methods is used to get the best ranking. Due to limited time, we only apply the pointwise...
6.2. Visual words/CNN/Visual words and CNN + traditional feature combination methods in text retrieval + learning to rank

approach.

The whole process described above is mature in text retrieval, so it can be easily transformed to image retrieval. But there are still some differences:

- In text retrieval, the term “document” usually represents a webpage which is uniquely defined by a hyperlink. The difference between a webpage and an image makes the feature combination methods also have some differences.
  - A webpage is a structure contains many parts, like header, title, URL, body, anchor, etc.. All of these natural properties can be regarded as features of the document on different levels. However, there is no explicit structures in an image, which results in our case only 26 features in Table 4.1 are used.
  - Webpages in World Wide Web are connected by hyperlinks. This connection can be used in the feature combination method. One of the most famous algorithm is PageRank, which can measure the importance of the webpage itself. Then this information can be used in ranking. However, there are no explicit connections between images, and the methods which measure the documents’ importance in text retrieval cannot be used in image retrieval.

- In text retrieval, usually the query is a word, a phrase, or a sentence, and the documents in the database usually contain much more words than the query. This implies the assumption that the query’s size is much smaller than the documents’ size in text retrieval. However, in image retrieval this generally does not hold anymore. It is common that the size of the query image is larger than the images in the database, or the number of the visual words contained in the query is more than the images to be retrieved. This may cause some algorithms in text retrieval performing bad in image retrieval.

We partially answer the question 1 by comparing the unsupervised performance of different image representations in method 0 and method 1 mentioned above. The influence of different image representations on supervised method is the other side of question 1. As a result, an alternative image representation method, the pre-trained ConvNet, is used. Now the traditional feature combination methods in text retrieval is still used as feature combination method. At last, similarly, learning to rank is used to get the best ranking.

One may doubt if using a CNN and the traditional feature combination methods in text retrieval together makes sense, because the traditional feature combination methods in text retrieval are specifically designed for feature vectors of words. A CNN can be seen as a structure consisting of many filters, whose outputs make up a image representation vector. There are still some reasons to let us use the traditional feature combination methods in text retrieval with the CNN:

- The feature combination methods mentioned in Table 4.1 are essentially methods measuring the similarity between two vectors. They are able to handle different types of input.

- When applying different image representation methods, different representations are extracted from an input image, i.e., they probably have different sizes, scale, sparsities, etc.. The traditional feature combination methods in text retrieval make the fusion of several different representations very easy. For example, let $I_{SIFT}$ denote the SIFT based visual word vector, and $I_{CNN}$ denote the feature vector extracted by a ConvNet. After the same feature combination method $comb$, like the traditional feature combination methods in text retrieval, the combined features of different representations can be concatenated for training or testing a ranking model:

$$
\nu_{q,d} = (comb_i (I_{SIFT}(q), I_{SIFT}(d)), comb_j (I_{CNN}(q), I_{CNN}(d)))
$$  \hspace{1cm} (6.1)

where $comb_i$ and $comb_j$ are feature combination methods, and their outputs are vectors. Actually the $comb_i$ and $comb_j$ do not have to be the same in Equation 6.1. But there are some potential problems of using different feature combination methods. For example, when the output size of $comb_i$ is much larger than $comb_j$, the training may be dominated by $comb_i$. In the experiments, we just select the simplest solution by using the same feature combination method for different image representations.
As mentioned above, with the same feature combination method, different image representation methods can be combined easily. For method 4, 5, we use two image representations: SIFT based visual words, and the descriptors extracted from a pre-trained ConvNet. Equation 6.1 shows the combined feature vector \( v_{q,d} \) from an input query \( q \) and a document \( d \). At last the learning to rank is used to train a ranking model.

As mentioned above, the comparison of method 2 and method 3 can partially answer question 1. And comparing method 4 with method 2, and method 4 with method 3, question 2 can be answered. The results of method 5 and method 4 show the answer of question 3. At last, method 6 answers question 4.

6.3. CNN + distance metric learning + learning to rank

All the feature combination methods used above are unsupervised (Table 4.1). They are easy to compute and do not need to be trained. But there are some limitations:

1. A large part of the unsupervised methods are manually designed for a specific type of problem, which means for different types of data, different image representation methods, or even different datasets, different feature combination methods need to be designed. And this needs some domain specific knowledge and, of course, is challenging.

2. The using of pre-defined feature combination methods implies the image representation methods also have to be hand designed. This splits the whole retrieval system into sub-systems as in Figure 2.2, and only the last part of the system is trainable. Compared to an end-to-end trainable system, it makes sense to guess that the sub-systems make some sacrifices on performance.

As a result, for method 7, we build an end-to-end fully trainable retrieval system by using the supervised feature combination method, i.e., distance metric learning. By given some training data and the objective to be optimized, the distance metric learning can find the best way to combine the features, i.e., the best distance metric. Compared to the unsupervised methods, the parameters in distance metric learning are automatically learned, which has the ability to catch more implicit information of the data. But on the other hand, it takes longer time for training. Furthermore, with distance metric learning, more parameters need to be tuned, which possibly makes the training more difficult to reach the global optimization, and there is a huge risk of overfitting.

In detail, for distance metric learning, Equation 4.11 and Equation 4.12 are used to combine two feature vector \( v_q \) and \( v_d \). It is clear that the combined output \( v_{q,d} \) is a vector. So the following steps should also be supervised, i.e., for ranking method, the only choice is learning to rank. And the CNN is used as image representation method. Based on Section 3.4, there are two ways of using it:

- **Using the pre-trained ConvNet as a fixed feature extractor.** When using the ConvNet as a fixed feature extractor, during the training, the ConvNet will stay constant, and no weights will be updated. However, the ConvNet is trained on some other tasks, so there is risk that the feature extracted from the input is not suitable to the tasks to be solved.

- **Re-train (part) of the ConvNet.** The second way is to tune (part of) the pre-trained ConvNet. In this case, the whole system is a fully trainable end-to-end retrieval system. However, a big ConvNet contains millions of parameters, which requires large training data to avoid overfitting. In our experiment, we only have datasets with thousands of labeled images. As a result, following a commonly used trick, we only tune the top layers of the ConvNet, because the lower layers usually catch some low-level image features which are almost suitable for all kinds of image related problem [17].

By analyzing method 7 and comparing it with method 2, 3, question 5 can be answered.

6.4. Summary

In this chapter, we introduce 8 different ways of embedding the methods mentioned in pervious chapters into the content based image retrieval framework, and their strengths and weaknesses are also be generally analyzed. In the experiments, all of them will be tested on some commonly used datasets.
7 Experiments of different image representations

In this chapter, we firstly introduce the datasets used in Chapter 7, Chapter 8, and Chapter 9. Secondly some common settings for all experiments are illustrated. And then we compare different image representations mentioned above in both unsupervised and supervised experiments to answer question 1 and question 2 in Chapter 6. At last a short summary is given as a overview of the whole chapter.

7.1. Datasets

1. **Oxford5k**[^1] [37] is a building image dataset. There are 5063 images and 55 queries in the dataset. For each query, the images in the database are divided into 4 categories:
   - Good - A nice, clear picture of the object/building. The average number of “Good” images for a query is 24.73.
   - OK - More than 25% of the object is clearly visible. The average number of “OK” images for a query is 26.91.
   - Bad - The object is not present.
   - Junk - Less than 25% of the object is visible, or there are very high levels of occlusion or distortion.

   In the evaluation stage, the images in “Good” and “OK” categories are considered as relevant images, the images in “Junk” category are ignored, and other images are irrelevant. The query image itself in the evaluation stage is also treated as a “Junk” image. On average, about 51.64 images are relevant to a query image.

2. **ukbench**[^2] [38]. This dataset contains 10200 images in total, which are divided into 2550 groups, and each group contains 4 images. So for each image, there are 3 images similar to it. And all images can be used as query. Similarly, in the evaluation stage, the query images are treated as “Junk” and are ignored.

3. **Holidays**[^3] [11] contains 1491 images in total, 500 of them are queries, and the remaining 991 are images in the database to be retrieved. Each of the 991 images in the database is relevant to one and only one query. So for each query, the average number of relevant images is 1.982.

---

[^1]: http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/
[^2]: http://www.vis.uky.edu/~stewe/ukbench/
[^3]: http://lear.inrialpes.fr/~jegou/data.php
[^4]: http://lear.inrialpes.fr/~jegou/data.php
many objects, which makes it is not easy to use one single class to describe the whole image. Furthermore, comparing the queries to their relevant images, it is common that the relevant parts are in the different positions of the images. Specifically, the Oxford5k dataset is more “difficult” than the Holidays dataset. Different from Oxford5k and Holidays dataset, ukbench dataset is more “clear”. A large proportion of the images in ukbench dataset only contain one single object, which even is in the center of the image. The relatively better structures of the images in the ukbench dataset makes it is easier than Oxford5k and Holidays for a retrieval task. Overall, the difficulty of the three datasets are: Oxford5k > Holidays > ukbench.

### 7.2. General experiment settings

In this part, we discuss some common settings and the hyperparameter selections for all experiments.

#### 7.2.1. Image representation

**Bag of visual words based method**

In the bag of visual words based method, the Hessian Affine detector is used to find the key points in an image, and the SIFT descriptor is used to describe each key point. In some cases, instead of the raw SIFT descriptor, a variant called rootSIFT [10] is used. Basically, a visual words vector is a histogram vector, and when calculating the distances between histograms, Hellinger kernel shows better performance than Euclidean distance. The rootSIFT just makes a simple transformation on the raw SIFT descriptors to enable the Hellinger kernel when calculating the distance. For simplicity, we directly use the dictionaries or visual words provided by [37], [13], and [11].

**Pre-trained ConvNet**

The AlexNet shows its competitive performance on the image classification competition ILSVRC2012. So we use a pre-trained AlexNet, and it is trained on the ILSVRC2012 ImageNet dataset. The structure of AlexNet is shown in Figure 3.6. As mentioned in Section 3.4, when using a pre-trained ConvNet as a feature extractor, the top $k$ layers need to be deleted and the output of the $k - 1$ layer is regarded as the representation extracted from the input image. The AlexNet has 5 convolutional layers, which are conv1, conv2, conv3, conv4, conv5, and three fully connected layers, which are fc6, fc7, fc8. In the experiments, the outputs of layers conv3, conv4, conv5, fc6, fc7, and fc8 are used as descriptors of the input image and are compared to each other. The sizes of the output of layers mentioned above are shown in Table 7.1.

When the pre-trained ConvNet is tuned, the learning rate for the newly added layers are set 10 times larger than the learning rate of the pre-trained part. The reason is the pre-trained part only needs to be fine tuned, and the newly added layers need a much larger learning rate because they are initialized in random value.

#### 7.2.2. Feature combination

**Traditional feature combination methods in text retrieval**

For two image representation vectors, a new 26 dimension feature vector is obtained by feature combination methods in Table 4.1. Although the methods in Table 4.1 is originally designed for vectors of words, as mentioned in Section 6.2, we still apply them on other image representations, i.e., the representations extracted from a pre-trained ConvNet.

**Distance metric learning**

From Section 7.1, the datasets to be tested are not large enough to train a full distance metric matrix in Equation 4.8. For example, when the output of layer fc6 of a pre-trained ConvNet is used as image representation, for an input image, a 4096 dimension feature vector can be obtained. To learn a full distance metric matrix, in total $\frac{(4096+1)4096}{2} = 8390656$ parameters need to be learned, which is much larger than the datasets’ sizes we use in the experiments. From Section 7.1, the biggest dataset
7.3. Visual word/CNN + unsupervised similarity measurements + sorting

In this section, we test the unsupervised methods on all datasets as baseline. The results are shown in Table 7.2, 7.3, 7.4 and 7.5.

Firstly, this section does not intent to find out what kind of image representation is the best for image retrieval problem, which is a challenging problem and needs more intensive experiments. The goal is just to show that different image representations have big influence on the mAP, for example, the best performance for Holidays dataset with visual word image representation is 55.53%, while using the CNN image representations it reaches 73.12%. Also for ukbench dataset, the best performance with the CNN is 83.40%, which is larger than the visual word based performance, 77.1%. On the other hand, the mAP for Oxford5k dataset with visual word (1M dictionary) is 61.72%, which is larger than the best mAP (47.39%) when using the CNN image representations. Now the question 1 in Chapter 6 can be answered: the CNN based image representation is not always better than SIFT based bag of visual words for image retrieval problem. The performance highly depends on the datasets and the

<table>
<thead>
<tr>
<th>Dataset</th>
<th>20K dictionary</th>
<th>200K dictionary</th>
<th>1M dictionary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oxford5k</td>
<td>33.26%</td>
<td>38.4% [39]</td>
<td>61.72%</td>
</tr>
<tr>
<td>ukbench</td>
<td>65.15%</td>
<td>77.1% [39]</td>
<td>-</td>
</tr>
<tr>
<td>Holidays</td>
<td>51.19%</td>
<td>55.53%</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 7.2: mAP of the visual words based image representations.

