Circuits and Systems
Mekelweg 4,
2628 CD Delft
The Netherlands
https://cas.tudelft.nl/

CAS-2022-00

M.Sc. Thesis

Search by Image: Deep Learning Based Image Visual Feature Extraction

Yanan Hu B.Sc.

Abstract

In recent years, the expansion of the Internet has brought an explosion of visual information, including social media, medical photographs, and digital history. This massive amount of visual content generation and sharing presents new challenges, especially when searching for similar information in databases —— Content-Based Image Retrieval (CBIR). Feature extraction is the foundation of image retrieval, making research into obtaining concrete features and representations of image content a vital concern. In feature extraction module, We first pre-process the target image and input it into a CNN to obtain feature maps for different channels. These feature maps can be aggregated into compact and global uniform descriptors by pooling. Then these global descriptors are further dimensionalised and normalised by whitening methods to obtain image feature vectors that are easy to compute and compare. In this process, the accuracy of the retrieval depends on how accurately the final feature vectors represent the meaning expressed by the target image. Therefore various CNN network structures, pooling methods and whitening methods are proposed to get more concrete feature vectors. In this thesis, our study (1) fine tunes the pre-trained CNNs, (2) optimizes the application of second order attention information in feature map, (3) applies and compares popular feature enhancement methods in both aggregating and whitening, (4) explores how to combine all strengths, and (5) propose a new model *ResNet-SOI*, which achieves 53.4(M) and 59.2(M) mAP on the challenging benchmark *ROxford5k+1M* and *RParis6k+1M*, and outperforms the state-of-art methods. Our prototype with GUI is available on GitHub(https://github.com/yanan-huu/Image-Search-Engine-for-Historical-Research).

Search by Image: Deep Learning Based Image Visual Feature Extraction

Thesis

submitted in partial fulfillment of the
requirements for the degree of

Master of Science

in

Electrical Engineering

by

Yanan Hu B.Sc.
born in Hengshui, China

This work was performed in:

Circuits and Systems Group
Department of Microelectronics
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Delft University of Technology

Delft University of Technology
Department of
Microelectronics

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for acceptance a thesis entitled **"Search by Image: Deep Learning Based Image Visual Feature Extraction"** by **Yanan Hu B.Sc.** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: 22 August 2022

Chairman: _____
prof.dr.ir. Justin Dauwels

Advisor: _____
prof.dr.ir. Justin Dauwels

Committee Members: _____
dr. Nergis Tömen

_____

Abstract

In recent years, the expansion of the Internet has brought an explosion of visual information, including social media, medical photographs, and digital history. This massive amount of visual content generation and sharing presents new challenges, especially when searching for similar information in databases —— Content-Based Image Retrieval (CBIR). Feature extraction is the foundation of image retrieval, making research into obtaining concrete features and representations of image content a vital concern.

In feature extraction module, We first pre-process the target image and input it into a CNN to obtain feature maps for different channels. These feature maps can be aggregated into compact and global uniform descriptors by pooling. Then these global descriptors are further dimensionalised and normalised by whitening methods to obtain image feature vectors that are easy to compute and compare. In this process, the accuracy of the retrieval depends on how accurately the final feature vectors represent the meaning expressed by the target image. Therefore various CNN network structures, pooling methods and whitening methods are proposed to get more concrete feature vectors.

In this thesis, our study (1) fine tunes the pre-trained CNNs, (2) optimizes the application of second order attention information in feature map, (3) applies and compares popular feature enhancement methods in both aggregating and whitening, (4) explores how to combine all strengths, and (5) propose a new model *ResNet-SOI*, which achieves 53.4(M) and 59.2(M) mAP on the challenging benchmark *ROxford5k+1M* and *RParis6k+1M*, and outperforms the state-of-art methods. Our prototype with GUI is available on GitHub(https://github.com/yanan-huu/Image-Search-Engine-for-Historical-Research).

Acknowledgments

As you start reading the first chapter of this thesis, I would have closed my final chapter in my TU Delft career.

The deepest and sincerest gratitude goes to my supervisor, Dr.ir Justin Dauwels. His continuous and invaluable guidance helped me complete this thesis. His expertise in machine learning has steereed me through many challenging circumstances.

My teammates Yuanyuan Yao and Qi Zhang have also brought unique perspectives and insights to my research. I appreaciate their support and efforts to this program. I could not have accomplished this retrieval search engine without their contribution. Also many thanks to Cristian Meo and Yanbo Wang for their generous advice and guidance.

I would also say thanks to my intrepid friends for their cheer at or beyond TU Delft. They have given me great comfort and support when I was at a loss and frustrated.

This thesis is dedicated to my parents without whom I could never be in the Netherlands pursuing my dream. Their unflagging support and unwavering faith are the best gift I've ever had. Mom and Dad, I love you from my bottom of heart.

Finally, thank you to myself for going through the ups and downs along the way, but never give up. The two years of studying abroad were not easy, but the joy and happiness I gained far outweighed the hard work.

Unknown future beckons, and I will go ahead bravely.

Yanan Hu B.Sc.
Delft, The Netherlands
22 August 2022

Contents

List of Figures

List of Tables

1 Introduction

This chapter presents an overview of the deep learning-based image search engine. We first present the project's rationale and objectives, followed by a discussion of the overall pipeline. Further, the feature extraction module in the pipeline, for which this paper is responsible, is explained. The final section outlines the thesis structure.

1.1 Image Retrieval

1.1.1 Problem Statement

In recent years, the expansion of the Internet has brought an explosion of visual information, including social media, medical photographs, and digital history. This massive amount of visual content generation and sharing presents new challenges, especially when searching for similar information in databases. Image retrieval approaches can substantially assist researchers with this endeavour by swiftly searching through thousands of images in milliseconds and locating a match for the object. Therefore, this deep learning-based work attempts to focus on feature extraction and optimization and collaborate on an image search system.

A typical approach for image search relies on annotated text retrieval. This method requires a textual annotation of the image, followed by a match under the annotated textual information. For instance, when searching for Eiffel Tower images, the user would enter "Eiffel Tower" as a query. The system would search the database for photographs with this text tag and deliver them to the user. Such methods incorporate the subjective opinion of the annotator and frequently accompany a biased interpretation of the image, which influences the search results. Additionally, there are also linguistic restrictions for text retrieval. Users may overlook results annotated in other languages if they type keywords in a specific language. Similarly, viewers do not know what keywords to choose if one image only contains an obscure landmark. The semantic difference between textual and image representations is the underlying source of all these issues.

For large-scale image retrieval, it is evident that text-based search is insufficient and that image-based search is the optimal strategy. This insight prompted us to create a content-based image search engine, in which the user may search for relevant photos with a single image.

1.1.2 Objectives

Inspired by numerous use cases, we will propose and validate our image retrieval technique, which satisfies the following objectives:

1) The representation of image features is abstract and global, enabling accurate representation of image content feature vectors, resulting in high accuracy of similar results for image search.

2) The feature representation is efficient and adaptable enough to support many types of digital historical photographs and large-scale database feature extraction projects.

3) The search performance of the semantic features of image material is effective and can reply fast and precisely to user search query requests.

4) To establish an image search system for digital historical materials and construct an open-source retrieval platform with improved model algorithm and scheme.

1.1.3 The Whole Picture

To achieve the digital historical search engine, we combine deep learning methods and propose a prototype including feature extraction, approximate nearest-neighbour search and re-ranking modules. Figure 1.1 shows the structure of this prototype.

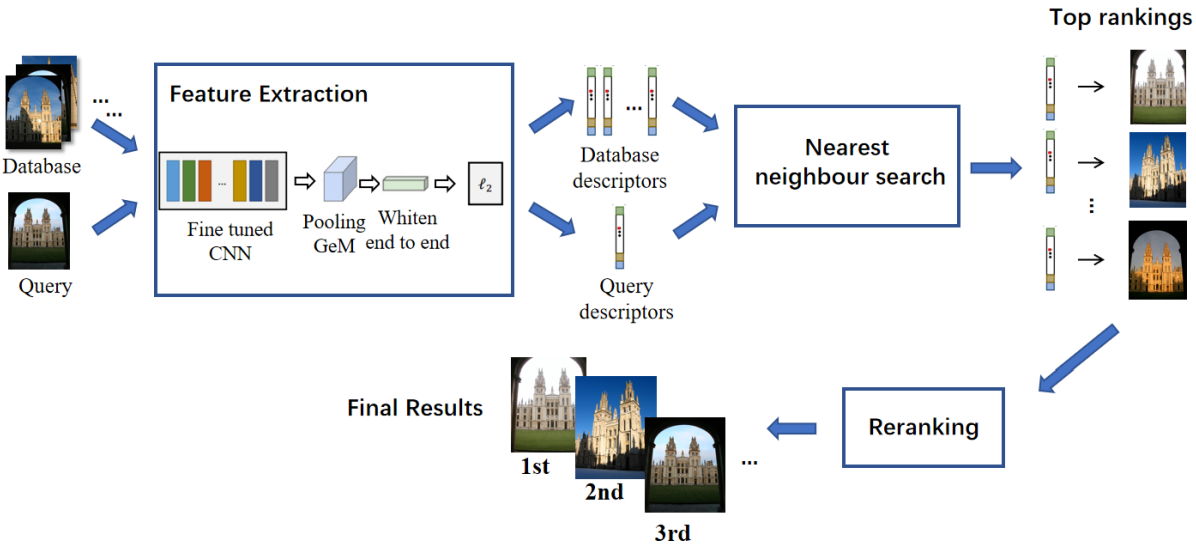


Figure 1.1: The whole pipeline.

In this prototype, we firstly extracted features from the query image online. Similarly, such features are also extracted from all images in the database offline. These features are extracted only once and then stored to process the image-based queries. The feature extractor comprises a CNN, followed by an aggregation layer, a whitening layer and an L2 normalization layer, which can be trained end-to-end with various loss functions. The extracted features are then forwarded to an approximate nearest-neighbour search module to produce the initial ranking results, refined by a re-ranking module, generating the final results.

This prototype was co-developed by Yuanyuan Yao, Qi Zhang and myself. I was responsible for the CNN-based feature extraction and enhancement module and jointly build the final retrieval system. The following section will provide an expanded introduction to this module.

1.2 Feature Extraction: From Image to descriptors

Imagine a user wants to search for an image, any relevant tag descriptions are missing, and this image is the only information provided. What can the retrieval system do now? How should this image be transformed into computer-intelligible data? These are tasks of the feature extraction module: efficiently converting a picture into a feature vector that can be readily processed and computed.

Good image features should possess the following three qualities.

Distinguishability
The retrieved image features must be easily distinguishable—the more considerable the variation in feature values between categories, the better. There should be significant disparities between the feature values of different categories.

Reliability:
Objects belonging to the same category should have comparable feature values. For instance, apples with varying degrees of ripeness typically have distinct hues. Green and red apples are both apples, but their colours are very different. Therefore using colour to identify apples is not a smart idea.

Independence
Individual characteristics should be independent of one another and unrelated to one another. Suppose two feature values represent the same attribute of an object. In that case, they will lead to data redundancy and a substantial increase in computing complexity.

1.3 Outline

Chapter 1 begins with an introduction to digital historical image retrieval systems, followed by the inspiration for the issue, the intended objectives, and the entire pipeline. The crux of this thesis, the objectives and implications of the image retrieval module, are then introduced, followed by a summary of the technique employed in this paper. Finally, the thesis's main structure is provided.

Chapter 2 examines the research on content-based image retrieval (CBIR) systems, focusing on CNN networks and CNN-based feature aggregation algorithms.

Chapter 3 described methodologies in depth. It begins with the design and application of CNN networks, followed by integrating related feature aggregation algorithms

and feature post-processing via whitening and dimensionality reduction. Finally, a combination of these strategies, the feature extraction pipeline proposed in this thesis, is given.

Chapters 4 demonstrate in detail, from an implementation standpoint, how the baseline and fine-tuned models are developed and trained. Also provided are experimental setups such as datasets and evaluation metrics. Finally, we present the results of the test experiments and the overall retrieval system.

Chapter 5 summarises the entire work's contributions and recommendations for further research.

2 Related Work

This chapter examines the research on image feature extraction, beginning with a discussion of content-based image retrieval systems (Section 2.1). Within the scope of this thesis, we focus on techniques employing deep learning and convolutional neural networks (CNNs). In Section 2.2, we provide a summary of off-the-shelf and fine-tuned CNN feature extraction models. And in Section 2.3, I discusses studies aimed at enhancing the feature quality from both the aggregation and embedding[6] viewpoints.

2.1 Content Based Image Retrieval(CBIR) Research

With the advancement of computer vision, pattern recognition, and machine learning, there is a requirement for efficient image management, organisation, and querying. How to search the continuously and rapidly expanding image library for the necessary images become a popular issue. Currently, there are three types of image retrieval: text-based image retrieval system (TBIR), content-based image retrieval system (CBIR), and semantic-based image retrieval system (SBIR). Figure 2.1 illustrates the various schemes of image retrieval and their fundamental mechanisms [1].

In TBIR system, retrieval is dependent on image labels derived from manual annotation. By entering keywords, the user gets matched images with related text tags in the database. The problem with this retrieval method is the incompatibility between human visual information comprehension and text labelling. The visual information is fixed and consistent for the same image, however, the linguistic description varies widely between individuals, which makes it impossible for manual annotations to adequately describe all the visual information in that image, resulting in retrieval errors. Moreover, manual tagged text is subject to language constraints, making international cross-language searching difficult. Furthermore, manual labelling of image files is nearly impossible when exploring large databases.

CBIR is a framework that can overcome these problems. The objective of content-based image retrieval is to locate similar images from a large-scale dataset against a query image. This retrieval approach employs the similarity of visual features of images to generate a similarity feature vector of database images, and then calculates the similarity distance between the query image and the database image to conduct an “image search by image” By specifying the aspects of picture content itself, image matching accuracy is enhanced. There is nearly endless potential for applications of CBIR, such as remote sensing, face recognition, e-commerce, and medicine.

Figure 2.2 depicts the overall framework of a CBIR system. CBIR systems can be separated into offline and online subsystems based on the stage of image information processing. The offline subsystem extracts feature from images in the database and stores them as descriptors, which are then encoded into indexes. The online subsystem

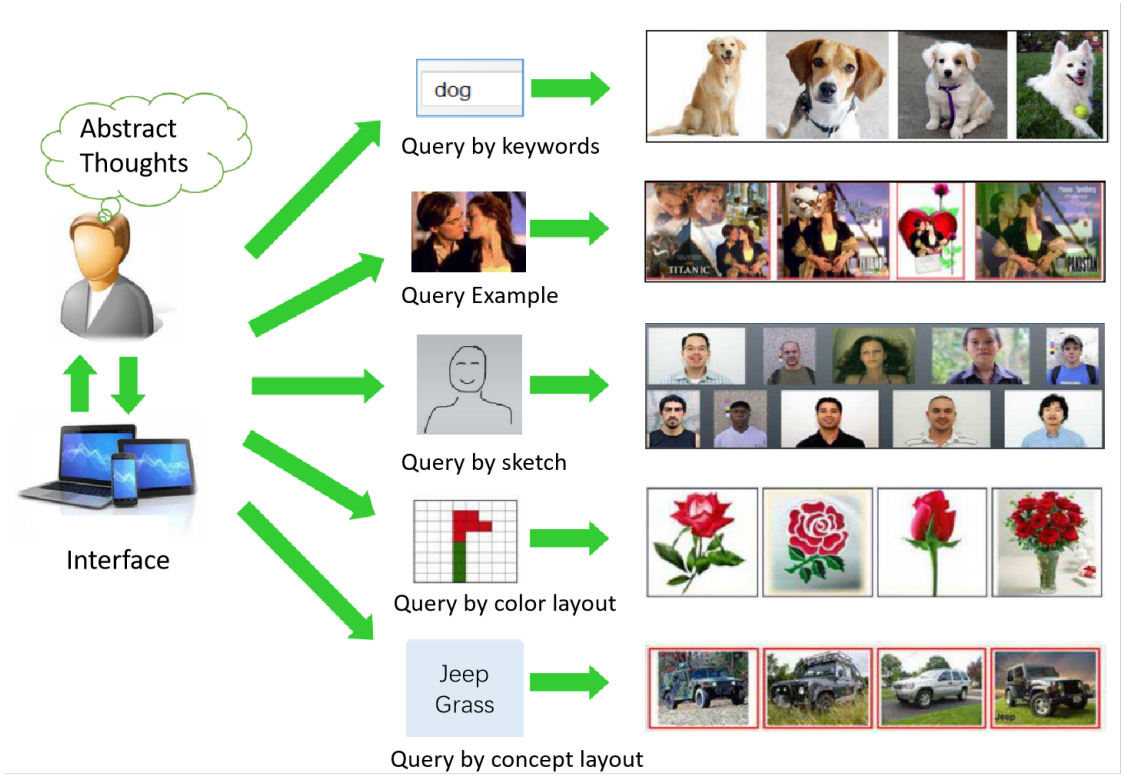


Figure 2.1: Schemes of image retrieval and their fundamental mechanisms [1].

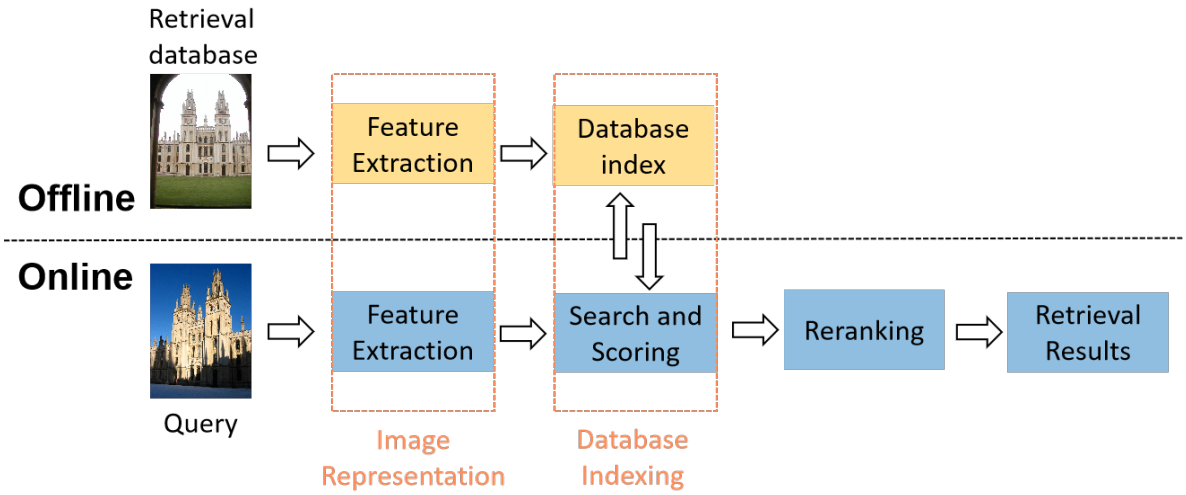


Figure 2.2: CBIR's general framework. Above and below the dashed lines are the offline and online subsystems, which share the identical image representation method.

employs the same Image representation process to extract features of the user-submitted query images and obtain the query descriptors. All database descriptors are scored according to similarity with the query descriptors, and images with a score higher than a predefined threshold are selected as first matches. Those first matches are then

filtered and scored again by the reranking algorithm. The final retrieval results are achieved by outputting them in descending order based on the re-ranking scores. From a technical viewpoint, CBIR systems are primarily dependent on image representation and database indexing/matching. This thesis focus on image representation and their enhancement.

2.2 CNN-based Extraction Methods

After elaborating on the research context of CBIR, we will now summarise the existing CNN-based feature extraction approaches and their enhancement in this section.

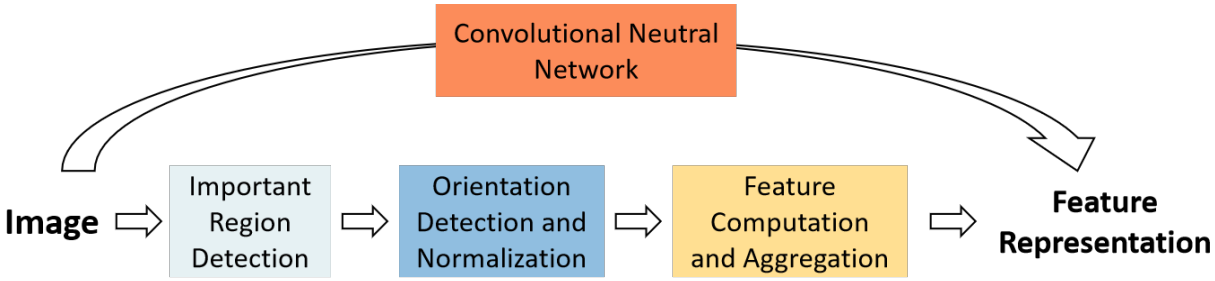


Figure 2.3: CNN based feature learning replaces the state-of-the-art pipeline of traditional hand-crafted feature representation[2].

Various handcrafted image representations such as SIFT [7] and Bag of visual Words (BoW) [8] dominated the early image feature field. In 2012, AlexNet, a deep convolutional neural network (DCNN), revolutionised the feature representation field with its groundbreaking performance in the ImageNet classification competition. Since then, deep convolutional neural network-based feature learning has replaced handcrafted local descriptors. Figure 2.3 depicts this transition. DCNNs learn abstract features from data through different architectural layers. In the past decade, CNN-based feature learning has made significant strides with the power of deep learning. In the following paragraphs, CNN-based feature extraction is further discussed in terms of the off the shelf model and fine-tuned model.