<table>
<thead>
<tr>
<th>fc6</th>
<th>fc7</th>
<th>fc8</th>
</tr>
</thead>
<tbody>
<tr>
<td>41.87%</td>
<td>38.13%</td>
<td><strong>37.09%</strong></td>
</tr>
<tr>
<td>40.58%</td>
<td>36.61%</td>
<td>37.08%</td>
</tr>
<tr>
<td>41.64%</td>
<td>37.28%</td>
<td>36.37%</td>
</tr>
<tr>
<td>47.39%</td>
<td>41.56%</td>
<td>32.83%</td>
</tr>
<tr>
<td>45.58%</td>
<td><strong>41.58%</strong></td>
<td>36.02%</td>
</tr>
</tbody>
</table>

Table 7.3: Oxford5k. mAP of the CNN based image representations.

(ukbench) contains only 10200 images. As a result, we only learn the simplified distance metric matrices as shown in Equation 4.11 and Equation 4.12.

7.2.3. Learning to rank

We only use the pointwise approach in learning to rank. Besides, for simplicity, the ranking problem is treated as a binary classification problem. The following classifiers are tried: logistic regression, linear SVM, naive bayes classifier, quadratic classifier, fisher classifier, perceptron classifier, polynomial classifier, nearest mean classifier, scaled nearest mean classifier, decision tree classifier, RandomForest, Adaboost, Gradient Boosted Decision Tree (GBDT). The above classifiers can be generally divided into two categories: RandomForest, Adaboost, and GBDT are complex non-linear classifiers, and the others are simple classifiers.

Besides, the proposed asymmetric losses in Section 5.2.1 are also evaluated.

7.3. Visual word/CNN + unsupervised similarity measurements + sorting

In this section, we test the unsupervised methods on all datasets as baseline. The results are shown in Table 7.2, 7.3, 7.4 and 7.5.

<table>
<thead>
<tr>
<th>fc6</th>
<th>fc7</th>
<th>fc8</th>
</tr>
</thead>
<tbody>
<tr>
<td>81.73%</td>
<td>77.34%</td>
<td>78.65%</td>
</tr>
<tr>
<td>81.09%</td>
<td>77.49%</td>
<td><strong>78.81%</strong></td>
</tr>
<tr>
<td>81.54%</td>
<td>77.49%</td>
<td>78.72%</td>
</tr>
<tr>
<td>81.07%</td>
<td>75.03%</td>
<td>78.07%</td>
</tr>
<tr>
<td><strong>84.40%</strong></td>
<td>82.15%</td>
<td>77.61%</td>
</tr>
<tr>
<td>84.36%</td>
<td><strong>83.13%</strong></td>
<td>77.92%</td>
</tr>
</tbody>
</table>

Table 7.4: ukbench. mAP of the CNN based image representations.
Table 7.5: Holidays. mAP of the CNN based image representations.

<table>
<thead>
<tr>
<th></th>
<th>conv3</th>
<th>conv4</th>
<th>conv5</th>
<th>fc6</th>
<th>fc7</th>
<th>fc8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unnormalized cosine distance, without ReLU</td>
<td>59.35%</td>
<td>59.78%</td>
<td>61.12%</td>
<td>66.49%</td>
<td>66.30%</td>
<td>64.40%</td>
</tr>
<tr>
<td>Unnormalized cosine distance, with ReLU</td>
<td>57.48%</td>
<td>59.16%</td>
<td>60.28%</td>
<td>66.10%</td>
<td>63.43%</td>
<td>63.58%</td>
</tr>
<tr>
<td>Normalized cosine distance, without ReLU</td>
<td><strong>61.42%</strong></td>
<td>60.69%</td>
<td>61.08%</td>
<td>71.89%</td>
<td>71.46%</td>
<td>65.47%</td>
</tr>
<tr>
<td>Normalized cosine distance, with ReLU</td>
<td>60.05%</td>
<td>60.03%</td>
<td>60.96%</td>
<td>71.76%</td>
<td>71.63%</td>
<td>65.80%</td>
</tr>
<tr>
<td>Normalized cosine distance, without ReLU, tf-idf weighted</td>
<td>60.88%</td>
<td><strong>60.81%</strong></td>
<td>61.13%</td>
<td>72.28%</td>
<td>72.05%</td>
<td>63.11%</td>
</tr>
<tr>
<td>Normalized cosine distance, with ReLU, tf-idf weighted</td>
<td>59.60%</td>
<td>60.07%</td>
<td><strong>62.41%</strong></td>
<td><strong>73.12%</strong></td>
<td><strong>72.29%</strong></td>
<td><strong>65.10%</strong></td>
</tr>
</tbody>
</table>

implementation details of visual words and CNN. In general, if the datasets contain more details, and the average number of the relevant images for a query is large, using the visual words based image representations with a large dictionary is better.

Secondly, the each image representations themselves also show some general properties.

- Visual words. From Table 7.2, we can see in general for visual word image representation, the performance highly depends on the size of the dictionary. When the dictionary size becomes larger, the whole retrieval process is more similar to a “real” nearest neighbor search. And consequently, the performance becomes better. But on the other hand, a large dictionary makes the computation less efficient. Furthermore, as mentioned in [11], a large dictionary is not so robust to the noise.

- CNN.
  - In Table 7.5, the performances of fully connected layers are much better than the convolutional layers. This suggests that the convolutional layer usually catches some low level image features, like lines, corners [40], while the fully connected layer is in higher level. As a result, the outputs of the fully connected layers usually are more preferable because they catch more global and high level information of the input image.
  - In Table 7.3, Table 7.4, and Table 7.5, the best performances are all provided by outputs of layer fc6. Moreover, for three fully connected layers, their performances are \( fc6 > fc7 > fc8 \) with the same weighting scheme and distance calculation method. The pre-trained ConvNet we use is trained on ImageNet dataset, which is a image classification dataset and contains 1000 classes in total. So the last layer (fc8) directly shows which class the input image is. However, for our retrieval problem, the images in the dataset usually contain many different objects, or do not belong to the 1000 classes of ImageNet. As a result, the layer fc8 performs worst in all fully connected layers.
  - For layers fc6 and fc7, the best performances are obtained by the normalized cosine distance with tf-idf weighted. This means the traditional manually designed method is indeed suitable for the retrieval problem.

### 7.4. Visual words + traditional feature combination methods in text retrieval + pointwise

Because we do not have enough time to perform all experiments on all datasets, the experiments in this section, Section 7.5, and Section 7.6 are done only on Holidays dataset.

We firstly try to roughly figure out for pointwise learning to rank, what kind of classification loss is a good choice. Since this is a supervised problem, we need to make the training/validation/testing data. For simplicity, we use top 60% queries and all images in the Holidays datasets to construct the training data. Then the following 20% queries are used to construct the validation data. And remaining queries are used for the testing data. Figure 7.1 shows the above data splitting method visually. In Holidays dataset, this means the top 300 queries are used to construct training data, the following 100 queries are for validation data, the remaining 100 queries are for testing data. In total, there are 745500 objects at all, where only 1491 objects are positive showing its high imbalance.

The image representation is visual word (20K dictionary) based feature vectors. The traditional feature combination methods in text retrieval is used as feature combination method. Some binary classifiers are tested. By using the coarse data splitting method mentioned above, we firstly filter out some promising classifiers, and then analyzing them in detail. The results are shown in Table 7.6.
7.4. Visual words + traditional feature combination methods in text retrieval + pointwise

Figure 7.1: Data split.

Table 7.6: Results of some pointwise classifiers.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>mAP (train)</th>
<th>mAP (val)</th>
<th>mAP (test)</th>
<th>Training time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic regression</td>
<td>60.15%</td>
<td>61.42%</td>
<td><strong>61.75%</strong></td>
<td>6.62</td>
</tr>
<tr>
<td>Linear SVM</td>
<td>32.63%</td>
<td>34.22%</td>
<td>33.13%</td>
<td>158.15</td>
</tr>
<tr>
<td>Naïve bayes classifier</td>
<td>33.77%</td>
<td>29.70%</td>
<td>36.18%</td>
<td>0.99</td>
</tr>
<tr>
<td>Quadratic classifier</td>
<td>17.24%</td>
<td>19.03%</td>
<td>20.09%</td>
<td>0.78</td>
</tr>
<tr>
<td>Fisher classifier</td>
<td>56.50%</td>
<td>52.25%</td>
<td>59.44%</td>
<td>3.03</td>
</tr>
<tr>
<td>Perceptron classifier</td>
<td>54.52%</td>
<td>51.45%</td>
<td>57.05%</td>
<td>15.67</td>
</tr>
<tr>
<td>Polynomial classifier</td>
<td>56.50%</td>
<td>52.25%</td>
<td>59.44%</td>
<td>7.09</td>
</tr>
<tr>
<td>Nearest mean classifier</td>
<td>5.97%</td>
<td>5.07%</td>
<td>4.29%</td>
<td>1.05</td>
</tr>
<tr>
<td>Scaled nearest mean classifier</td>
<td>32.60%</td>
<td>32.72%</td>
<td>39.34%</td>
<td>1.41</td>
</tr>
<tr>
<td>Decision tree classifier</td>
<td>44.47%</td>
<td>43.12%</td>
<td>45.83%</td>
<td>34.85</td>
</tr>
<tr>
<td>Random Forest</td>
<td>98.74%</td>
<td>52.40%</td>
<td>55.31%</td>
<td>964.91</td>
</tr>
<tr>
<td>Adaboost</td>
<td>53.68%</td>
<td>49.05%</td>
<td>53.84%</td>
<td>589.47</td>
</tr>
<tr>
<td>Gradient Boosted Decision Tree (GBDT)</td>
<td>89.78%</td>
<td>54.23%</td>
<td>51.01%</td>
<td>195.95</td>
</tr>
</tbody>
</table>

The “mAP (test)” column in Table 7.6 needs to be noticed, from it, we can know:

- Logistic regression outperforms all other classifiers for at least 2%. That is why we develop the asymmetric losses based on the logistic loss in Section 5.2.1.

- As mentioned in Section 7.2.3, the binary classifiers are roughly divided into two types: complex and simple classifiers. From Table 7.6, three complex classifiers (Random Forest, Adaboost, and GBDT) are worse than some simple classifiers, which possibly shows on one hand that the features we have now are “good enough” for a simple classifier. And on the other hand, this implies the training data, especially the positive data, are not enough, so a simple classifier can already get a satisfactory performance, and using a complex one cannot expose more information of the data, but is easy to be overfitting. Actually the Random Forest and GBDT are indeed overfitting.

- The mAPs for training, validation, and testing data of most classifiers are on the same level, which possibly means the learning curves have already converged.

Based on the results in Table 7.6 and the training time, we select some classifiers to draw the learning curves to further check if the learning curves have already converged, and how many training objects are needed for this dataset. The selected classifiers are logistic regression, naive bayes classifier, fisher classifier, perceptron classifier, polynomial classifier, and the scaled nearest mean classifier. The learning curves are shown in Figure 7.2. In each sub-figure of Figure 7.2, the blue line represents the training accuracy, and the red line represents the testing accuracy. The number of training queries are 2, 5, 10, 20, 50, 100, 200, 300, 500.

From Figure 7.2, it is clear still in general logistic regression performs best. And after the learning curve of logistic regression converges, the mAP is around 60%, which is better than the unsupervised baseline (51.19% in Table 7.2). More importantly, most of the learning curves in Figure 7.2 converge when the number of training queries are only 100~200, which is very fast. It suggests that with current
Experiments of different image representations

(a) The logistic regression.
(b) The naive bayes classifier.
(c) The fisher classifier.
(d) The perceptron classifier.
(e) The polynomial classifier.
(f) The scaled nearest mean classifier.

Figure 7.2: Learning curves for the visual words based image representation.

features, a good performance can be reached only with a small amount of training objects, which is very useful in the real world applications.

7.5. CNN + traditional feature combination methods in text retrieval + pointwise

Inspired by the results in last section, we use the CNN as the image representation and make the learning curve figures with the same classifiers used in last section. Figure 7.3 shows the results of using the output of layer \( fc6 \) as image representation. Comparing Figure 7.3 to Figure 7.2, some insights still hold:

- The logistic regression outperforms other classifiers after converging.
- The convergences in Figure 7.3 still happen in the early stage, and this means a good classifier can be trained without many objects available.

The same phenomenon are also observed when using the output of layer \( fc7 \) or layer \( fc8 \) as image representation. The learning curves of all classifiers are put in the Appendix (Figure A.1, and Figure A.1). Here we only show the learning curves of the best classifier, i.e., logistic regression, of layer \( fc7 \) and layer \( fc8 \) in Figure 7.4.

In Table 7.5, the best unsupervised mAP for \( fc6, fc7, fc8 \) are 65.80%, 72.29%, 73.12%, respectively. Comparing to the corresponding supervised results, i.e., results of the logistic regression, in Figure 7.3a, Figure 7.4a, and Figure 7.4b, they are basically on the same level. The fact that the supervised method does not have a notable improvement for the CNN based image representations confirms our doubt mentioned in Section 6.2: the traditional feature combination methods in text retrieval may not be the most suitable choice for other kinds of representations instead of the word based representations. But anyway, the traditional feature combination methods also do not harm the performance. Since at the same time it provides a very convenient way of feature fusion, i.e. by a simple concatenation shown in Equation 6.1, it is still very useful. The fused features are tested in the next section.
7.6. Visual word and CNN + traditional feature combination methods in text retrieval + pointwise

In Section 7.4 and Section 7.5, we analyze the performances of the supervised method on the visual word and CNN based image representations individually. The results show that for visual words based image representation, there is a notable improvement, while for CNN, the performances are on the same level to the unsupervised method. One big advantage of the traditional feature combination methods in text retrieval is that it provides a straightforward way of feature fusion by concatenation. In this section, we analyze the influence of the fused feature by drawing the learning curves of logistic regression in Figure 7.5. The feature fusion follows Equation 6.1, and the \( \text{combs} \) in Equation 6.1 we use here are methods described in Table 4.1. As a result, the objects in Figure 7.5a, 7.5b, and 7.5c are 52 (=26*2) dimensions, and the objects in Figure 7.5d are 104 (=26*4) dimensions. Note that we also make the learning curves of other selected classifiers (naïve bayes classifier, fisher classifier, perceptron classifier, polynomial classifier, and the scaled nearest mean classifier) in Appendix A.2. Results show that for all type of fused features, the logistic regression always has the best performance after convergence.

Comparing Figure 7.5a with Figure 7.2a and Figure 7.3a, it is clear after converging, the performance of the concatenation of two kinds of representations is better than each of them individually. In detail,
Experiments of different image representations

(a) Visual word + CNN (fc6).

(b) Visual word + CNN (fc7).

(c) Visual word + CNN (fc8).

(d) Visual word + CNN (fc6, fc7, fc8).

Figure 7.5: Learning curves of the logistic regression for the visual words + CNN based image representation.

After converging the mAP for testing data is close to 80% in Figure 7.5a, which is better than 60% in Figure 7.2a, and 70% in Figure 7.3a. The same results hold for the fused features of visual word + CNN (fc7), and visual word + CNN (fc8). These results show that

- The visual words based image representation and the CNN based image representation catch different information of an image.
- The straightforward feature fusion methods based on the traditional feature combination methods in text retrieval can indeed improve the performance.

In Figure 7.5, except Figure 7.5d, all learning curves converge in an early stage where the number of training queries is smaller than 200. One may doubt why Figure 7.5d converges slower than other sub-figures in Figure 7.5, and the stable mAP for testing data does not have a clear advantage to Figure 7.5a, or Figure 7.5b. The reason possibly is we directly adopt the best hyperparameters found in Section 7.4 without a fine tune for simplicity.

7.7. Summary

In this chapter, we firstly introduce the datasets used in the experiments, followed by some general experiment settings. Then the experiments in this chapter compare different image representations, i.e., the visual word based and the CNN based representations, with both unsupervised and supervised methods.

Question 1 and 2 in Chapter 6 are answered. For question 1, the answer is the image representation selection highly depends on the datasets, and the detailed settings of the representations. In detail, when the average number of relevant images for a query is large, using the visual words based image representation method with a large dictionary is a better choice. On the other hand, the CNN based
image representation usually performs better. For question 2, firstly the fusion of multiple image representations can be done by a concatenation. Secondly, the fused features should and indeed improve the performance with different image representations. And possibly the improvements depends on the degree of differences between them.
Experiments of the asymmetric losses

From the above chapter, the logistic regression is the best common binary classifiers for image retrieval problem with the pointwise learning to rank. That is why we propose some improvements in Section 5.2.1 based on the logistic loss. As mentioned in Section 5.2.1, the top positions in the ranking problem have dominated influences on the final performance. The asymmetric losses is specifically designed to make the top positions have higher probabilities to be correct. The experiments in this chapter are done on all three datasets to answer question 3 in Chapter 6. In detail, for each dataset, the cross validation is used to compare the performances of the powered asymmetric loss (Equation 5.10) and the logistic loss (Equation 5.3). In Section 8.4, different asymmetric losses are compared to show what is the key in designing a good asymmetric loss for the retrieval problem. At last, the transfer ability of the whole system is tested to answer the question 4 in Chapter 6 by training a classifier on one dataset and testing it on another one.

8.1. Holidays

Referring to the learning curves in last chapter (Figure 7.5), and considering the training time, we select 50 queries to construct the training data, and the others as testing data. To get the best performance, the visual word (20K dictionary), the CNN (fc6, fc7, fc8) are used as image representations to produce the 104 dimensions fused features for training and testing. As mentioned above, in Holidays dataset, on average each query has 1.982 relevant images in the database. In this case, the top positions of the ranked list dominate the AP. So the powered asymmetric loss should have a better performance than the logistic loss. This anticipation is proved by the cross validation results in Table 8.1, and Table 8.2.

Table 8.1: Accuracies of cross validation on Holidays dataset. The bold rows need to be focused on.

<table>
<thead>
<tr>
<th>Loss</th>
<th>All objects</th>
<th>Positive objects</th>
<th>Negative objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>powered asymmetric</td>
<td>99.9080±0.0111%</td>
<td>55.1536±3.1620%</td>
</tr>
<tr>
<td></td>
<td>logistic</td>
<td>99.8978±0.0141%</td>
<td>49.2974±3.9855%</td>
</tr>
<tr>
<td>Test</td>
<td>powered asymmetric</td>
<td>99.9059±0.0049%</td>
<td>54.2779±2.2206%</td>
</tr>
<tr>
<td></td>
<td>logistic</td>
<td>99.8939±0.0041%</td>
<td>48.3343±1.7900%</td>
</tr>
</tbody>
</table>
Table 8.2: AUCs and mAPs of cross validation on Holidays dataset. The bold rows need to be focused on.

<table>
<thead>
<tr>
<th>Loss</th>
<th>exp+log</th>
<th>logistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUC</td>
<td>Training data</td>
<td>55.1536±3.1620%</td>
</tr>
<tr>
<td>Testing data</td>
<td>0.93%±0.07%</td>
<td>1.23%±0.13%</td>
</tr>
<tr>
<td>mAP</td>
<td>Training data</td>
<td>79.23%±2.03%</td>
</tr>
<tr>
<td>Testing data</td>
<td>78.77%±1.70%</td>
<td>76.69%±0.88%</td>
</tr>
</tbody>
</table>

Figure 8.1: Average differences of AP of the powered asymmetric loss and the logistic loss corresponding to the amount of relevant images a query has. The blue line shows the average differences of APs to the amount of relevant documents. The green line shows the number of queries with the corresponding amount of relevant documents.

The experiments of the asymmetric losses

Table 8.2 shows the AUC and mAP for both losses. It is clear that for testing data, the powered asymmetric loss has better performances on both AUC and mAP. As a result, we can basically conclude that at least for Holidays dataset, the powered asymmetric loss is better than the logistic loss.

Figure 8.1 roughly shows how powered asymmetric loss improves the AP compared to the logistic loss. The average differences between APs of powered asymmetric loss and logistic loss corresponding to the queries with the same amount of relevant images are indicated by the blue line in Figure 8.1. And the green line shows the number of queries for a specific amount of relevant images. From Figure 8.1, we can see when the amount of relevant images is smaller than 2, the powered asymmetric loss and logistic loss have very similar performances. Then with the increasing of the amount of relevant images, the powered asymmetric loss gradually has notable improvements compared to logistic loss. At last when the amount of relevant documents is larger than 6, the number of queries are very small, which results in big noises and cannot be analyzed statistically. So in general we can conclude the powered asymmetric loss improves the retrieval performance when the amount of relevant images to a query is about 3~5. And this is also proved in the experiments on other datasets later.

We also check the influence of the hyperparameter $a$, and the results are shown in Figure 8.2. By checking the sub-figures 8.2b, 8.2e, 8.2g, 8.2h, and 8.2i, it is clear that there is a turnover when $a$ is close to $1 + 1e^{-2}$. When $a < 1 + 1e^{-2}$, the train/test accuracies for positive objects are higher, and the train/test accuracies for negative objects are smaller than 1. However when $a > 1 + 1e^{-2}$, the train/test accuracies for negative objects are extremely close to 1, and the train/test accuracies for positive objects become smaller. This means that when $a$ is very small (close to 1), the logistic part in the powered asymmetric loss prevails over the exponential part. However when $a$ becomes larger, the exponential part gradually dominates the loss and makes the whole loss work as a retrieval loss. One may doubt why this turnover happens when $a$ is very small (much smaller than $e$), and the reason probably is the number of negative objects (744009=745500-1491) is much larger than the positive objects (1491). Furthermore, in sub-figure 8.2i we can see after $a > 1 + 1e^{-2}$, by increasing $a$, the mAP also increases.
8.1. Holidays

The biggest drawback for the powered asymmetric loss is that it imports a new hyperparameter $a$, and tuning this parameter needs additional work. The basic principle for this parameter is that it should be set large enough to ensure the classifier classify all negative objects correctly. Another shortcomings is that in the powered asymmetric loss, there is an exponential term, so the data needs to be pre-processed carefully to avoid numerical problems. Especially as shown in Figure 8.2i, $a$ needs to be set extremely large, i.e., $1 \times 10^{12}$ or larger, to get a good mAP. Under this condition, the training process takes longer time and is very easy to have numeric problems. So one has to repeat the training many times with different initialization settings or with more careful pre-processing.

At last, Figure 8.3 shows some qualitative results where the powered asymmetric loss outperforms the logistic loss for queries with different amount of relevant images. In each sub-figures, the leftmost column shows the query image, and the query is followed by the top retrieved images. The first row shows the ground truth relevant images for the query. The second and third rows present the retrieval results of using the logistic loss and the powered asymmetric loss, respectively. From Figure 8.3, it is more straightforward to see that the powered asymmetric loss makes improvements on logistic loss by retrieving the top positions correctly.

Figure 8.4 has a similar structure to Figure 8.3 but shows the queries on which both the powered asymmetric loss and the logistic loss fail. Basically there are two types of situations both losses fails:

1. Some queries, like in Figure 8.4a, and 8.4b, are even difficult for human beings. For these queries, the ground truth relevant images contain exactly the same objects as in the queries. However, the wrong retrieved images usually contain the different objects from the same class with the
Experiments of the asymmetric losses

Table 8.3: Accuracies of cross validation on ukbench dataset. The bold rows need to be focused on.

<table>
<thead>
<tr>
<th>Loss</th>
<th>exp+log</th>
<th>logistic</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Train</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All objects</td>
<td>99.9816%±0.0041%</td>
<td>99.9810%±0.0016%</td>
</tr>
<tr>
<td>Positive objects</td>
<td>93.3431%±10.4141%</td>
<td>49.2974%±3.9256%</td>
</tr>
<tr>
<td><strong>Negative objects</strong></td>
<td>99.9999%±0.0001%</td>
<td>99.9985%±0.0004%</td>
</tr>
<tr>
<td><strong>Test</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All objects</td>
<td>99.9807%±0.0039%</td>
<td>99.9793%±0.0008%</td>
</tr>
<tr>
<td>Positive objects</td>
<td>54.2333%±7.9079%</td>
<td>55.3949%±4.0351%</td>
</tr>
<tr>
<td><strong>Negative objects</strong></td>
<td>99.9878%±0.0016%</td>
<td>99.9968%±0.0019%</td>
</tr>
</tbody>
</table>

Table 8.4: AUCs and mAPs of cross validation on ukbench dataset. The bold rows need to be focused on.