2.2.1 Off-the-Shelf models

There are several popular off the shelf backbone architectures for CNN-based feature extraction, including AlexNet [4], VGGNet [9], GoogleNet [10] and ResNet [5], as listed in table 2.1. Theoretically, CNN can be viewed as a collection of nonlinear functions composed of convolutional layers, pooling layers and nonlinear layers [11]. In a CNN, the input image is convolved with filters of the convolutional and pooling layers to extract features, beginning with low-level features and progressing to high-level features. The filters in the same layer have the same size but differing parameters.

AlexNet was the first DCNN and won the ImageNet ILSVRC competition in 2012, significantly exceeding the runner-up. AlexNet acquires features using five cascaded convolutional layers and three fully connected (FC) layers. During training and testing, input images are typically scaled to a fixed size. Inspired by AlexNet, VGGNet

Table 2.1: Off the shelf CNN models.

Models Size	Size Layers	Layers Training Set	Models
OverFeat	144M	6+3	ImageNet
AlexNet [4]			ImageNet
VGGNet [9]	138M	13+3	ImageNet
GoogleNet [10]	11M	22	ImageNet
ResNet [5]	44.6M	101	ImageNet

employs more parameters, deeper layers, and smaller convolution filters. It is trained using multiple scales, with the training images cropped and rescaled to increase retrieval feature invariance. Afterwards, GoogleNet and ResNet continue to deepen the network and avoid fully connected layers. GoogleNet substantially reduces network parameters amount, while ResNet proposes skipping connections to overcome the vanishing gradient problem during training. They won the ILSVRC 2014 and 2015 challenges respectively, indicating that the network's depth is crucial to the algorithm's outstanding performance.

Depending on how the images are fed to the CNN models, the off-the-shelf CNN models can be further divided into single-pass methods and multi-pass methods [3].

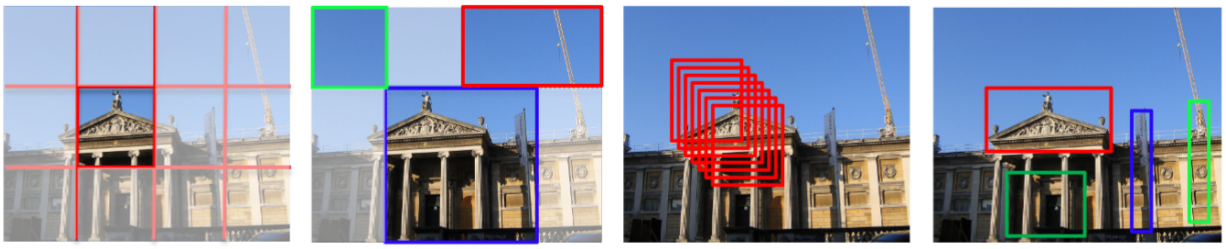


Figure 2.4: Image patch generation methods: Rigid grid, Spatial pyramid modeling (SPM), Dense patch sampling and Region proposals (RPs).[3]

The single-pass CNN method feeds the entire image into an off-the-shelf model to extract features. The advantage of this method is that the image is only fed once and extraction is relatively fast and efficient. Correspondingly, however, the single input may activate a focus on non-essential information like the background in the image, thus affecting the accuracy of the retrieval. A reliable solution is to embed and aggregate deep features to improve feature discriminability. Another option is to consider distinct regions in the feature map as different subvectors and combine them to represent the input image [12][13]. The details of those enhancement methods will be discussed in Section 2.3.

Multi-pass CNN techniques are more time-consuming than single-pass solutions because many patches at various scales must be created and supplied into the network. As demonstrated in Figure 2.4, multiscale image patches are typically generated using sliding windows [14][15] or Spatial Pyramid Model (SPM)[16].

2.2.2 Fine-tuned models

Even though off-the-shelf CNN models have achieved exceptional retrieval accuracy and precision, research into fine-tuned CNN models remains quite popular. Fine-tuned networks update and complement the pre-stored parameters of the off the shelf network. They generate image descriptors in a direct, end-to-end manner, and are quite robust of inter-class variation. The next paragraphs will explain diverse types of fine-tuned networks.

From the perspective of the target task, fine-tuned CNN networks can be divided into two types architectures: classification-based networks and validation-based networks. Classification-based networks are trained to classify architectures into predefined categories. Validation-based networks, on the other hand, are more often used for fine-tuning. There are many representatives and outstanding studies in this area. Radenovic et al. proposed replacing the fully connected layer with a MAC layer and using a 3D architectural model with camera localisation for training data mining and construction. Siamese network is also employed to compute the loss of training batches. Gordo et al. fine-tuned ternary loss networks with regional networks to improve the distinguishability of focused features.

Table 2.2: Configuration scheme for fine-tuning CNN networks in a supervised and unsupervised manner.

Type	Method	Backbone	Output Layer	Loss Function	Dimension
Supervised Fine-tuning	DELF [17]	ResNet101	Conv4 Block	CE Loss	2048
	Faster R-CNN	VGG16	Conv5	Regresion Loss	512
	SIFT-CNN[18]	VGG16	Conv5	Siamese Loss	512
	NetVLAD [19]	VGG16	VLAD Layer	Triplet Loss	256
	Deep Retrieval[20]	ResNet101	Conv5 Block	Triplet Loss	2048
Unsupervised Fine-tuning	GeM [21]	VGG16	Conv5	Siamese Loss	512
	SfM-CNN [22]	VGG16	Conv5	Siamese Loss	512
	IME-CNN [23]	ResNet101	IME Layer	Regresion Loss	2048
	MDP-CNN [24]	ResNet101	Conv5 Block	Triplet Loss	2048

From another viewpoint of the given information (image annotation or other supervisory information), fine-tuned networks can be divided into supervised fine-tuning and unsupervised fine-tuning. Supervised fine-tuning can be split into fine-tuning via classification Loss and fine-tuning via pairwise Ranking Loss based on the loss function scheme. When class labels are available for a new dataset, the best approach is to optimise the parameters of a pretrained CNN model based on cross-entropy loss, which is supervised fine-tuning based on classification. And when affinity information representing similar and dissimilar pairs is given, a fine-tuning approach based on pairwise rank loss is a much better option for learning an optimal metric. Table 2.2 lists the configuration schemes for fine-tuning CNN networks in a supervised and unsupervised manner.

2.3 CNN-based Enhancement Methods

As discussed in the preceding section, CNN-based image feature extraction networks are susceptible to variable degrees of feature defocus. Background information and distracting information from secondary objects are frequently present in the extracted features, diminishing features' representativeness and recognizability. Therefore, feature enhancement techniques aimed at optimising feature quality are a popular research direction. In this section, we review the relevant research progress regarding both feature aggregation (Section 2.3.1) and feature embedding (Section 2.3.2).

2.3.1 Feature Aggregation

In CNN networks, features in the convolutional layer maintain more structural detail. Neurons in the convolutional layer are exclusively coupled to local regions, ensuring that the generated features retain more local information. To increase the robustness of CNN features, several methods aggregate the local descriptors into a compact global feature for matching.

Sum/average pooling and max pooling are two straightforward aggregation techniques for generating global features [25][26]. The sum/average pooling is less discriminative since it considers all activation outputs of the convolution layer, diminishing the significance of highly activated features [27]. Consequently, maximum pooling approaches are appropriate for sparse features with low probability activations. These approaches perform no additional operations on the feature map before pooling and are therefore referred to as direct pooling.

Several sophisticated aggregation methods, such as channel weighting or spatial weighting of convolutional feature maps, have been investigated to optimise direct pooling. These aggregation techniques aim to emphasise the significance of focal features. Babenko et al. [28] presented aggregated convolutional features (SPoC) to obtain compact descriptors with Gaussian centres preprocessed a priori. Similar to the R-MAC approach developed by Tolias et al. [29], regions in the feature map can be treated as different subvectors. R-MAC generates a global description of an image by aggregating CNN activation properties of spatial regions with a particular aggregation algorithm. The literature [30] [20] enhances the R-MAC method by optimising the triadic ranking loss function and applying region recommendation networks to improve the region pooling technique of the original R-MAC method. As a remedy, Pang et al. [31] employ thermal diffusion during the aggregation phase to measure convolutional features and decrease the negative effects of burstiness.

2.3.2 Feature Embedding

The primary objective of feature embedding is also to increase the CNN features' identifiability and to achieve the final local/global retrieval features.

As explained previously, CNN network convolutional features are parsed as local descriptors and output as feature maps. Each vector in the feature map corresponds to a region of the input image and is referred to as a "Dense SIFT feature." Inspired by this insight, several works have employed embedding methods for SIFT features, such

as BoW [8], VLAD [32], and FV [33], to embed and encode regional feature vectors and subsequently aggregate them into a global descriptor. By translating individual characteristics to a high-dimensional space before aggregation, feature embedding increases their discriminative potential. After feature embedding, PCA reduces the dimensionality of features, while whitening reduces the co-occurrence between features.

Different embedding techniques have distinct computational features. BoW and VLAD are computed in rigid Euclidean space. Their performance is directly proportional to the number of centroids, whereas FV captures higher-order statistics and enhances the efficacy of feature embedding at the expense of larger memory costs. A more advanced concept is to integrate VLAD into a deep network in which each spatial column feature is utilised to build clusters using k-means. This concept inspired the development of NetVLAD [19].

3 Methodology

This chapter will examine the theoretical methodology utilised in this paper. We first introduce feature aggregation techniques that further improve the feature maps provided by the CNN model (Section 3.1). Following that, the augmented features' whitening and dimensionality reduction strategies are described (Section 2.3). In addition, this chapter introduces the fine-tuning CNN models with second-order attention and the combination with various loss functions (Section 3.3). Finally, the proposed solution pipeline is presented by combining the abovementioned methods (Section 3.4).

3.1 Feature Aggregation

As stated in Section 2.2, the neurons of the convolutional layer of the CNN network are only related to local regions, ensuring that the generated features retain more local information. To improve the robustness of CNN features, we need to apply feature enhancement algorithms to aggregate local descriptors into a compact global feature for matching. Pooling is a representative aggregation strategy, and three pooling methods employed in this paper are detailed in this section.

With image I as input, the pre-trained deep convolutional network outputs deep convolutional features in the final convolutional layer. The output consists of C feature maps, each with height H and width W. The input image I is then represented by a collection of $H \times W$ C-dimensional vectors, which are the deep convolutional features we are concerned with. The following aggregation methods all further process this $H \times W \times C$ feature map.

3.1.1 Sum-Pooled Convolution (SPoC)

Sum-Pooled Convolutional (SPoC) [28] descriptor is the aggregation of unencoded deep convolutional features. SPoC associates each deep convolutional feature f calculated from the image I with the spatial coordinates (x, y) corresponding to this feature's spatial position in the map stack. In this thesis, we optimize the post processing of SPoC discriptors. Figure 3.1 shows the working scheme of SPoC. The process can be distilled as follows:

1) **Sum Pooling**
   The construction of SPoC descriptors starts with the sum pooling of depth characteristics. Set the target discriminative global representation as $\psi(I)$. The similarity between a pair of images can be expressed as a scalar product between their corre-

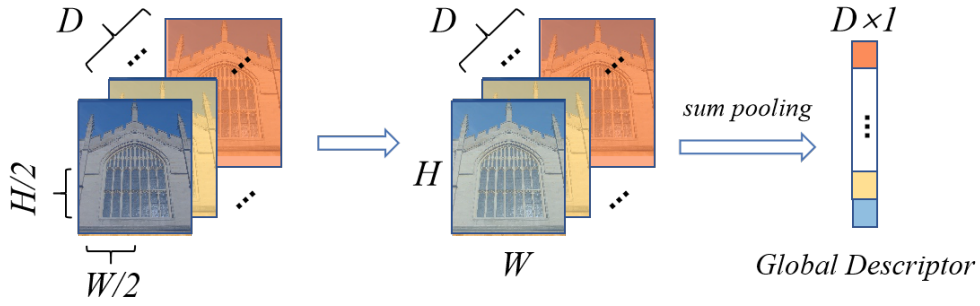


Figure 3.1: SPOC Working Scheme.

sponding identifiers.

$$\psi_1(I) = \sum_{y=1}^{H} \sum_{x=1}^{W} f_{(x,y)}, \tag{3.1}$$

$$\text{sim}\left(I_1, I_2\right) = \left\langle \psi\left(I_1\right), \psi\left(I_2\right) \right\rangle = \sum_{f_i \in I_1} \sum_{f_j \in I_2} \left\langle f_i, f_j \right\rangle, \tag{3.2}$$

2) **Centering Prior**
Next, we will concentrate on the image's focal point. In most instances, the object of interest is positioned in the image's geometric centre. With a simple weighted heuristics algorithm, we assign greater weight to the features from the centre of the feature map stack by replacing equation 3.1 with:

$$\psi_2(I) = \sum_{y=1}^{H} \sum_{x=1}^{W} \alpha_{(x,y)} f_{(x,y)}. \tag{3.3}$$

Coefficients $\alpha(w, h)$ are solely determined by the spatial coordinates h and w. We employ the Gaussian weighting technique in particular:

$$\alpha_{(x,y)} = \exp\left\{ -\frac{\left(y - \frac{H}{2}\right)^2 + \left(x - \frac{W}{2}\right)^2}{2\sigma^2} \right\}. \tag{3.4}$$

3) **Post Processing**
After obtaining the representation, additional whitening, dimension reduction, and normalisation will be performed. The original SPoC [28] methodology employs L2 normalisation and PCA dimension reduction. This article optimises the post-processing further by using both learnt whitening and end-to-end whitening, producing superior outcomes. These post-processing techniques will be elaborated upon in Section 3.3.

3.1.2 Regional Maximum Activation of Convolutions (RMaC)

RMAC [29] pooling is based on utilising a variable window for sliding windows. RMAC, unlike other sliding windowing algorithms, does not perform sliding windowing on the

image but rather on the feature map, which significantly accelerates feature extraction. Figure 3.2 depicts the RMAC Pooling sliding window method.

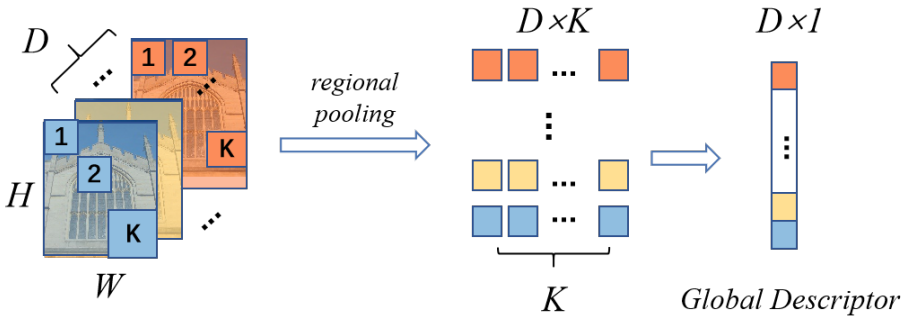


Figure 3.2: RMAC Working Scheme.

We now consider defining a collection of R regions of varying sizes on the CNN feature map. We collected samples of the square regions at L distinct scales. The regions were sampled uniformly so that the overlap between successive sections was close to 40%. We uniformly sampled $l(l+m-1)$ rectangles of width $2\min(W,H)/l+1$ at every other scale l. Figure 3.3 depicts the rationale for this subscale sampling.

The related feature vectors are computed in each region and then post-processed using l2 normalisation and the whitening technique described in Section 3.3. Finally, a single global feature vector is generated by summing the set of all regional feature vectors.

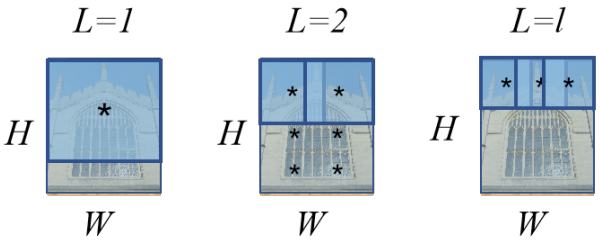


Figure 3.3: Region Sampling Scheme. There are three different levels of square region ($L = 1, 2, ...l$). When $L = 1$, the square region is maximal and its height and width is equal to the $\min(H, W)$. Two uniform regions was sampled. When $L = 2$,the width of the sample region is $2/3\min(H, W)$ and 6 regions are sampled. When $L = l$, regions of width $2\min(W, H)/l+1$ are sampled.

3.1.3 Generalized Mean Pooling (GeM)

Generalized Mean (GeM) [21] Pooling is a compromise that incorporates the maximum and average pooling methods. The key to attaining this balance is the learnable pooling parameter p. By altering the value of p, GeM can focus on different fine-grained image regions. Using the input feature map $\mathcal{X}$, the output features of GeM pooling can be

represented as a single D-dimensional feature vector ψ. GeM can be understood as maximum pooling when pkß and average pooling when $pk = 1$.

$$\psi^{(g)} = \left[\psi_1^{(g)} \dots \psi_k^{(g)} \dots \psi_K^{(g)}\right]^\top, \quad \psi_k^{(g)} = \left(\frac{1}{|\mathcal{X}_k|}\sum_{x\in\mathcal{X}_k} x^{p_k}\right)^{\frac{1}{p_k}}. \tag{3.5}$$

The pooling parameter pk may be specified manually or learned. This process is differentiable and serves as a part of the backpropagation. Equation 3.6 and 3.7 explains this process. This thesis discovers the ideal range of values for the pooling parameter pk using a series of experiments (Section 4.2). Similarly, the global features derived from GeM are post-processed using the whitening method described in Section 3.3.

$$\frac{\partial \psi_k}{\partial x_i} = \frac{1}{|\mathcal{X}_k|}\psi_k^{1-p_k} x_i p_k - 1, \tag{3.6}$$

$$\frac{\partial \psi_k}{\partial p_k} = \frac{\psi_k}{p_k^2}\left(\log\frac{|\mathcal{X}_k|}{\sum_{x\in\mathcal{X}_k} x^{p_k}} + p_k \frac{\sum_{x\in\mathcal{X}_k} x^{p_k}\log x}{\sum_{x\in\mathcal{X}_k} x^{p_k}}\right). \tag{3.7}$$

As a summary, the whole process of GeM is shown in Figure 3.4

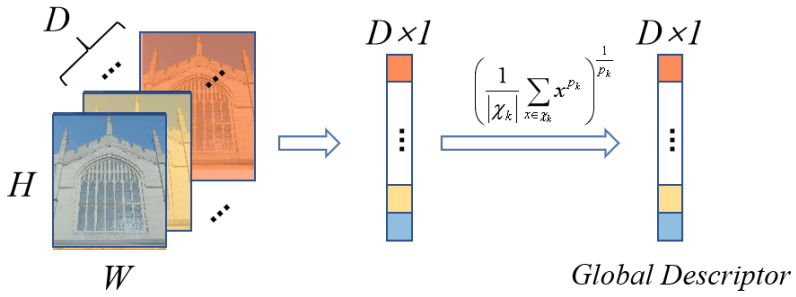


Figure 3.4: The Whole Process of GeM.

3.2 Whitening and Dimension Reduction

In this section, the post-processing of feature vectors is considered. After pooling aggregation, feature descriptors need to be whitened.

The whitening of the data representation is vital for image retrieval.High-dimensional features carry a wealth of information. Whitening removes the correlation between features by transforming the original data into a set of linearly independent representations of each dimension through a linear transformation. Its significance lies in reducing the dimensionality of the feature vectors while eliminating redundant information. The benefits of whitening are further emphasised in the case of CNN-based descriptors [34][35].

Whitening in deep learning usually learns from generative models unsupervised via Principal Components Analysis (PCA). In contrast, this thesis applies a more efficient discriminative learning whitening [21], and a fully integrated end-to-end whitening [30], respectively.

3.2.1 Discriminative Learned Whitening

The discriminative learned whitening approach utilises labelled data provided by 3D models and linear discriminant projections first described by Mikolajczyk [36]. Two components comprise the projection: whitening and rotation. The whitening coefficient is the inverse of the square root of the matching pairs covariance matrix $C_S^{-\frac{1}{2}}$.

where

$$C_S = \sum_{Y(i,j)=1} (\overline{\mathbf{f}}(i) - \overline{\mathbf{f}}(j))(\overline{\mathbf{f}}(i) - \overline{\mathbf{f}}(j))^\top. \tag{3.8}$$

The rotation part is the PCA of the nonmatching pairs covariance matrix in the whitened space eig $\left(C_S^{-\frac{1}{2}} C_D C_S^{-\frac{1}{2}}\right)$, where

$$C_D = \sum_{Y(i,j)=0} (\overline{\mathbf{f}}(i) - \overline{\mathbf{f}}(j))(\overline{\mathbf{f}}(i) - \overline{\mathbf{f}}(j))^\top. \tag{3.9}$$

Eventually, the projection can be expressed as

$$P = C_S^{-\frac{1}{2}} \text{ eig} \left(C_S^{-\frac{1}{2}} C_D C_S^{-\frac{1}{2}}\right), \tag{3.10}$$

when $P^{\top^\top}(\overline{\mathbf{f}}(i) - \mu)$, and μ is the pooled feature vector.

To restrict the dimensions of the descriptors to D, we only use the eigenvectors corresponding to the D greatest eigenvalues. $l2$ subsequently normalises the projected vectors.

This whitening method uses all available training pairings while focusing on discriminative information between matched and unmatched pairs. Discriminative learned whitening finally makes a superior performance compared to the commonly used PCA whitening.

3.2.2 End to End Whitening

The end-to-end whitening [30] method is implemented through a whitening layer at the network's end. The whitening is optimised end-to-end during fine-tuning, using the same training data in batch mode and the convolutional filter.