<table>
<thead>
<tr>
<th>Loss</th>
<th>exp+log</th>
<th>logistic</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AUC</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Training data</td>
<td>0.0259%±0.0275%</td>
<td>0.0409%±0.0362%</td>
</tr>
<tr>
<td>Testing data</td>
<td>0.0270%±0.0032%</td>
<td>0.0431%±0.0024%</td>
</tr>
<tr>
<td><strong>mAP</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Training data</td>
<td>93.97%±2.97%</td>
<td>90.63%±0.09%</td>
</tr>
<tr>
<td>Testing data</td>
<td>93.49%±0.48%</td>
<td>89.97%±0.39%</td>
</tr>
</tbody>
</table>

objects in the queries, which are hard to be distinguished. For example, it is very difficult to get the differences between two different but visually similar tropical fishes.

2. Figure 8.4c, 8.4d, and 8.4e give another type of situation where both losses fails: there are big scale/rotation/angle/cluttering differences between the queries and their relevant images. In those cases, our retrieval system puts the images with similar scenes to the queries in the top positions, which results in bad performances.

From the analysis of two situations above, we can see the failing is not about the classifiers/losses, but is more related to the image representations. The current system uses two types of image representation: the SIFT based visual words, and the CNN. They either only make descriptions on some key points in the image (SIFT), or describe the whole image with a single vector (CNN). The key point level representation is too “micro” while the image level representation is too “macro”. The failed examples in Figure 8.4 indicates that some “object” level image representations are needed to further boost the retrieval performance.

8.2. ukbench

We do the similar experiments on ukbench dataset to further compare the powered asymmetric loss and the logistic loss. In ukbench dataset, as mentioned above, there are 10200 images which are in 2550 groups, and all images can be query. We empirically select 510 images from 510 different groups as queries to construct the training data, and the others are used as testing data. Just as before, the visual word (20K dictionary), CNN (fc6, fc7, fc8) are used as the image representations. The cross validation results are shown in Table 8.3, and 8.4, in which the hyperparameter $\alpha$ of the powered asymmetric loss is set to $1e12$.

In ukbench dataset, each query has 3 relevant images and all other images are irrelevant, which is similar as Holidays dataset where on average each query has 1.982 relevant images. Table 8.3 also gives a similar results. Both ”Train accuracy, negative objects” and ”Test accuracy, negative objects” of the powered asymmetric loss is higher than the logistic loss. This further proves our anticipation: the powered asymmetric loss firstly ensures all negative objects are classified as negative, and under this condition finds the best decision boundary. One may doubt why the standard deviation of ”Train accuracy, positive objects” and ”Test accuracy, positive objects” of the powered asymmetric loss are much larger than the logistic loss in Table 8.3. This probably because with different training data, after finding the decision boundary to make all negative objects negative, the number of remaining objects changes drastically. But this usually will not have a big influence on the retrieval performance, because the retrieval performance depends on the relative distances of the objects to the decision boundary instead of the absolute position of the boundary. For example, if the linear classifier is used, the decision boundary is a line, and in this case the retrieval performance is only affected by the slope and has no relation to the intercept.
Figure 8.3: Some retrieval results of Holidays dataset where the powered asymmetric loss outperforms the logistic loss. The leftmost column contains the query, and other columns contain ground truth relevant images, and images being retrieved. The query image and the retrieved images are separate by the vertical blue line. The first row shows the ground truth relevant images. The second and third rows show the retrieved images by using the logistic loss and the powered asymmetric loss, respectively. The images in the green boxes are relevant, while the red boxes indicate irrelevant. All images are scaled to fit the figure.
Figure 8.4: Some retrieval results of Holidays dataset where both the powered asymmetric loss and the logistic loss fail. This figure has a similar structure as Figure 8.3.
Table 8.4 shows that the powered asymmetric loss has a notable improvements compared to the logistic loss. Especially, for the “Test mAP”, the improvement is about 3.5%, which is better than the corresponding improvements of Holidays dataset (about 2%). This probably because the amount of relevant images for a query for ukbench is 3, and for Holidays is 1.982.

Figure 8.5 shows the influence of parameter $a$ on ukbench dataset. Because the average number of relevant documents for a query of ukbench is similar as Holidays datasets, sub-figures of Figure 8.5 have similar manners to Figure 8.2. When parameter $a$ is very small (smaller than $1 + 1e^{-2}$), the logistic part plays an important role in the powered asymmetric loss. When $a$ becomes larger, it is more like a “ranking loss”, which means the top positions are focused on, and the mAP also becomes better. Moreover, when parameter $a$ is larger than $1 + 1e^{-2}$, with the increasing of $a$, the mAP also gradually becomes better, but at the same time the training also becomes harder to go to the local optimal. At last, some visual examples where the powered asymmetric loss is better than the logistic loss are presented in Appendix (Figure A.7).

8.3. Oxford5k

As mentioned in Section 7.1, in Oxford5k dataset, each query has 51.64 relevant images, and there are 55 queries in total. Compared with Holidays and ukbench datasets, Oxford5k has much less queries, and each queries has much more relevant images. Based on these properties of Oxford5k, we have the following assumptions:

- Because the training/testing data are split based on queries, and Oxford5k has less queries, the
Table 8.5: Accuracies of cross validation on Oxford5k dataset. The bold rows need to be focused on.

<table>
<thead>
<tr>
<th>Loss</th>
<th>exp+log</th>
<th>logistic</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Train</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All objects</td>
<td>99.3596%±0.1472%</td>
<td>99.6277%±0.1157%</td>
</tr>
<tr>
<td>Positive objects</td>
<td>27.7546%±2.0169%</td>
<td>61.6074%±4.3684%</td>
</tr>
<tr>
<td>Negative objects</td>
<td>99.9984%±0.0022%</td>
<td>99.9592%±0.0225%</td>
</tr>
<tr>
<td><strong>Test</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All objects</td>
<td>98.9738%±0.1936%</td>
<td>99.0163%±0.3880%</td>
</tr>
<tr>
<td>Positive objects</td>
<td>18.0202%±1.6757%</td>
<td>37.7564%±1.9483%</td>
</tr>
<tr>
<td>Negative objects</td>
<td>99.9982%±0.0022%</td>
<td>99.7852%±0.2255%</td>
</tr>
</tbody>
</table>

Table 8.6: AUCs and mAPs of cross validation on Oxford5k dataset. The bold rows need to be focused on.

<table>
<thead>
<tr>
<th>Loss</th>
<th>AUC</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AUC</strong></td>
<td>Training data</td>
<td>55.1536%±3.1620%</td>
</tr>
<tr>
<td></td>
<td>Testing data</td>
<td>8.88%±0.48%</td>
</tr>
<tr>
<td><strong>mAP</strong></td>
<td>Training data</td>
<td>67.62%±2.36%</td>
</tr>
<tr>
<td></td>
<td>Testing data</td>
<td>68.92%±1.82%</td>
</tr>
</tbody>
</table>

Experiments results may be more noisy.

- Because the powered asymmetric loss focuses on the top positions, the improvements on Oxford5k dataset may be not so dramatic compared to other datasets.

We empirically select 25 queries to construct the training data, and the remaining queries to construct the testing data. And still the visual word (20K dictionary), CNN (fc6, fc7, fc8) are used as image representations. The experiments results are shown in Table 8.5, and Table 8.6, in which the parameter $a$ is set to $1e8$ for the powered asymmetric loss. It is clear that some common observations we have for other datasets still hold:

- The “Train accuracy, negative objects” and “Test accuracy, negative objects” of the powered asymmetric loss are larger than logistic loss in Table 8.5.
- The mAP of testing data of powered asymmetric loss is larger than the logistic loss in Table 8.6.

There are also some differences:

- For Oxford5k dataset, the AUC for testing data of the powered asymmetric loss is smaller than the logistic loss, which is different from the above two datasets. However the mean of AUCs of two losses are very close (8.88% and 8.19%), and the standard deviation of the logistic loss (1.51%) is much larger than the powered asymmetric loss (0.48%). So two AUCs are still on the same level.
- The improvements of the powered asymmetric loss to the logistic loss is about 1.5%, which is smaller than the Holidays dataset (about 2%) and the ukbench dataset (about 3.5%). This proves our assumption that when a query has many relevant images, the improvements become less obvious.

Figure 8.6 shows further experiment results of the influence of parameter $a$. Compared to Figure 8.2 and Figure 8.5, the manner of Figure 8.6 is a bit different. We can see the data points in the sub-figures of Figure 8.6 usually have higher standard deviations than ukbench or Holidays datasets, which means the data of Oxford5k is more noisy. But still we can see there is a turnover happens when $a$ is close to 1 or $1 + 1e^{-2}$. When $a < 1 + 1e^{-2}$, the loss is dominated by the logistic part, and when $a > 2$, the loss becomes a “ranking loss” and focuses on the top positions.

At last, some visual retrieval results are given in Appendix (Figure A.8).

8.4. Comparison of different asymmetric losses

From the above experimental results, the powered asymmetric loss performs better than the symmetric logistic loss. There are two key differences between the powered asymmetric loss and the symmetric logistic loss:
8.4. Comparison of different asymmetric losses

(a) Train accuracy.

(b) Train accuracy for positive objects.

(c) Train accuracy for negative objects.

(d) Test accuracy.

(e) Test accuracy for positive objects.

(f) Test accuracy for negative objects.

(g) AUC on testing data.

(h) Train mAP.

(i) Test mAP.

Figure 8.6: Influence of the hyperparameter $\alpha$ on OXfords5k.

1. The powered asymmetric loss applies the exponential loss for the negative objects.

2. A hyperparameter $\alpha$ is used as the trade-off between the two parts of the loss, and there are different ways of applying the hyperparameter $\alpha$:
   
   (a) Set weights to different objects by multiplication as in Equation 5.4, Equation 5.8, and Equation 5.11.
   
   (b) Set weights to different objects by raising a power as in Equation 5.7, Equation 5.10, and Equation 5.13.
   
   (c) The weighted losses can also be normalized to make them go through point $(0, 1)$, as in Equation 5.5, Equation 5.9, and Equation 5.12.

In order to further explore what makes the powered asymmetric loss better than the symmetric loss, and check the influence of the outliers, we give a brief comparison of different losses proposed in Section 5.2.1. The cross validation results for the Holidays dataset are shown in Table 8.7. In general, from Table 8.7, the powered asymmetric loss performs best, but the differences of the test mAP for different losses are very small. In order to check if the differences are significant, we use the paired Student’s $t$-test to compare the losses, and the results are given in Table 8.8. If we set the threshold of the p-value as 0.050, then with a larger p-value, the null hypothesis cannot be rejected, otherwise if the p-value is smaller than 0.050, we reject the null hypothesis. In this case, the null hypothesis is “the performance of one classifiers is not significantly larger/smaller than the other one”. Now from Table 8.8 we can see that the losses are divided into two groups:
Table 8.7: Comparison of different losses on Holidays dataset.

<table>
<thead>
<tr>
<th>Loss</th>
<th>mAP</th>
<th>α</th>
</tr>
</thead>
<tbody>
<tr>
<td>symmetric (logistic)</td>
<td>76.61%±0.63%</td>
<td>76.69%±0.88%</td>
</tr>
<tr>
<td>weighted-symmetric</td>
<td>78.56%±2.55%</td>
<td>77.27%±1.21%</td>
</tr>
<tr>
<td>normalized-weighted-symmetric</td>
<td>81.08%±4.42%</td>
<td>76.90%±0.90%</td>
</tr>
<tr>
<td>powered-symmetric</td>
<td>81.19%±2.97%</td>
<td>78.12%±1.08%</td>
</tr>
<tr>
<td>weighted-asymmetric</td>
<td>78.55%±2.62%</td>
<td>77.22%±1.17%</td>
</tr>
<tr>
<td>normalized-weighted-asymmetric</td>
<td>81.08%±4.42%</td>
<td>76.90%±0.90%</td>
</tr>
<tr>
<td>powered-asymmetric</td>
<td>79.23%±2.03%</td>
<td>78.77%±1.70%</td>
</tr>
<tr>
<td>weighted-nonconvex</td>
<td>80.90%±3.38%</td>
<td>77.32%±1.06%</td>
</tr>
<tr>
<td>normalized-weighted-nonconvex</td>
<td>81.08%±4.42%</td>
<td>76.90%±0.90%</td>
</tr>
<tr>
<td>powered-nonconvex</td>
<td>81.19%±2.96%</td>
<td>78.12%±1.03%</td>
</tr>
</tbody>
</table>

Table 8.8: The paired Student’s t-test to compare different classifiers using test mAP. The cells with p-value smaller than 0.050 have gray backgrounds.