This whitening layer is modelled directly from a fully connected layer and contains the projection and rotation in Section 3.3.1. In other words, discriminative learning whitening is integrated into a layer that whitens the training batch directly. This approach requires substantial training data. It is highly integrated and straightforward to apply to multiple datasets, which is advantageous. Nonetheless, it has the disadvantages of being computationally intensive and slow to converge.

Regarding final mAP performance, discriminative learnt whitening and end-to-end whitening are comparable, and both significantly outperform PCA whitening.

3.3 Optimisation strategies

Although off-the-shelf CNN models have achieved remarkable retrieval accuracy, there is still a need for research on fine-tuning CNN models for several specific tasks. This

thesis combines many outstanding fine-tuning researches to optimise existing CNN models in training, network layer and loss function design.

3.3.1 Fine-tuning Pretrained CNN

Deep learning frequently demands large training datasets, substantial computational resources, and long model training time. In practice, models are rarely taught from scratch but frequently pre-trained. Typically, pre-trained models can already extract both shallow local features and deep abstract characteristics. Fine-tune adjusts the network model's structure, selectively loads pre-training parameters, and then retrains the model using the task-specific dataset. This way allows us to obtain a high-performing model with a minimal amount of data and computational expense.

In this research, we have chosen popular CNNs, such as ALexNet, ResNet, and VGGNet, which are trained using the considerable training set ImageNet, as the pre-trained networks. In this article, we utilise the Sfm120k and Google Landmarks version datasets, which have a comparable data structure to ImageNet but are less in size.

Considering the medium size of the training dataset and the medium closeness to the original dataset, we maintain the model structure and initial parameters from the pre-trained network for retraining. Since the pre-trained network already has suitable beginning parameters, we use a slower learning rate during fine-tuning to avoid altering the parameters too rapidly. A lower learning rate also makes the model's learning more precise.

3.3.2 Second-order attention layer

The input image processed by a fully convolutional network generates an $H \times W \times D$ feature map f. We undertake more fine-tuning in the following discussion and experiments using GeM pooling as the baseline. Initially, the feature map is aggregated into the global descriptor vector $d = GeM(f, p)$, where p is the pooling parameter:

$$\mathrm{GeM}(\mathbf{f}, p) = \left(\frac{1}{N} \sum_{i=0}^{N} f_i^p \right)^{\frac{1}{p}}. \tag{3.11}$$

As previously stated, p in equation 3.8 is adjustable, and p can be changed for each local contribution of feature map F to the global descriptor d according to its associated feature activation. The GeM[21] method is thereby regarded as a first-order measure. It assumes that each location on the map is independent and dismisses the influence of each spatial characteristic relative to the others. In addition, the global descriptor d has a finite receptive region, resulting in a relatively restricted contribution from each local feature. The second-order attention (SOA) layer [37] is a good method to incorporate spatial information into the feature pooling. Figure 3.5 visualize the SOA layer structure and our implementation process.

First, we generate two projections of feature map f which is query q and key k. Then q and k are flattened and transformed into a matrix of $D \times HW$.The second-order attention map z is then computed as

$$\mathbf{z} = \mathrm{softmax}\left(\alpha \cdot \mathbf{q}^\top \mathbf{k}\right). \tag{3.12}$$

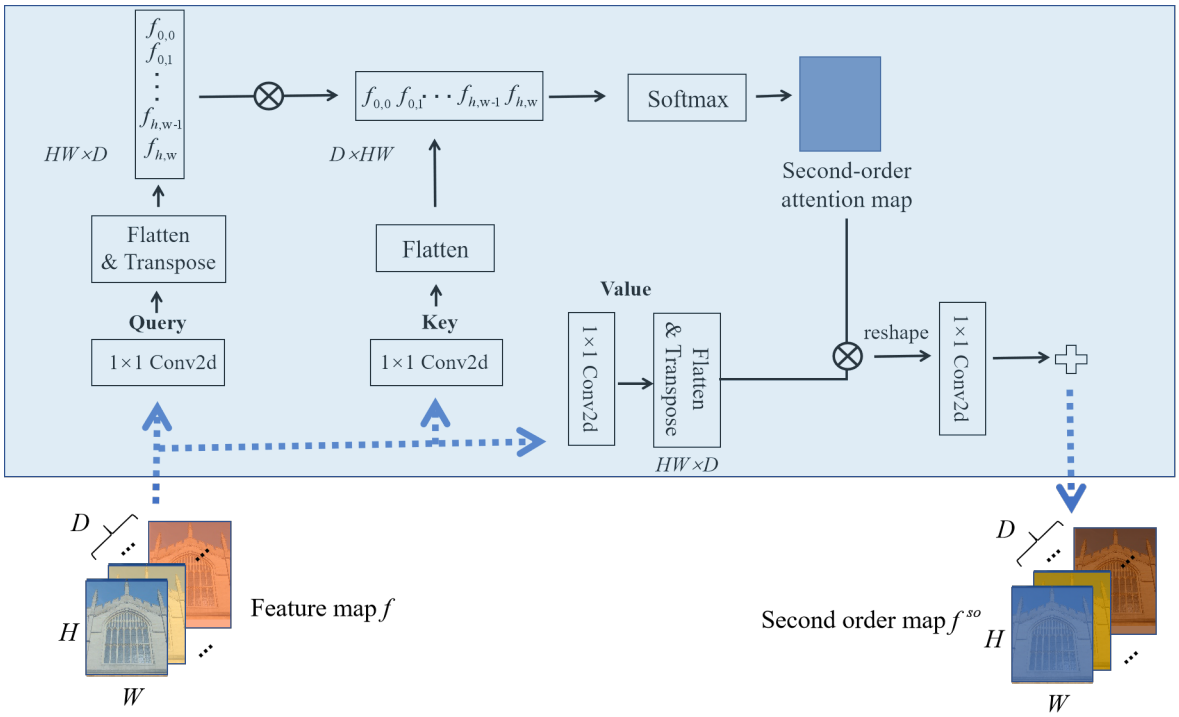


Figure 3.5: The SOA Layer.

where α is a scale factor and z has the shape $HW \times HW$, which allows the feature map f to combine all the local features of the whole map. Then similarly to q and k, a third projection value head v of f is generated with a shape $HW \times D$. Finally, we can obtain the second-order attentional feature map f^{so} from the first-order attentional feature map. ψ is another 1×1 convolution to control the influence of the attention. In the feature map f^{so}, a new feature $f^{so}_{i,j}$ is re-represented as a function of the features from all positions in f, and therefore interpretable as a function of the whole input image:

$$\mathbf{f}^{so} \equiv \mathbf{f} + \psi(\mathbf{z} \times \mathbf{v}), \tag{3.13}$$

$$f^{so}_{i,j} = g\left(\mathbf{z}_{ij} \odot \mathbf{f}\right). \tag{3.14}$$

Following the final second-order attention layer, the new feature map f^{so} will be used as input to the pooling layer for the subsequent feature augmentation and pooling step. The final global features generated by the GeM pooling will have more comprehensive local information:

$$\mathrm{GeM}\left(\mathbf{f}^{so}p\right) = \left(\frac{1}{N}\sum_{i=0}^{N} f_i^{so^p}\right)^{\frac{1}{p}}. \tag{3.15}$$

In this thesis, we further optimised the Second-Order Attention (SOA) layer. According to the research, the insertion position of the SOA layer may be any fully-convolutional block in the CNN. This selection ranges from cov1 to cov5 for the

ResNet101 utilized in this work. T Ng et al. [37] chose to link the SOA to both cov4 and conv5 and attained high retrieval precision.

Table 3.1: Retrieval precision using various SOA embedding schemes. ResNet101-GeM serves as our baseline. Results for the Medium and Hard regimens are provided in mAP.

Components		Medium		Hard	
		ROxford	*RParis*	*ROxford*	*RParis*
None (Baseline)	ResNet101-GeM	67.3	80.6	44.3	61.5
Spatial (SOA)	ResNet101+SOA 4	68.2	81.0	45.7	62.3
	ResNet101+SOA 5	68.3	81.3	45.9	62.8
	ResNet101+SOA 4,5	68.6	81.4	46.9	63.7

This thesis investigates further the selection of SOA block insertion sites. Table 3.1 displays the retrieval precision outcomes for various sites. Comparing inserting SOA after *cov*4 and *cov5* to inserting it only after *cov4* or *cov5* yields a tiny (0.2%) improvement in accuracy. Furthermore, when we insert SOA after only one fully-convolutional block, *cov5* improves mAP by 1.1%, but *cov4* only improves mAP by 0.9%. In order to reduce the necessary computing resources, we have decided to insert the SOA block just after *cov5*. This embedding optimization maximizes retrieval precision while enhancing computing efficiency. In this paper, we refer to this optimized method of applying second-order attention information as Second-Order Information (SOI).

3.3.3 Loss Functions

Suppose we have two images and have extracted the corresponding features using CNN. We want to conduct a simple comparison task, i.e., determine if the two images correspond to the same object. How would we typically approach this problem? An easy approach is to compare the similarity of the extracted vectors. However, the "inter-class" boundary of the features recovered by the CNN is not particularly large, making it easy for classification to fail when an adversarial sample is introduced.

Several matching network architectures and accompanying loss functions have been proposed to overcome this issue. In this paper, four loss functions are applied and compared: Contrastive Loss [38], Triplet Loss [39], Listwise Average Precision (AP) Loss [40] and Second-order Similarity (SOS) Loss [37].

3.3.3.1 Contrastive Loss

Before introducing contrastive loss [38], it is essential to have a solid grasp of Siamese Networks. A Siamese Network is the integration of two similar or identical sub-networks, each of which receives input and produces the appropriate feature vector. When the sub-networks are identical, they can be constructed by a single network, followed by the manual construction of the batch pairs. Figure 3.6 depicts the structure of a Siamese network.

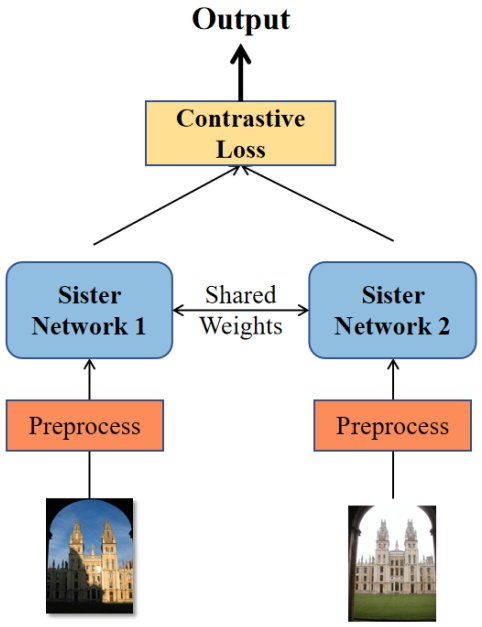


Figure 3.6: The Siamese Network.