<table>
<thead>
<tr>
<th>One-tailed p-value</th>
<th>weighted-symmetric</th>
<th>normalized-weighted-symmetric</th>
<th>powered-symmetric</th>
<th>weighted-asymmetric</th>
<th>normalized-weighted-asymmetric</th>
<th>powered-asymmetric</th>
<th>weighted-nonconvex</th>
<th>normalized-weighted-nonconvex</th>
<th>powered-nonconvex</th>
</tr>
</thead>
<tbody>
<tr>
<td>logistic</td>
<td>0.161</td>
<td>0.155</td>
<td>0.034</td>
<td>0.167</td>
<td>0.155</td>
<td>0.021</td>
<td>0.093</td>
<td>0.155</td>
<td>0.034</td>
</tr>
<tr>
<td>weighted-symmetric</td>
<td>0.290</td>
<td>0.027</td>
<td>0.152</td>
<td>0.155</td>
<td>0.006</td>
<td>0.447</td>
<td>0.290</td>
<td>0.006</td>
<td>0.141</td>
</tr>
<tr>
<td>normalized-weighted-symmetric</td>
<td>0.029</td>
<td>0.306</td>
<td>0.160</td>
<td>0.026</td>
<td>0.228</td>
<td>0.157</td>
<td>0.027</td>
<td>0.141</td>
<td></td>
</tr>
<tr>
<td>powered-symmetric</td>
<td></td>
<td>0.017</td>
<td>0.029</td>
<td>0.072</td>
<td>0.011</td>
<td>0.029</td>
<td>0.490</td>
<td></td>
<td></td>
</tr>
<tr>
<td>weighted-asymmetric</td>
<td></td>
<td>0.306</td>
<td>0.004</td>
<td>0.401</td>
<td>0.305</td>
<td>0.008</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>normalized-weighted-asymmetric</td>
<td></td>
<td>0.026</td>
<td>0.228</td>
<td>0.354</td>
<td>0.027</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>powered-asymmetric</td>
<td></td>
<td></td>
<td>0.003</td>
<td>0.026</td>
<td>0.070</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>weighted-nonconvex</td>
<td></td>
<td></td>
<td></td>
<td>0.228</td>
<td>0.018</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>normalized-weighted-nonconvex</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.027</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. The logistic loss, the weighted symmetric loss, the normalized weighted symmetric loss, the weighted asymmetric loss, the normalized weighted asymmetric loss, and the normalized weighted nonconvex loss.

2. The powered symmetric loss, the powered asymmetric loss, and the powered nonconvex loss.

There are no significant differences between the losses within the same group, and the losses in different groups are statistically different. Now by checking the losses in different groups, we can conclude that the key component in an asymmetric loss is the way of setting the trade-off between the two parts of the loss. In particular, in order to significantly improve the retrieval performance, an appropriate power should be raised to the loss for the negative objects.

Besides, from Table 8.7, one may observe that the performances of the normalized weighted symmetric, the normalized weighted asymmetric loss, and the normalized weighted nonconvex loss are the same. When training a classifier, its weights usually is initialized in the scale of $1e^{-2}$, and the training data is also standardized (transformed to zero mean and standard deviation), so the $s(x)$ usually is not very large, i.e., usually its absolute value is smaller than 10. Here we give a brief mathematical proof to show the three losses are almost identical.

Firstly, it is obvious that the losses for $y = 1$ are identical (they are all logistic losses when $y = 1$). Secondly, when $y = -1$, the three losses are:
8.4. Comparison of different asymmetric losses

\[
\text{Loss}_{\text{normalized-weighted-symmetric}}(y = -1) = \frac{a}{\log 2} \log \left( 1 + \exp \left( s(x) + c \right) \right)
\]

\[
= \frac{a}{\log 2} \log \left( 1 + \exp \left( s(x) + \log \left( \exp \left( \frac{\log 2}{a} \right) - 1 \right) \right) \right)
\]

\[
(8.1)
\]

\[
\text{Loss}_{\text{normalized-weighted-asymmetric}}(y = -1) = a \exp (s(x) + c)
\]

\[
= a \exp (s(x) + - \log a)
\]

\[
= a \exp (s(x)) \exp (- \log a)
\]

\[
= \exp (s(x))
\]

\[
(8.2)
\]

\[
\text{Loss}_{\text{normalized-weighted-nonconvex}}(y = -1) = \frac{2a}{1 + \exp (-s(x) - c)}
\]

\[
= \frac{2a}{1 + \exp (-s(x) + \log(2a - 1))}
\]

\[
= \frac{2a}{1 + \exp(-s(x))(2a - 1)}
\]

\[
(8.3)
\]

For simplicity, we let \( m = \exp(s(x)) \), then:

\[
\text{Loss}_{\text{normalized-weighted-symmetric}}(y = -1) = \frac{a}{\log 2} \log \left( 1 + m \left( \exp \left( \frac{\log 2}{a} \right) - 1 \right) \right)
\]

\[
(8.4)
\]

\[
\text{Loss}_{\text{normalized-weighted-asymmetric}}(y = -1) = m
\]

\[
(8.5)
\]

\[
\text{Loss}_{\text{normalized-weighted-nonconvex}}(y = -1) = \frac{m}{1 + \frac{m - 1}{2a}}
\]

\[
(8.6)
\]

For the normalized weighted nonconvex loss, by the Taylor expansion, we have:

\[
\text{Loss}_{\text{normalized-weighted-nonconvex}}(y = -1) = \frac{m}{1 + \frac{m - 1}{2a}}
\]

\[
= m \left( 1 - \frac{m - 1}{2a} + \left( \frac{m - 1}{2a} \right)^2 - \left( \frac{m - 1}{2a} \right)^3 + ... \right)
\]

\[
(8.7)
\]

Because \( s(x) \) usually is smaller than 10, \( m = \exp(s(x)) \) is also not so large compared to \( a \), which from Table 8.7 is \( 1e6 \) for the normalized weighted nonconvex loss. So we have \( \frac{m - 1}{2a} = 0 \). Then Equation 8.7 is approximate to:

\[
\text{Loss}_{\text{normalized-weighted-nonconvex}}(y = -1) = m \left( 1 - \frac{m - 1}{2a} + \left( \frac{m - 1}{2a} \right)^2 - \left( \frac{m - 1}{2a} \right)^3 + ... \right)
\]

\[
\approx m
\]

\[
(8.8)
\]

For the normalized weighted symmetric loss, we still use the Taylor expansion to expand it:
Experiments of the asymmetric losses

\[ \text{Loss}_{\text{normalized-weighted-symmetric}}(y = -1) = \frac{a}{\log 2} \log \left( 1 + m \left( \exp \left( \frac{\log 2}{a} \right) - 1 \right) \right) \]

\[ = \frac{a}{\log 2} \left( m \left( \exp \left( \frac{\log 2}{a} \right) - 1 \right) - \left( m \left( \exp \left( \frac{\log 2}{a} \right) - 1 \right) \right)^2 \right. \]

\[ \left. + \left( m \left( \exp \left( \frac{\log 2}{a} \right) - 1 \right) \right)^3 + ... \right) \]

(8.9)

The \( \exp \left( \frac{\log 2}{a} \right) \) can also be expanded by the Taylor expansion:

\[ \exp \left( \frac{\log 2}{a} \right) = 1 + \frac{\log 2}{a} + \frac{\left( \frac{\log 2}{a} \right)^2}{2!} + \frac{\left( \frac{\log 2}{a} \right)^3}{3!} + ... \]

\( \approx 1 + \frac{\log 2}{a} \)  

(8.10)

Then Equation 8.9 is:

\[ \text{Loss}_{\text{normalized-weighted-symmetric}}(y = -1) \approx \frac{a}{\log 2} \left( m \left( \log 2 \right) - m \left( \frac{\log 2}{2} \right)^2 + m \left( \frac{\log 2}{a} \right)^3 + ... \right) \]

\[ \approx m - \frac{m^2 \log 2}{2} + \frac{m^3 \left( \frac{\log 2}{a} \right)^2}{3} + ... \]

\( \approx m \)  

(8.11)

Now from Equation 8.5, Equation 8.8, and Equation 8.11, we can see when \( s(x) \) is small, the normalized weighted symmetric loss, the normalized weighted asymmetric loss, and the normalized weighted nonconvex loss are almost identical. Figure 8.7a and 8.7b give intuitive visualizations of the three losses when \( y = -1 \), where the only difference between the two figures is the limitation of the horizontal axis. From these two figures, it is clear that when \( s(x) \) is small, the three losses are almost identical.

Besides, in Table 8.7, the performances for the powered symmetric loss and the powered nonconvex loss are also very close, which is shown visually in Figure 8.7c. In Figure 8.7c, the powered symmetric and the powered nonconvex losses are very close.

At last, Figure 8.7c shows the losses in group 2, and the logistic loss, and Figure 8.7d shows the losses in group 1 except their normalized versions. Comparing Figure 8.7c and Figure 8.7d, we can generally conclude that a steeper pointwise loss is better for the retrieval problem.

8.5. Transfer learning

From the previous experiments, we can basically conclude that the weighted asymmetric loss is better than the symmetric logistic loss, and for all datasets, the supervised methods have good performances. However all experiments are done within one dataset, i.e., one dataset is split for training and testing,
8.5. Transfer learning

(a) The normalized weighted symmetric loss, the normalized weighted asymmetric loss, and the normalized weighted nonconvex loss, when $y = -1$.

(b) The normalized weighted symmetric loss, the normalized weighted asymmetric loss, and the normalized weighted nonconvex loss, when $y = -1$.

(c) The logistic loss, the powered symmetric loss, the powered asymmetric loss, and the powered nonconvex loss, when $y = -1$.

(d) The logistic loss, the weighted symmetric loss, the weighted asymmetric loss, and the weighted nonconvex loss, when $y = -1$.

Figure 8.7: Different pointwise ranking losses, where the hyperparameters as in Table 8.7 are used. Note that to make the visualizations clear, in different sub-figures of, we use different scales for the horizontal and vertical axes.

which makes the whole model hard to be compared with unsupervised methods. Currently the majority of the image retrieval methods are unsupervised, where an implicit assumption is held: in a real content based image retrieval system, the search engine usually has no prior knowledge of what kinds of query image the user will input. As a result, the unsupervised method is applied to avoid catching dataset-specific characteristics by the supervised method. So the performances we have above in this section cannot be directly compared with unsupervised methods. In order to perform the comparison and answer the question 6 in Chapter 6, we do the transfer learning experiments in this section, i.e., the classifiers are trained on one dataset and applied on the others.

In detail, the settings of the experiments are: the SIFT based visual words and the CNN are used as the image representations, the traditional text based feature combination method is used, and the pointwise learning to rank with the powered asymmetric loss is used for ranking. The results are shown in Table 8.9. From Table 8.9, we can see that in general for the same testing dataset, with different training datasets, the mAPs are at the same level, which indicates the transfer learning works well. The reasons probably are:

- In the previous experiments, where a single dataset is split into training and testing parts, only a small proportion of queries are used to construct the training data. Consequently, the training data may not cover enough dataset-specific characteristics. In detail, for Holidays dataset, 50
Table 8.9: mAP of transfer learning. * represents that the training and testing datasets are the same. In detail, the value with * are taken from Table 8.2, Table 8.4, and Table 8.6, respectively.

<table>
<thead>
<tr>
<th></th>
<th>Testing</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Holidays</td>
<td>ukbench</td>
<td>Oxford5k</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Training</td>
<td>Holidays</td>
<td></td>
<td>78.77%±1.70% *</td>
<td></td>
<td>93.03%</td>
<td></td>
<td>67.06%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ukbench</td>
<td></td>
<td>80.51%</td>
<td></td>
<td>93.49%±0.48% *</td>
<td></td>
<td>66.46%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Oxford5k</td>
<td></td>
<td>77.11%</td>
<td></td>
<td>91.27%</td>
<td></td>
<td>68.92%±1.82% *</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 8.10: The optimal hyperparameter $a$ of transfer learning. * represents that the training and testing datasets are the same.

<table>
<thead>
<tr>
<th></th>
<th>Testing</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Holidays</td>
<td>ukbench</td>
<td>Oxford5k</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Training</td>
<td>Holidays</td>
<td>1e12 *</td>
<td>1e12</td>
<td>1e20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ukbench</td>
<td>1e10 *</td>
<td>1e12</td>
<td>1e14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Oxford5k</td>
<td>1e2</td>
<td>1e2</td>
<td>1e8 *</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

out of 500 queries are used to construct the training data in Section 8.1. For ukbench dataset, 510 out of 10200 queries are used, and for Oxford5k dataset, 25 out of 55 queries are used to construct the training data.

- The feature combination method, i.e., the traditional feature combination method in text retrieval, is directly borrowed from the text retrieval, which is designed by some experts. It probably only represent some characteristics which are general enough for different datasets.

Besides, although only a small proportion of queries are used to construct the training data, when training and testing are done on the same dataset, the performance usually is better than the transfer learning, which means in Table 8.9 usually the cells with * have the best performances. The only exception is the Holidays dataset, where the classifier trained on ukbench has better performance. The reason is in Section 8.1, only 50 out of 500 queries are used to construct the training data. From Figure 7.5, when only 50 queries are used to construct the training data, the learning curve has not converged yet, which means that by increasing the size of the training data in Section 8.1, the mAP could become higher and finally surpass the transfer learning results. The comparison of the results in this report and the state-of-the-art results are put in Appendix (Table A.1).

Note that the mAPs in Table 8.9 are optimized, i.e., for the same training dataset the hyperparameters are optimized individually on different testing datasets, which is not so convincing because from a supervised learning perspective, the testing dataset should only be used once. As a result, we also gives the relationship of parameter $a$ and the mAP in Figure 8.8. From Figure 8.8, we can see in most of the cases, we can get a good mAP with a large range of the hyperparameter $a$ to be selected. For instance, Figure 8.8c indicates that when trained on the ukbench dataset, and tested on the Holidays dataset, the classifiers have almost the same performances with $a$ from $1e6$ to $1e14$. Consequently, Table 8.9 gives a fair comparison of the supervised and unsupervised methods.

Finally, Table 8.10 gives all optimal selections of the hyperparameter $a$, from which we can see in general $a$ should be large enough. The exceptions happen when the training dataset is Oxford5k, where the reason is in Oxford5k dataset, the average number of relevant images for a query is 51.46 while the average number of relevant images for Holidays and ukbench are 1.982 and 3, respectively. As analyzed above, when increasing the parameter $a$, the asymmetric loss gives larger penalties to the false positive objects, which results in the classifier tends to classify as many objects to negative as possible. So the asymmetric loss usually performs better when the number of relevant documents for a query is not very large. As a result, when using the asymmetric loss in the transfer learning, the hyperparameter $a$ should be chosen based on the average number of relevant documents a query has.

8.6. Summary
In this chapter, we firstly compare the powered asymmetric loss with the logistic loss on three different datasets. The experimental results answer the question 3 in Chapter 6: for the retrieval problem, the powered asymmetric loss indeed has an improvement compare to the logistic loss. And the powered
asymmetric loss focuses on the top positions, so the powered asymmetric loss is most suitable for the queries with not so many relevant images. In general, with the increasing of parameter $a$ in the powered asymmetric loss, the performance will also be improved. But on the other side, a larger $a$ makes the training become more difficult and need more careful pre-processing of the data to prevent the numerical problems.

Besides, different asymmetric losses are compared. The experimental results reveal that the key to design a good asymmetric loss for the retrieval problem is the way of setting the trade-off between two parts of the losses. Specifically, a power larger than 1 should be raised to the loss for the negative objects.

At last, we evaluate if the model trained on one dataset can be transferred to other datasets. The experimental results show that with appropriate settings, especially the appropriate feature combination method, the model trained on one dataset can be transferred to other datasets while keeping a comparable performance. As a result, the supervised ranking methods used in this chapter can be directly compared with other unsupervised methods.
Experiments of an end-to-end trainable retrieval system

In the previous two experiments chapters, all supervised methods used the traditional feature combination methods in text retrieval. The traditional feature combination methods in text retrieval are fixed and hand designed by experts with domain specific knowledge. We can expect that this is not optimal and the representation power of the image representations is reduced. Instead, a learnable feature combination method, distance metric learning, is introduced as an alternative. Since the distance metric learning gives a vector with multiple items as output instead of a single value, the following ranking method should also be supervised, so the learning to rank approach is chosen. Furthermore, in order to construct an end-to-end fully learnable system, the CNN based image representation is applied. Overall, the experiments in this chapter test the end-to-end trainable retrieval system. All the experiments in this chapter are build on a widely used open-source deep learning framework: Caffe [41].

9.1. ConvNet structure

Our CNN system is based on the pre-trained AlexNet as shown in Section 3.3.6. In the previous experiments in Section 7.3, we use different layers’ outputs as image representations. The results indicate that the output of fc6 has the highest performance among all layers. Therefore in this section the output of layer fc6 is used as input for the distance metric learning.

For the distance metric learning, two kinds of distance metric are used, which are presented in Equation 4.11, and Equation 4.12. The two network structures are shown in Figure 9.1 which only contains the simplified versions. And the full structures are put in the Appendix (Figure A.9 and Figure A.10).

9.1.1. ConvNet for Equation 4.11

The ConvNet in Figure 9.1a uses Equation 4.11 as the feature combination method. In detail, layer pair_data accepts a pair of images as input, which is a query-image pair in our case with a label 1 (relevant) or -1 (irrelevant) as shown in Equation 5.1. The pair data is then sliced by the layer slice_data into layer data and data_p. Then each slice is propagated forward by the “pre-trained net”, which is a part of the AlexNet. The output of layers fc6 and fc6_p are two 4906 dimension vectors, which are \( v_q \) and \( v_d \) in Equation 4.11. They are concatenated by layer concat1, which gives an output vector of 8192 (=4096+4096) dimension:

\[
\mathbf{v}_{\text{concat1}} = \left( v_{q,1}, v_{q,2}, ..., v_{q,g}, v_{d,1}, v_{d,2}, ..., v_{d,r} \right)
\]

(9.1)

The \( \mathbf{v}_{\text{concat1}} \) is fed to a fully-connected layer fc9, whose weight matrix is a vertical stack of two diagonal matrices:

\[
W_{\text{fc9}} = \begin{bmatrix}
I_{4096} \\
-I_{4096}
\end{bmatrix}
\]

(9.2)
Experiments of an end-to-end trainable retrieval system

(a) Net structure using the feature combination method in Equation 4.11.

(b) Net structure using the feature combination method in Equation 4.12.

Figure 9.1: Structures of the end-to-end systems, where the "pre-trained net" means the layer conv1 to conv5 of the AlexNet (Figure 3.6).
where $W_{fc9}$ is the weight matrix of layer $fc9$, and $I_{4096}$ means an identity matrix. So $W_{fc9}$ is a $8192 \times 4096$ matrix:

$$W_{fc9} = \begin{bmatrix}
1 & 0 & \ldots & 0 \\
0 & 1 & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & 1 \\
-1 & 0 & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & -1
\end{bmatrix}$$

During training, layer $fc9$ is fixed, and the output of $fc9$ is a 4096 dimension vector

$$v_{fc9} = \text{concat} \cdot W_{fc9} = (v_{q1} - v_{d1}, v_{q2} - v_{d2}, \ldots, (v_{q|t|} - v_{d|t|}))$$

Then the $v_{fc9}$ is fed into layer $power1$, which is an element wise power layer with a power strength 2. So

$$v_{power1} = \left((v_{q1} - v_{d1})^2, (v_{q2} - v_{d2})^2, \ldots, (v_{q|t|} - v_{d|t|})^2\right)$$

The layers concat, $fc9$, and $power1$ together construct the distance metric shown in Equation 4.11. These three layers are fixed during training.

Finally, layer $fc10$ has a $4096 \times 2$ weight matrix, which is randomly initialized with zero mean and small standard deviation. During training, the parameters of layer $fc10$ are updated. The output of layer $fc10$ is a $1 \times 2$ vector, and the loss is calculated based on this vector and the label. The two losses shown in Figure 9.1 are softmax loss, which calculates the multinomial logistic loss of the softmax of its inputs. Since we use the binary pointwise learning to rank, the softmax loss is identical to the binary logistic loss which we have evaluated in the previous chapters. In later experiments, we also replace it with the powered asymmetric loss to further compare these two losses.

Now the whole system is end-to-end learnable, where all weights can be updated during training.

9.1.2. ConvNet for Equation 4.12

Figure 9.1b shows the ConvNet which uses the distance metric described in Equation 4.12. It has a similar structure to ConvNet in Figure 9.1a. The layers from pair_data to concat of the two ConvNets are identical, which are introduced in the pervious subsection. As shown in Equation 4.1, the output of layer concat is a $8192 \times 2$ vector, which is the naive distance metric defined in Equation 4.12. Then the fully-connected layer $fc11$ with a $8192 \times 2$ weight matrix accepts the output of layer concat as input, and generates a $1 \times 2$ vector to calculate the loss. The layer $fc11$ is initialized randomly with zero mean, small standard deviation and its weights are updated during training.

9.2. General training settings

Limited by time, we only perform the experiments on Holidays dataset. Ideally, it should be best to also do the cross validation as shown in Table 8.1 and Table 8.2, but tuning a ConvNet takes very a long time even with a GPU-implementation. As a result, we directly adopt the settings in Section 7.4 and use the data splitting method shown in Figure 7.1, i.e., the top 300 queries are used to construct the training data, the following 100 queries are used as validation data, and the remaining 100 queries are for testing data. With this data splitting method, there are 745,500 (=500*1491) objects in total, in which 447,300 are training objects, and both validation and testing data contain 149,100 objects.

In all 745,500 objects, only 1491 are positive, i.e., only 1491 query-image pairs are relevant. So it is a highly imbalanced dataset. The batch gradient decent is used to update the weights with a batch size of 256. When using the highly imbalanced training data, it is possible that in some iterations that all objects in a batch are negative, which will harm the training. So we avoid this problem by:

- Oversampling the positive query-image pairs by adding some data argumentations to the source images, like the rotation/flip/translation.
- Randomly undersampling the negative data to ensure that the number of negative data is the same as the positive data.

After the sampling, there are 304,896 training objects, in which the number of positive and negative objects are 152,488.

During training, at first the training objects are shuffled, and then in each iteration of the batch gradient decent, 256 objects are used to update the weights. When every object is iterated once, all objects are shuffled again, and the previous steps are repeated. Furthermore, if the weights of "pre-trained net" are updated, their learning rates are set 10 times smaller than the randomly initialized layers, and only layer fc6 are tuned to avoid overfitting.

To best explore how to train a good end-to-end retrieval system, and answer the question 3, 5 in Chapter 6, we test the following options:

- Distance metric in Equation 4.11.
  1. Train both fc10 and fc6 with logistic loss.
  2. Fix layer fc10, and only tune layer fc6 with logistic loss.
  3. Fix the "pre-trained net", and only train the layer fc10 with logistic loss.
  4. Fix the “pre-trained net”, and only train the layer fc10 with exp+log loss.

- Distance metric in Equation 4.12.
  1. Train both fc11 and fc6 with logistic loss.
  2. Fix layer fc11, and only tune layer fc6 with logistic loss.
  3. Fix the “pre-trained net”, and only train the layer fc11 with logistic loss.
  4. Fix the “pre-trained net”, and only train the layer fc11 with exp+log loss.