Siamese networks frequently apply contrastive loss to process pairs of data efficiently. In contrast to normal learning systems, where the loss function is a sum over samples, the contrastive loss function runs across pairs of samples. Let X_1, X_2 represent a pair of input vectors for a Siamese network and let Y represent the binary label associated with the input pairs. If X_1, X_2 are considered similar, $Y = 0$; if they are considered distinct, $Y = 1$. The learnable parametric distance function D_W is defined as the Euclidean distance between the output G_w.

$$D_W\left(\vec{X}_1, \vec{X}_2\right) = \left\| G_W\left(\vec{X}_1\right) - G_W\left(\vec{X}_2\right) \right\|_2, \tag{3.16}$$

while loss L_W is given in its general form as

$$\mathcal{L}(W) = \sum_{i=1}^{P} L\left(W, \left(Y, \vec{X}_1, \vec{X}_2\right)^i\right), \tag{3.17}$$

$$L\left(W, \left(Y, \vec{X}_1, \vec{X}_2\right)^i\right) = (1-Y)L_S\left(D_W^i\right) + Y L_D\left(D_W^i\right), \tag{3.18}$$

where x is the $i-th$ labelled sample pair, L_S and L_D are partial loss functions for pairs of similar and dissimilar points respectively, and P is the number of training pairs (which may be as large as the square of the number of samples). L_S and L_D must be designed to minimise L relative to W so that similar pairs have the lowest D_W feasible and dissimilar pairs have the highest D_W possible.

Further, the loss function can be finally expressed as

$$L\left(W, Y, \vec{X}_1, \vec{X}_2\right) = \quad (1-Y)\frac{1}{2}\left(D_W\right)^2 + (Y)\frac{1}{2}\left\{\max\left(0, m - D_W\right)\right\}^2. \tag{3.19}$$

As a summary, the whole structure of contrastive loss is shown in Figure 3.7.

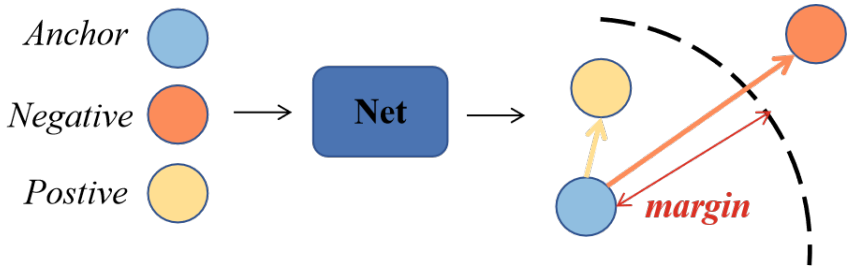


Figure 3.7: The Structure of contrastive Loss.

3.3.3.2 Triplet Loss

Similar to contrastive loss, the objective of triplet loss [39] is to regulate the distance between features to distinguish matches. Triplet loss seeks an embedding $f(x)$ from an image x into a feature space $\mathbb{R}^d$. The embedding is denoted by the expression $f(x) \in \mathbb{R}^d$. It encapsulates a picture x within a d-dimensional Euclidean space. In addition, the embedding is constrained to exist on the d-dimensional hypersphere, $i.e. \|f(x)\|_2 = 1$

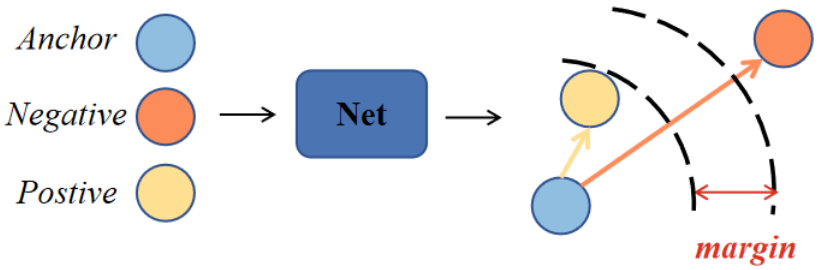


Figure 3.8: The structure of triplet Loss.

In contrast to contrastive loss, which emphasises the distance between positive examples and anchors, triplet loss concerns the difference in distance between positive and negative images. In other words, for an image x_a (anchor), we want to ensure that it is close to the similar image x_p (positive) and distant from the dissimilar picture x_n (negative) while maintaining a noticeable difference(margin) between the two kinds of distances. Figure 3.8 illustrate this concept.

Thus we want

$$\|x_i^a - x_i^p\|_2^2 + \alpha < \|x_i^a - x_i^n\|_2^2, \forall \left(x_i^a, x_i^p, x_i^n\right) \in \mathcal{T}, \tag{3.20}$$

where α is a margin that is enforced between positive and negative pairs. $\mathcal{T}$ is the set of all possible triplets in the training set and has cardinality N. Then the loss is given as

$$L = \sum_{i}^{N} \left[\left\| f\left(x_i^a\right) - f\left(x_i^p\right) \right\|_2^2 - \left\| f\left(x_i^a\right) - f\left(x_i^n\right) \right\|_2^2 + \alpha \right]_+ . \tag{3.21}$$

The corresponding gradient calculation formula for the three samples(anchor, postive and negative) is

$$\begin{aligned} \frac{\partial L}{\partial f\left(x_i^a\right)} &= 2 * \left(f\left(x_i^a\right) - f\left(x_i^p\right)\right) - 2 * \left(f\left(x_i^a\right) - f\left(x_i^n\right)\right) = 2 * \left(f\left(x_i^n\right) - f\left(x_i^p\right)\right) \\ \frac{\partial L}{\partial f\left(x_i^p\right)} &= 2 * \left(f\left(x_i^a\right) - f\left(x_i^p\right)\right) * (-1) = 2 * \left(f\left(x_i^p\right) - f\left(x_i^a\right)\right) \\ \frac{\partial L}{\partial f\left(x_i^n\right)} &= 2 * \left(f\left(x_i^a\right) - f\left(x_i^n\right)\right) * (-1) = 2 * \left(f\left(x_i^a\right) - f\left(x_i^p\right)\right) . \end{aligned} \tag{3.22}$$

3.3.3.3 Listwise Average Precision (AP) Loss

The aforementioned contrastive and triplet losses are pairwise losses that use the generation of a batch pair and the computation and adjustment of the Euclidean distance between the pairs to determine and learn a match. These methods do not directly refer to the matching scores, i.e. the average accuracy; thus, their contribution to the final ranking is limited. Several methods have been proposed based on this inspiration to directly learn ranking accuracy. They are collectively referred to as ranking loss.

This thesis's listwise AP loss [40] is an efficient ranking loss. The triple loss produces gradient updates based on a limited sample size, which may not be consistent with the ranking criteria. In contrast, the listwise loss considers many photos and optimises the average accuracy obtained from these images. Figure 3.9 illustrates this unique type of processing.

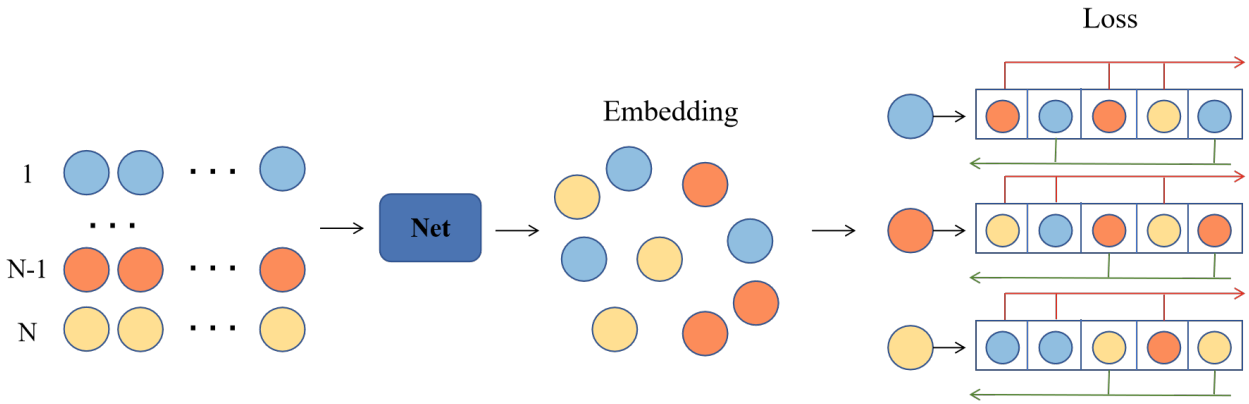


Figure 3.9: AP Loss Structure. Different colours represent different classes. Instead of being combined as pairs, every image is regarded as an independent input to compare and compute the average precision of matches with all other images.

As one of the Instance Retrieval (IR) metrics, AP does not depend on thresholds, rank positions, or the number of relevant images. Therefore, it's simpler and more suitable for different queries to apply AP than other metrics. AP can be expressed as a function of S_q and Y_q. S_q denotes the similarity of query q and $Y_q in 0, 1$ denotes the quality of the rank $R(S_q)$ relative to the relevance of the actual image. Y_q is 1 if query q is relevant to the matching image, and 0 otherwise.

$$\mathrm{AP}\left(S^q, Y^q\right) = \sum_{k=1}^{N} P_k\left(S^q, Y^q\right) \Delta r_k\left(S^q, Y^q\right). \tag{3.23}$$

where ΔP_k is the precision at rank k, i.e. the proportion of relevant items in the k first indexes, which is given by:

$$P_k\left(S^q, Y^q\right) = \frac{1}{k} \sum_{i=1}^{k} \sum_{j=1}^{N} Y_j^q \mathbb{1}\left[R\left(S^q, i\right) = j\right], \tag{3.24}$$

Δr_k is the incremental recall from rankings k 1 to k, which is given by:

$$\Delta r_k\left(S^q, Y^q\right) = \frac{1}{N^q} \sum_{j=1}^{N} Y_j^q \mathbb{1}\left[R\left(S^q, k\right) = j\right]. \tag{3.25}$$

The difficulty with Listwise AP loss is how to translate the discrete average accuracy into a continuously differentiable approximation. We divide the interval $[-1, 1]$ into $M - 1$ intervals of equal size, each with metric $\Delta = \frac{2}{M-1}$. The $mth (1 \leq m \leq M)$ interval is centred on b_m. First, the values of p_m and Δr_k are calculated for each bin. A soft assignment is then used as a replacement for the metrics. The function $\delta : \mathbb{R} \times \{1, 2, \ldots, M\} \rightarrow [0, 1]$ is defined by the fact that each $\delta(\cdot, m)$ is a triangular kernel of width 2Δ centred on b_m, i.e.

$$\delta(x, m) = \max\left(1 - \frac{|x - b_m|}{\Delta}, 0\right). \tag{3.26}$$

Thus, the quantized precision Pm and incremental recall rm are obtained for each bin m as follows:

$$\begin{aligned} \hat{P}_m\left(S^q, Y^q\right) &= \frac{\sum_{m'=1}^{m} \delta\left(S^q, m'\right)^\top Y^q}{\sum_{m'=1}^{m} \delta\left(S^q, m'\right)^\top \mathbf{1}}, \\ \Delta \hat{r}_m\left(S^q, Y^q\right) &= \frac{\delta\left(S^q, m\right)^\top Y^q}{Nq}, \end{aligned} \tag{3.27}$$

The resulting quantized average precision, represented by AP_Q, is a smooth function with respect to S_q, as shown by:

$$\mathrm{AP_Q}\left(S^q, Y^q\right) = \sum_{m=1}^{M} \hat{P}_m\left(S^q, Y^q\right) \Delta \hat{r}_m\left(S^q, Y^q\right). \tag{3.28}$$

3.3.3.4 Second-order Similarity (SOS) Loss

Second-order Similarity (SOS) Loss is a loss calculation method applied to global descriptors proposed by T. Ng et al [37].and inspired by SOS Net [41]. It does not independently serve as a loss function but is combined with triplet loss.