### 9.3. Distance metric in Equation 4.11
#### 9.3.1. Train both fc10 and fc6
The results are shown in Figure 9.2 and the second row of Table 9.1. Figure 9.2a shows the training loss at each iteration on the batch of 256 objects. Figure 9.2b shows the classification accuracies on all training or validation objects. From Figure 9.2a, it is clear that the training loss decreases. But we have also observed some problems:

![Figure 9.2](image)
### 9.3. Distance metric in Equation 4.11

#### Table 9.1: Retrieval performances of different settings for the distance metric in Equation 4.11.

<table>
<thead>
<tr>
<th>mAP</th>
<th>Train</th>
<th>Validation</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train both fc10 and fc6</td>
<td>9.44%</td>
<td>4.89%</td>
<td>4.11%</td>
</tr>
<tr>
<td>Fix fc10, and only tune fc6</td>
<td>30.40%</td>
<td>3.95%</td>
<td>3.74%</td>
</tr>
<tr>
<td>Fix fc6, and only train fc10</td>
<td>10.65%</td>
<td>5.42%</td>
<td>5.27%</td>
</tr>
<tr>
<td>Fix fc6, and only train fc10 with a large regularization term</td>
<td>11.44%</td>
<td>7.35%</td>
<td>15.45%</td>
</tr>
<tr>
<td>Fix fc6, and only train fc10 with the powered asymmetric loss</td>
<td>23.21%</td>
<td>14.10%</td>
<td>21.17%</td>
</tr>
</tbody>
</table>

- From Figure 9.2b, the training is overfitting, where the training accuracy is close to 1, and the validation accuracy is only about 0.8. The reason probably is that layer fc6 is a fully-connected layer, which contains 16,777,216 (=4096*4096) parameters, and layer fc10 contains 8192 (=4096*2) parameters. The big gap between the number of parameters and the size of the training data indicates the training is very easy become overfit. And there are three potential solutions:
  1. Fix layer fc6 and only train layer fc10, or the other way around.
  2. Use a larger dataset.
  3. Referring to Section 3.2.3.2, another solution is to increase the regularization strength by changing the setting of hyperparameters.
    (a) Increase the regularization term.
    (b) Use more dropout layer.
    (c) Use additional pooling layer.

- When considering the retrieval evaluation criteria, from the second row of Table 9.1, all of the training/validation/testing mAP are much worse than the performance of using traditional text feature combination methods shown in Table 7.6. The training mAP is only 9.44%, which seems very small, but it is not far from the expectation in some senses. Since the batch gradient decent is used, in each training iteration, the weights are optimized based on the specific batch. So in Figure 9.2b, the training accuracy is always close but smaller than 1, and after stable, the training accuracy is about 99%. From Figure 2.4, for a query with a few relevant images, the AP is sensitive to the top positions. The 99% training accuracy means for 1490 objects, about 15 (14.9) objects are misclassified. When calculating the average precision for a query with 2 relevant images, if 15 irrelevant objects are ranked higher than the relevant objects, the AP is 9.01%. If 8 (=15/2) irrelevant objects are ranked higher than the relevant object, the AP is 15.57%. Both 9.01% and 15.57% are close to the mAP we obtained (9.44%). There are some potential solutions for this problem:
  1. Add more training data.
  2. Increase the batch size.
  3. Use a loss which focuses on the top positions, like the powered asymmetric loss.

In the following sections, we gradually evaluate each different solution, and give a conclusion why this end-to-end system does not have good performance.

#### 9.3.2. Fix fc10, and only tune fc6

As mentioned above, tuning both layer fc10 and layer fc6 needs to update much more parameters than the number of training objects. A straightforward solution is to only tune one of them to decrease the number of parameters to be learned. In this section we fix the layer fc10, and only tune layer fc6. In theory, it makes sense to estimate we cannot get a good performance using this setting because the weights of layer fc10 are initialized randomly. The experiments in this section intent to check to what degree the overfitting could be by only tuning layer fc6. The results are shown in Figure 9.3 and Table 9.1.

Comparing Figure 9.3b to Figure 9.2b, when only tuning layer fc6, the training accuracy is even closer to 1, and the validation accuracy is a bit worse. The retrieval performance also suggests the same result: the training mAP is bigger while the validation and testing mAP are smaller. So we can
9. Experiments of an end-to-end trainable retrieval system

Figure 9.3: Fix the layer fc10, and only tune the layer fc6.

conclude that since the layer fc6 has much more parameters than the size of the dataset, even only tuning layer fc6, the training is very easy to get stuck in a local optimum and be overfitting. As a result, with the current training data, we decide to fix the weights in layer fc6.

9.3.3. Fix fc6, and only train fc10
In the last section, we fix the randomly initialized layer fc10, and only tune layer fc6. The result demonstrates that only tuning layer fc6 still results in overfitting, which indicates that if the layer fc10 needs to be learned from the random initialization, the layer fc6 should be fixed to decrease the number of parameters. In this section, we fix the layer fc6, and only tune layer fc10. Because layer fc10 has much less parameters (8192) than layer fc6 (16777216), we estimate that the overfitting should be released a bit in this section. The results are shown in Figure 9.4 and Table 9.1.

From Figure 9.4b, the overfitting still exists, and when compared with the previous results, the degree of overfitting does not have a notable improvement, which means even if only layer fc10 is trained, the amount of the training data is still far from enough. As a result, other methods to prevent overfitting, e.g., increasing the regularization term, adding more Dropout layer, are tried later. When it comes to mAP, both validation and test mAPs have a little improvements compared to the previous results in this section. However, they are still much worse than the results in Table 7.6.

9.3.4. Fix fc6, and only train fc10 with a large regularization term
As shown in the last section, even only the layer fc10 is trained, the overfitting still exists, which indicates with the current setting of hyperparameters, the size of the training data is too small. In this section we solve the overfitting problem by changing the hyperparameters, and limited by the time, only a large regularization term is applied. In detail, we only tune the layer fc10 with layer fc6 fixed, and the global regularization strength is set 1e7 times larger than the settings in Section 9.3.3. The results are shown in Figure 9.5 and Table 9.1.

By comparing Figure 9.5a and Figure 9.4a, when using a much larger regularization term, the convergence becomes much faster, which probably can be explained as during learning, the objective function is dominated by the regularization term and the classifier just finds and stays at a closest local minimum with a small regularization penalty. By comparing Figure 9.5b and Figure 9.4b, it is clear that with a larger regularization strength, the overfitting is indeed reduced, but it is not removed completely. The reason probably is the overfitting of the model is not only influenced by the regularization, but also other factors, like the network structure (the number of the dropout layer, pooling layer), the batch size, etc.. At last, when checking the fifth row of Table 9.1, by reducing the overfitting, the overall retrieval performances are better. However, the training mAP also becomes better, which is counterintuitive. Now we can only guess this is caused by the learning stays at a better local minimum, and it suggests
9.3. Distance metric in Equation 4.11

9.3.5. Fix fc6, and only train fc10 with the exp+log loss

As demonstrated in Section 9.3, with distance metric learning and pointwise learning to rank, it is possible to get relatively high accuracy but the retrieval mAP is still very low. One main reason is that the retrieval performance is highly influenced by the false positive objects. As a result, to further improve the retrieval performance, we try the exp+log loss shown in Equation 5.10. To make the training easier, we select the parameter $a = 1e5$. Besides the difference in the loss function, the other settings in this section are the same as Section 9.3.3. The results are shown in Figure 9.6 and Table 9.1.

By comparing Figure 9.6b with Figure 9.4b, the degree of overfitting is very close where the difference between the training accuracy and the testing accuracy is less that 20% after the curves are stable. When it comes to the retrieval performance, the applying of the exp+log loss has about 16% improvement in testing performance compared with the logistic loss, which further demonstrates that...
Table 9.2: Retrieval performances of different settings for the distance metric in Equation 4.12.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Train</th>
<th>Validation</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train both fc11 and fc6</td>
<td>0.21%</td>
<td>0.50%</td>
<td>0.53%</td>
</tr>
<tr>
<td>Fix fc11, and only tune fc6</td>
<td>0.18%</td>
<td>0.67%</td>
<td>0.58%</td>
</tr>
<tr>
<td>Fix fc6, and only train fc11</td>
<td>0.14%</td>
<td>0.69%</td>
<td>0.46%</td>
</tr>
<tr>
<td>Fix fc6, and only train fc10 with the powered asymmetric loss</td>
<td>5.25%</td>
<td>5.60%</td>
<td>5.68%</td>
</tr>
</tbody>
</table>

the exp+log loss indeed improves the retrieval performance by avoiding the false positive objects. However, the 21.17% testing mAP is still much worse than the methods in Section 7.4 (Table 7.6) where the whole system is not end-to-end. The bad retrieval performance of the systems in this section may suggest that more training data is needed.

![Figure 9.6: Fix the layer fc6, and only train the layer fc10 with the powered asymmetric loss.](image)

9.4. **Distance metric in Equation 4.12**

The structure of the ConvNet with the distance metric in Equation 4.12 is shown in Figure 9.1b. Inspired by the previous section, in this section, we try the following settings:

- Train both fc6 and fc11.
- Fix fc11, and only tune fc6.
- Fix fc6, and only train fc11.
- Fix fc6, and only train fc11 with the exp+log loss.

Here we only give the retrieval performances in Table 9.2. All the retrieval performances in Table 9.2 are very bad compared to the distance metric in Equation 4.11. This indicates that the simple concatenation of two image representation is not a good feature combination method. At last, although the overall performances are very bad, we can still observe that the exp+log loss is much better than the naïve logistic loss.

9.5. **Summary**

In this chapter, some end-to-end trainable retrieval system, i.e., some ConvNets, are applied, and the experimental results are given on the Holidays dataset. The experimental results show that the Holidays dataset is too small to train big ConvNets. Therefore the question 5 in Chapter 6 is partially answered:
to train an end-to-end retrieval system consists of large neural networks, the training data should also be large enough. So in future, other larger datasets, like the ukbench, can be evaluated. Besides, the experiments in this chapter further prove our previous conclusion: the powered asymmetric loss is better than the symmetric logistic loss for the retrieval problem. At last, for the selection of the distance metric, the classical distance metric described in Equation 4.11 is much better than using the naïve vector concatenation as the distance metric shown in Equation 4.12.
10

Summary and future work

10.1. Summary
The content based image retrieval is defined as retrieving the images which contain the same objects as the query image. There are many evaluation criteria for a retrieval problem, and we select the widely used mean Average Precision (mAP) to measure the performance of our retrieval systems. The mAP calculates the mean of the Average Precision (AP) for all queries. And the AP is a position based evaluation criterion which considers the order of all returned documents for a query by averaging the precisions at all positions. In short, the mAP on the one hand is order related, and on the other hand, is a single value criterion, which makes it very popular in evaluating the retrieval tasks.

In this report, we gave a general framework for a content based image retrieval system with three main parts: image representation, feature combination, and ranking. Then we introduced the classical bag-of-visual-words based method, and illustrated how to embed it into our framework in Chapter 2.

We proposed some alternatives for each part in the framework. The classical SIFT based visual words ignores spatial information, and the global characteristics of the input image. This inspired us to use the CNN as the image representation, which proves its generalization power in many tasks. For the feature combination method, two candidates were proposed: the traditional feature combination method in text retrieval, and the distance metric learning. At last, the fact that the classical bag-of-visual-words based method is unsupervised makes the entire system manually designed and tuned. So we introduced the supervised ranking algorithm, i.e., the learning to rank, as the ranking method.

There are three main approaches in the learning to rank: the pointwise approach, the pairwise approach, and the listwise approach. Limited by the time, only the pointwise approach was applied in the experiments, where we treated the ranking as a binary classification problem, and in such a case, almost all classification losses can be set as the objective for the ranking problem. Except for the classical classification losses, we also proposed some asymmetric losses which were specifically designed for the ranking problem. As mentioned in Section 2.3, the top positions of the retrieved list dominate the influence on the average precision, and as a result, the wrongly classified irrelevant documents (the false positive objects) will degrade the mAP greatly. The asymmetric losses were proposed in order to better fit this characteristic of the average precision, where the number of the false positive objects are reduced by giving them higher penalties.

When interpreting from a system perspective, the biggest shortcoming for the traditional bag-of-visual-words based method is that its every part is designed manually, i.e., the feature combination method should fit the specific image representation, and the ranking method should fit the form of the combined features. All parts are closely linked, coherent, influencing each other, and in a way, inseparable. This makes the extension of the system also has to fit the existing parts, which is not trivial. However, because we implemented the retrieval system in a modular way, it was very easy for us to extend the system, and to experiment with many different combinations of representations, combiners and rankers. Some combinations seem very interesting and promising, for example, the CNN based image representation + distance metric learning + learning to rank make the whole system end-to-end trainable.

At last some experiments were conducted by using different combinations of the methods proposed
in each part. Limited by the time, we did not try and compare all combinations, and instead, only some critical ones, in order to answer some questions we want to explore. Here we list the questions and the conclusions obtained from the experimental results:

• Q: Is the CNN based image representation better than the SIFT based visual words? If so (or if not), in what degree?
  A: In general, it depends. With different datasets, different settings of the image representations, the performances vary a lot. In detail, if the number of relevant images for a query is very large, and the images in the datasets contain many objects, the SIFT based visual words with a large dictionary size usually performs better. If the a query does not have many relevant images, or the the images in the datasets have simple structure like the ukbench dataset, the CNN based image representation is a better choice.
  But the CNN based image representation contains the color and global information of an input image. So the information caught by the CNN based image representation is different from the SIFT based visual words.

• Q: How to fuse multiple image representations? Is the retrieval performance improved by doing so?
  A: When applying our framework, a concatenation of the combined features can be used to fuse multiple image representations.
  By fusing the image representations which catch different information, the retrieval performance can be improved. For example, the CNN based image representation contains the color and global spatial information of an input image, which is different from the SIFT based visual words. As a result, the fusion of these two image representations has a notable improvement compared with using them individually. On the other hand, if two fused image representations are very similar, e.g., two descriptors extracted from different layers of the same ConvNet, there is no improvements.

• Q: When using the pointwise learning to rank, is the asymmetric loss better than the symmetric loss? If so, what makes the asymmetric loss better?
  A: Because the asymmetric loss attaches importance on preventing the false positive objects, which have big influence on the retrieval evaluation criterion mAP, it outperforms the symmetric loss. The experimental results from Section 8.1 to Section 8.3 prove that the powered-asymmetric loss is better than the logistic loss. More importantly, the superiority of the powered-asymmetric loss is observed in different settings, i.e., different datasets, different image representations, and different feature combination methods. As shown in Chapter 9, even when training an end-to-end retrieval system, the powered-asymmetric loss is still better than the logistic loss.
  Besides, there are different ways of setting the trade-off between the losses for the positive and negative objects, e.g., one can multiply a weight to the loss part for the negative objects, or raise a power to it, or just use different losses for different objects. In Section 8.4, different asymmetric losses are compared by performing the Student’s t-test. The experimental results reveal that the key in designing a good asymmetric loss for the ranking problem is the way of setting the trade-off parameter. Specifically, by raising a power to the loss for the negative objects, the retrieval performance can be improved significantly.

• Q: Are the supervised method and unsupervised method comparable?
  A: In Section 8.5, we conduct the experiments of transfer learning for the supervised methods. In detail, the classifiers are trained on one dataset, and tested on others. The results indicate that for the transfer learning, the mAPs still are still in a competitive level. As a result, the supervised method has enough generalization power, and can be directly compared with other unsupervised methods.
  Besides, when performing the transfer learning, one needs to set a hyperparameter to balance the two parts of the asymmetric loss empirically. The strategy is that the selection of the hyper-parameter depends on the the average number of relevant documents a query has in the training dataset. Specifically, if the average number of relevant documents a query in the training data
is small, the hyperparameter can be set very large, and otherwise it should not be so large to prevent the drop of the performance.

• Q: How to build an end-to-end trainable retrieval system? Can it improve the performance? If so (or if not), why?
A: In our framework, using the CNN based image representation + distance metric learning + learning to rank makes the whole system end-to-end trainable. We evaluated some end-to-end trainable systems on the Holidays dataset, where all systems employ the CNN as the image representation, the distance metric learning as the feature combination method, and the pointwise learning to rank as the ranking method. Essentially an entire end-to-end system is a big neural network and it is optimized by gradient decent.

All the experimental results are much worse than than the systems with a few individual sub-modules where the same image representations are applied. This answers the question and indicates that with current datasets, the end-to-end trainable system cannot give improvements in performance. We conclude that the reason is the retrieval datasets we use are not large enough, so the training is prone to stop at some bad local minima.

• Q: What advice would you give for building a content based image retrieval?
A: First of all, the image representation is the most crucial part of a content based image retrieval system. Some appropriate image representations are the fundamentals of building a good retrieval system, where the “appropriate” indicates that the different image representations should catch some useful but different information from an image.

Secondly, the traditional feature combination methods in text retrieval shows its good performance and generalization power. It is a good choice to use it. One can even add more methods in text retrieval into Table 4.1.

Thirdly, the supervised learning to rank shows its superiority to the unsupervised sorting. In the pointwise learning to rank, the powered asymmetric loss is significantly better than the logistic loss. Besides, the transfer learning results in the experiments demonstrate that the classifier trained on one dataset is general enough to process the data from other datasets.

At last, when using the supervised method, training and testing on a large dataset make the work become easier.

10.2. Future work
The content based image retrieval is a very general topic, and there are still many possibilities to improve the retrieval system. Here we give some suggestions based on our research.

• The SIFT based visual words method only focuses on the local key points and ignores the global spatial information, while on the other hand, the CNN image representation extracts a single vector from the whole image. They are like two ends of a thread, so they can be improved in opposite directions.

1. The SIFT based visual words can be improved by adding more spatial information, under the condition that the efficiency requirement can be met. For example, [42] proposes a geometry-preserving visual phrases which encodes more spatial information into the raw SIFT based visual words.

2. The CNN based representation is a global image representation, and it can be improved by focusing on the local patches of the image. A straightforward improvement is to manually extract some patches from the input image and calculate the CNN representations for all of them [43].

• In this paper, we introduce two feature combination methods. The first one is the traditional feature combination methods in text retrieval, which directly borrows some techniques from the text retrieval. The second one is the distance metric learning. And both of them can be improved.
1. The traditional feature combination method in text retrieval used in this paper is shown in Table 4.1, where we only use some simple similarity measurements, and many other methods in the text retrieval are still available to be introduced into the image retrieval. For example, as illustrated in Section 4.1.2, we do not apply the query independent model into the image retrieval which leaves it an interesting future topic. For example, [44] develops a visual version of the PageRank.

2. For the distance metric learning algorithms, we give two candidates. The first one is the commonly used distance metric shown in Equation 4.5, and considering the feature space size, we only use a part of the distance metrics as shown in Equation 4.11. The second one is a simple concatenation (Equation 4.12). There are some potential improvements:

(a) Since in Equation 4.8, the size of the distance metric vector space is quadratic in the size of the input feature, it is not trivial to learn such a big distance metric. Many works are devoted to efficiently learning distance metric and keeping a satisfying performance [45], which can be seen as an extension of the research in this paper.

(b) Except for the two distance metrics introduced in this paper, are there any other distance metrics in particular designed for image retrieval can be used? For example, [18] gives a truncated histogram based distance metric for content based image retrieval.

In this paper, we give a comprehensive research and comparison of the pointwise learning to rank ranking methods. Moreover, we also propose a list of new asymmetric losses, which are specifically designed for the ranking problem. The experimental results shows the superiority of the powered asymmetric loss compared to the symmetric classification loss, which indicates that the ranking problem has its own characteristics, and the common algorithms may not perfectly fit the ranking problem. As mentioned in Section 5.2.1, the biggest characteristic of the ranking problem is that the top positions dominate the retrieval performance. The powered asymmetric loss tries to solve this problem by giving the false positive objects higher penalties, while this does not solve the problem essentially, i.e., when the irrelevant documents are ranked lower than the relevant documents, it does not matter if they are classified as false positive or false negative. As a result, the pairwise learning to rank is a natural choice to solve this problem, which focuses on the relative relations between the documents. However the pairwise approaches also suffer from that the training space becomes quadratically larger than the pointwise approaches. Except for the pairwise approaches, even the listwise approaches can also be tried.

Besides, all of the three approaches mentioned above do not directly set the evaluation criterion (e.g. the mAP) as the objective function, instead, usually an approximation is used. The reason is that the commonly used evaluation criteria for the ranking problem are non-continuous. As a result, it is natural to propose a question: is it possible to directly optimize the discrete evaluation criteria? [46] introduces a pairwise ranking loss which can integrate with the discrete evaluation criteria, while it is still an open question.

At last, different learning to rank approaches can be compared in order to check which is the best for the image retrieval problem.

In this paper, we do not evaluate the efficiency of our system, because it is highly influenced by the implementation and the hardware. But it can be done in the future research.

As shown in Chapter 9, the end-to-end trainable system performs much worse than the systems with some fixed sub-systems. By trying different settings, we conclude the reason is that the current dataset is too small to train a big end-to-end system. As a result, gathering more data and constructing a larger dataset is a promising option. With a larger dataset, it makes sense to estimate the content based image retrieval will be further improved by applying an end-to-end trainable system.
Appendix

A.1. Learning curves for the CNN image representation

In Section 7.5, we train several binary classifiers based on the CNN (layer $fc7$, layer $fc8$) based image representations, and traditional feature combination methods in text retrieval. The learning curves are shown in Figure A.1 and Figure A.2, respectively.

![Learning curves for the CNN image representation](image)

(a) The logistic regression.  
(b) The naive bayes classifier.  
(c) The fisher classifier.  
(d) The perceptron classifier.  
(e) The polynomial classifier.  
(f) The scaled nearest mean classifier.

Figure A.1: Learning curves for the CNN ($fc7$) based image representation with pointwise learning to rank.

A.2. Learning curves for the visual word and CNN image representations

In Section 7.6, we train several binary classifiers based on the fused feature of the CNN and the visual words based image representations. Figure A.3, Figure A.4, and Figure A.5 show the learning curves of the fused feature of the visual words and the CNN ($fc6$), CNN ($fc7$), CNN ($fc8$) based image representations, respectively. Figure A.6 shows the results of the fused feature of the visual words and all three CNN representations.
(a) The logistic regression. (b) The naive bayes classifier. (c) The fisher classifier. (d) The perceptron classifier. (e) The polynomial classifier. (f) The scaled nearest mean classifier.

Figure A.2: Learning curves for the CNN (fc8) based image representation with pointwise learning to rank.

Table A.1: Performance comparison between the best transfer learning results and the state-of-the-art without re-ranking.

<table>
<thead>
<tr>
<th></th>
<th>mAP</th>
<th>Our work</th>
<th>State-of-the-art</th>
</tr>
</thead>
<tbody>
<tr>
<td>Holidays</td>
<td>80.5%</td>
<td>89.7% [47]</td>
<td></td>
</tr>
<tr>
<td>ukbench</td>
<td>93.0%</td>
<td>95.0% [47]</td>
<td></td>
</tr>
<tr>
<td>Oxford5k</td>
<td>67.0%</td>
<td>84.4% [47]</td>
<td></td>
</tr>
</tbody>
</table>

A.3. Visual retrieval examples of the powered asymmetric loss and the logistic loss

In Chapter 8, we compare the powered asymmetric loss and the logistic loss on three different datasets. We put some visual retrieval examples here. Figure A.7 shows some visual examples of the ukbench dataset where the powered asymmetric loss is better than the logistic loss. Figure A.8 shows some results of Oxford5k dataset where the powered asymmetric loss outperforms the logistic loss. Note that in Figure A.8 we only select the queries with less than 15 relevant images because of the limitation of the page size.

A.4. Comparison with the state-of-the-art

The comparison between the best transfer learning results in Table 8.9 and the state-of-the-art is given in Table A.1.

A.5. Full ConvNet structures

In Chapter 9, we build some end-to-end retrieval systems based on AlexNet. For simplicity, in Figure 9.1, we only provide the simplified version of ConvNets we build. Here we show the full structures of them in Figure A.9 and Figure A.10. The parts in the brown boxes are the “pre-trained net” in Figure 9.1.
Figure A.3: Learning curves for the visual words + CNN (fc6) based image representations.

Figure A.4: Learning curves for the visual words + CNN (fc7) based image representations.
(a) The logistic regression.
(b) The naïve bayes classifier.
(c) The fisher classifier.
(d) The polynomial classifier.
(e) The scaled nearest mean classifier.

Figure A.5: Learning curves for the visual words + CNN (fc8) based image representations.

Figure A.6: Learning curves for the visual words + CNN (fc6, fc7, fc8) based image representations.
Figure A.7: Some retrieval results of the ukbench dataset where the powered asymmetric loss outperforms the logistic loss. This Figure has a similar structure as Figure 8.3.
Figure A.8: Some retrieval results of the Oxford5k dataset.
Figure A.9: The full structure of the ConvNet in Figure 9.1a, in which the parts in brown box are the "pre-trained net" in Figure 9.1a.
Figure A.10: The full structure of the ConvNet in Figure 9.1b, in which the parts in brown box are the “pre-trained net” in Figure 9.1b.