The Triplet loss proposed in Equation 3.18 is used as a first-order similar loss $\mathcal{L}_{LOS}$:

$$\mathcal{L}_{LOS} = L = \sum_{i}^{N} \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]_+ . \tag{3.29}$$

To maximise the utilisation of spatial information, the distances between input pairs are utilised secondarily. The definition of second-order similarity loss $\mathcal{L}_{SOS}$ is as follows:

$$\mathcal{L}_{SOS} = \frac{1}{|\{(f(x_i^a), f(x_i^p), f(x_i^n))\}|} \sqrt{\sum_{\{(f(x_i^a), f(x_i^p), f(x_i^n))\}} \left(\|f(x_i^a) - f(x_i^n)\|^2 - \|f(x_i^p) - f(x_i^n)\|^2 \right)^2}. \tag{3.30}$$

The final objective loss function is a mixture of first- and second-order loss for global descriptors produced using second-order spatial attention, balanced by λ:

$$\mathcal{L} = \mathcal{L}_{FOS} + \lambda \mathcal{L}_{SOS}. \tag{3.31}$$

3.4 A Summary: Proposed Feature Extraction Pipeline

Based on these methods presented in this chapter, this thesis combines, optimizes and finally proposes the complete pipeline for feature extraction in image retrieval.

At the same time, the pipeline is subdivided into baseline and fine-tuned, depending on whether the SOI block is used as a fine-tuning method, as shown in Figure 3.10 and Figure 3.11.

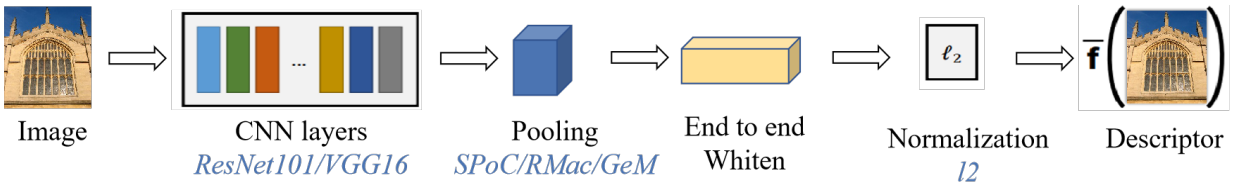


Figure 3.10: Feature extraction baseline pipeline. The technical approach used for a particular module is marked in blue.

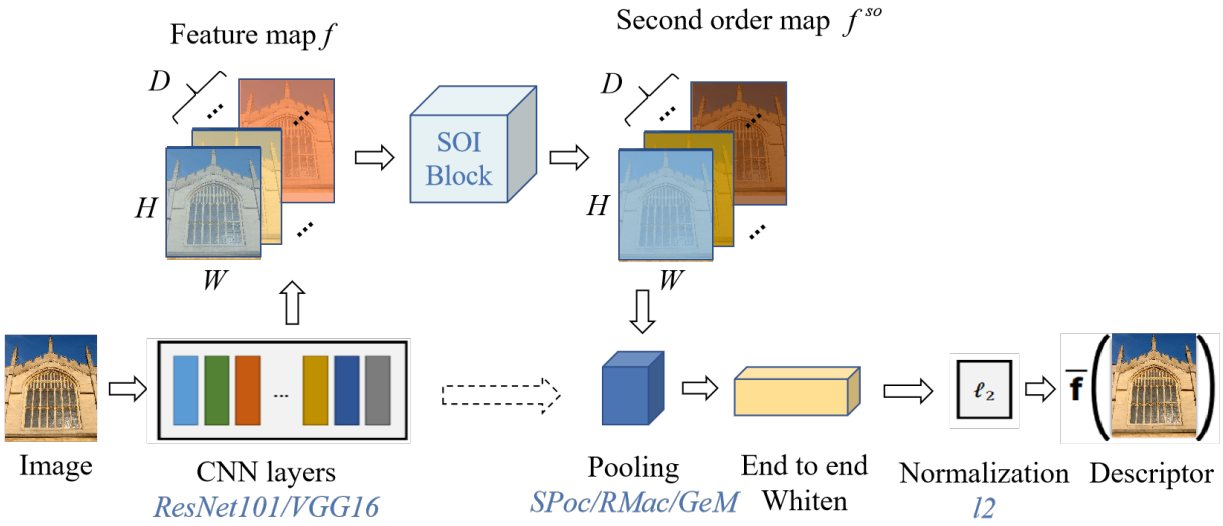


Figure 3.11: Feature extraction fine-tuned pipeline. The technical approach used for a particular module is marked in blue. The detailed structure of the SOA Block was described in the previous Section 3.2.1.

Our suggested feature extraction pipeline includes the following distinctions and enhancements over existing methods from literature:

1. Pre-trained CNNs are often applied directly to image retrieval with a large training dataset as a basis and a reliable network structure and parameters. However, pre-trained CNNs are frequently not well adapted to the target dataset's features when confronted with specific retrieval tasks. Therefore, we use new medium-sized datasets, retain the network settings, and fine-tune the pre-trained CNN with a lower learning rate. The fine-tuned CNN significantly increases the target retrieval accuracy.
2. AP Loss[40] is a ranking loss type that uses mAP as the sole basis for learning. It has a distinct learning mechanism compared to pairwise losses, like contrastive loss and triplet loss. In previous studies, AP loss has been less combined with feature enhancement methods which pairwise loss often matches. This thesis investigates the combination of AP Loss with various pooling aggregation methods and whitening techniques to achieve good retrieval mAP.
3. Second Order Attention (SOA)[37] is a method that enhances the utilization of interaction information between feature map features. This thesis investigates and optimizes the SOA's incorporation and application, proposing the SOI. SOI optimizes the embedding position by removing the *conv_5* layer from SOA, which contributes little to the final mAP. Through this optimization, we simplify the structure of SOA without sacrificing accuracy.
4. Pooling, whitening and second-order information application are popular research topics in feature enhancement, and many representative approaches exist. However, different combinations of these approaches have not been fully explored. This thesis investigates the application of a combination of various feature enhancement methods

in an attempt to obtain all the advantages to produce better retrieval results.

4 Experiments and Results

After examining the theoretical underpinnings of all image representations and CNN models, we will set up our experiments in this part. First, we specify the dataset utilised in our experiments, the data pre-processing techniques, and the metrics used to evaluate the network's performance. Then, we explain the architecture and strategies of our neural network model and compare the validation experiments to demonstrate why we built it up in this manner. Finally, the loss function design and hyperparameter settings for Baseline and Fine-tuned models' training implementations are presented.

4.1 Experiments Setup

In this section, we specify the experimental setup. We begin by describing the dataset and the preprocessing steps involved in the experiment. Then, the evaluation metrics applied to compare the network's performance will be discussed.

4.1.1 Datasets

The datasets utilised in this chapter's studies are categorised as model training and test datasets. The training set comprises the large-scale SfM 120k [21] and Google Landmarks datasets [42], while the test dataset comprises the widely used Oxford5k [43], Paris6k [44], and their ideally cleaned versions ROxford5k and RParis6k [45].

Structure from Motion(Sfm) 120K

The Sfm 120k [21] dataset employs the geolocation information and camera information contained in the images for 3D modelling using the Structure from Motion method. The dataset also employs SfM to identify similar structures from unsupervised data as a training set, avoiding manual annotation. Positive samples are selected to observe sufficient 3D Points in conjunction with the query photographs, while negative samples are derived from various 3D models. The entire procedure is automated. The complete collection produced 1,474 3D models that were recreated. After deleting overlapping models, there are 713 3D models with more than 163 data points. The 3D models contain between 25 and 11k pictures.

GoogleLandmarks version2

GoogleLandmarks [42] represents a new standard for large-scale, fine-grained instance recognition and image retrieval in artificial and natural landmarks. The Google Landmarks Collection V2 contains over 5 million images and over 200,000 unique instance labels, making it the most extensive dataset for instant recognition. It is separated into three subsets: (i) 118k query images annotated with ground truth, (ii)

4.1 million training images of 203k landmarks with labels that can be used for training, and (iii) 762k indexed images of 101k landmarks. This paper uses the training subset in the training procedure. In addition, the Google Landmark Dataset V2 contains landmarks from 246 of the 249 nations included in the ISO 3166-1 country code list.

Oxford-5k

Oxford-5k [43] includes 5,062 photographs of 11 Oxford structures. Each image was acquired from Flickr searching for a particular Oxford landmark. These image sets were manually annotated to provide thorough ground truth for eleven distinct landmarks, each represented by five potential queries, which gives 55 queries against which an object retrieval system can be assessed. All of the photographs in Oxford-5k have a resolution of 1024 by 768 pixels, making him a significant benchmark in image retrieval.

Paris-6k

Paris-6k [44] consists of 6412 photographs gathered from Flickr. It is organised into 12 sections, each focusing on a distinct architectural style in Paris. The collection includes 500 images of evaluation queries and 55 queries with bounding boxes. The same four types of labels annotate the photos as in the Oxford-5k dataset. Similar to Oxford-5k, all images in Paris-6k are also of high resolution (1024*768). The annotation and evaluation processes for Oxford-5k and Paris-6k were recently revised. New query and interference photos were added to both datasets, resulting in the Revisited Oxford and Revisited Paris datasets, which are discussed in greater detail below. Due to the prevalence of Oxford-5k and Paris-6k, we concentrated on evaluating the performance of the original datasets.

Revisited Oxford-5k and Revisited Paris-6k

Revisited Oxford-5k (ROxford-5k) and Revisited Paris-6k (RParis-6k) [45] are further optimisation of the Oxford-5k and Paris-6k datasets. In the new datasets, extra effort was made to ensure the ground truth's dependability when creating new annotations for both datasets. Formerly erroneous marks are eliminated. Three additional protocols with varying detection difficulty levels are introduced: easy, medium, and hard. The images within each query group are manually tagged and relabeled under the various detection techniques. Table 4.1 illustrates the effects of reassigning labels to the original dataset.

Table 4.1: Reassignment of annotations to the original Oxford-5k and Paris-6k datasets.

$\mathcal{R}$ oxford				
Labels	Easy	Hard	Uncl.	Neg.
Positive	438	50	93	1
Junk	50	222	72	9
Negative	1	72	133	63768

$\mathcal{R}$ Paris				
Labels	Easy	Hard	Uncl.	Neg.
Positive	1222	643	136	6
Junk	91	813	835	61
Negative	16	147	273	71621

These protocols enable a fair comparison between various approaches. Fifteen new and complex queries were introduced for each dataset. The images in these new queries' are all related to the original landmarks but are more challenging to identify.

4.1.2 Data Preprocessing

The primary objective of image preprocessing is to increase the accuracy of feature extraction and picture identification by removing extraneous information from the image, boosting the detectability of relevant information, and simplifying the data as much as possible. The preprocessing of image matching in deep learning primarily consists of geometric transformation and normalization.

The objective of picture geometry transformation is to standardize the size of the input image to correspond with the fixed input size of the neural network. On the other hand, image normalisation increases the image quality, allowing it to display more detail and increasing its distinction. In addition, it selectively highlights or suppresses features of the image that are of interest. This section describes how the experiment handles the geometric transformation and normalization process.

First, we intentionally set the image scale parameter imgsize, which specifies the input image's longest side size and is typically set to 1024. This transformation is implemented using the thumbnail function of the Pillow package. It allows us to scale the image while maintaining its aspect ratio and without surpassing the original image's dimensions.

After the picture size has been set, the Transform function module is used to execute an additional transformation. $Transform.ToTensor$ turns the image range from (0,255) to (0,1), and then $Transform.Normalization$ transforms the range from (0,1) to (-1,1). Specifically, the $Transform.Normalization$ function executes the following operations for each channel:

$$Image = (Image - mean)/std. \tag{4.1}$$

Figure 4.1 depicts the geometrically converted and normalised image. It will be given to the CNN network for the subsequent feature extraction stage.

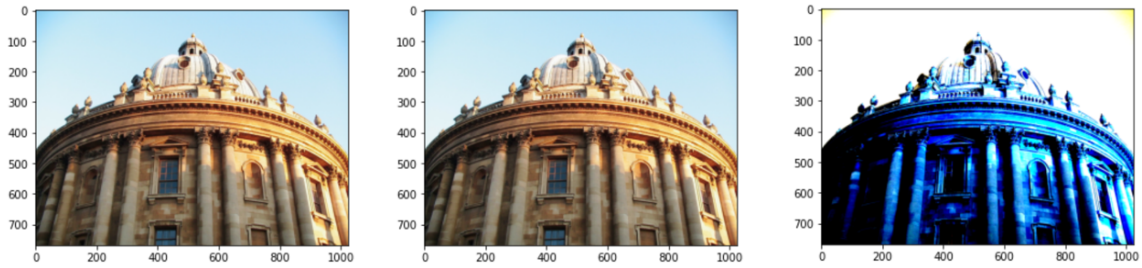


Figure 4.1: The image preprocessing is illustrated, from left to right, with the original image, the scale-transformed image, and the tensor normalized image.

4.1.3 Evaluation Protocol

This thesis evaluates image retrieval results based on the metrics mean average precision (mAP) and extract time per query image, which capture precision and efficiency,

respectively. In addition, the multi-scale evaluation protocol is also described in this section.

4.1.3.1 Mean Average Precision

Mean Average Precision (mAP) is a performance metric for object detection models. Before introducing the definition and calculation of mAP, it is necessary to grasp several concepts. In information retrieval, we are frequently concerned with "what proportion of the recovered information is of interest to the user" and "how much of the material of interest to the user gets retrieved," two questions Precision and Recall can address. Precision is the proportion of true cases among all positive predictions or the accuracy of the prediction. Recall is the proportion of accurate predictions among all positive predictions or the coverage of correct predictions. Table 4.12 provides a complete analysis of the ideas of true and false positive cases and true and false negative instances.

Table 4.2: True positive examples(TP), Fulse positive examples(FP), True negative examples(TN) and Fulse negative examples(FN).

		Reality	
		Positive	Negative
Model prediction	Positive	TP	FP
	Negative	FN	TN

Based on these concepts, we may express Precision (P) and Recall (R) as

$$P = \frac{TP}{TP + FP}, \tag{4.2}$$

$$R = \frac{TP}{TP + FN}, \tag{4.3}$$

while accuracy is the ratio of actual positive and negative cases to the whole sample and is stated as

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}. \tag{4.4}$$

Calculating the Precision and Recall for each prediction yields the P-R curve. Ideally, we would like a model to have both high precision and a high Recall, but as shown in Figure 4.2, there is a fluctuation between the two mutually supporting. Therefore, optimising the detection capability of a model is fundamentally a process of the tradeoff between Precision and Recall. Based on the Precision-Recall curve, a numerical metric may be derived by computing the mean of the Precision values corresponding to each recall value: AP (Average Precision).

$$AP = \sum_{i=1}^{n-1} \left(r_{i+1} - r_i\right) p_{\text{interp}}\ \left(r_{i+1}\right), \tag{4.5}$$

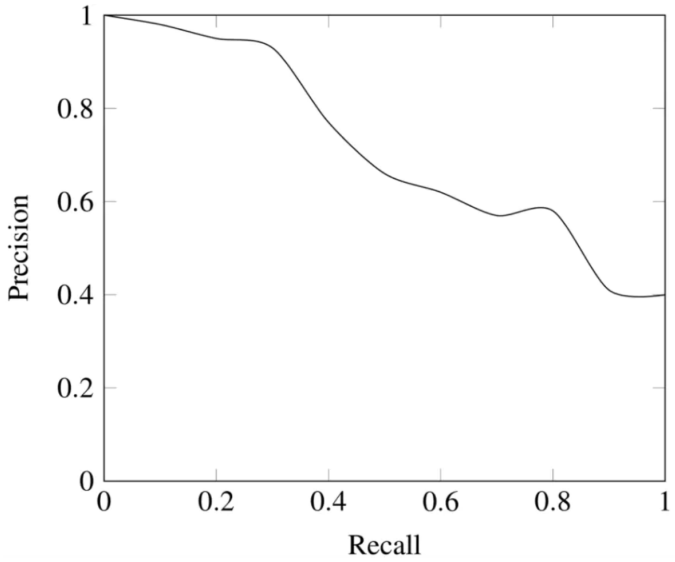


Figure 4.2: A typical P-R Curve.

AP measures how well the trained model finds the target category. AP also indicates how well the trained model can identify the category of interest.

Once the AP has been determined, the computation of the mAP is straightforward. The AP is computed for all categories, and then the average is used to determine the mAP. The mAP evaluates the accuracy with which the trained model detects each category. If a $K-$category query is assumed, the formula for calculating mAP is

$$mAP = \frac{\sum_{i=1}^{K} AP_i}{K}. \tag{4.6}$$

4.1.3.2 Extract time per query image

For a user-centric search system, the effectiveness is equivalent to speed. In other words, the primary concern is the time required for one search to produce results when a user enters an image. As explained in Section 3.4, our retrieval system consists of offline database image feature extraction and online query image feature extraction. Data processing and feature extraction in the Database and index building are offline. The online component of Query image feature extraction has the greatest effect on the effectiveness and speed of this retrieval system. This is why Extract time per query image is utilised as this paper's evaluation parameter for efficiency.

We will now introduce the measure of extract time per query picture (T_e). The test dataset is supplied to our feature extraction model, which extracts and stores the queries from the test set. Assuming that the test dataset contains M database images and N query images, the database images and query images will be extracted independently for the test. We record the time T at which all N queries are extracted, then

$$T_e = \frac{T}{N}. \tag{4.7}$$

The extracted database feature vector and the query feature vector are independently stored, indexed, and compared to determine the final retrieval result. It is essential to notice that the matching time is orders of magnitude less than the feature extraction time, emphasising the significance and applicability of extract time per query image as a parameter for validating retrieval speed.

4.1.3.3 Multi scale Evaluation

This thesis introduces the multi-scale assessment mechanism for testing the model. In a single-scale evaluation, the image dimensions given to the CNN are limited to 1024 × 1024 pixels. Figure 4.3 depicts how, in multi-scale evaluation, the input images are scaled to different dimensions and then fed into the network collectively. The global descriptors from many scales are finally merged into one global descriptor. The learnt whitening procedure operates on this unified multi-scale global descriptor. If not specified otherwise, single-scale evaluation is applied in our investigations.

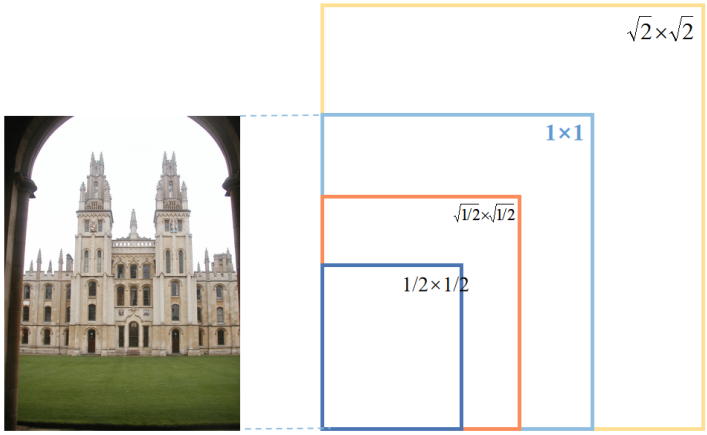


Figure 4.3: Multi scale division.

4.2 Comparison Validation of Configuration

As discussed in Chapter 3, our model pipeline for feature extraction consists of a base CNN model, a pooling layer for feature enhancement, and a whitening procedure for dimensionality reduction. Each of these three categories contains an abundance of implementation ideas and methodologies. Which methodology will be utilized in each module, and why did this thesis make these decisions? I will address these concerns in this section through a series of comparative validation experiments.

4.2.1 Comparison Validation of Backbone Network

Network architecture is the cornerstone of CNN models. We use three alternatives fine-tuned CNN architectures as the backbone in this paper: fine-tune AlexNet (Alex),

fine-tuned ResNet (Res), and fine-tuned VGGNet (VGG). In light of their prevalence in the current image matching and retrieval field, these network architectures were chosen for comparison in this section. We develop models based on each using a uniform pooling strategy (GeM) to assess their performance on the testing datasets.

4.2.1.1 Diverse Networks Experiments

Before presenting the exact experimental parameter settings, the topologies of the three underlying networks are briefly described.

Fine tuned AlexNet

The AlexNet [4] model comprises five convolutional layers and three fully connected layers, as depicted in Figure 4.4. AlexNet introduces ingeniously Local Response Normalization (LRN), which simulates a neurobiological function known as lateral inhibition to achieve local inhibition. LRN makes the response's greater values proportionally larger, enhancing the model's capacity for generalization. In the final fully connected layer, drop-out is introduced. During the deep learning network training, drop-out indicates that the neural network units are momentarily removed from the network (while preserving their weights) and no longer respond to the forward and reverse sent input. It effectively prevents the model from overfitting while accelerating computation due to the reduced network complexity.

In this thesis, we fine-tune the training of AlextNet using the Sfm120k [21] dataset

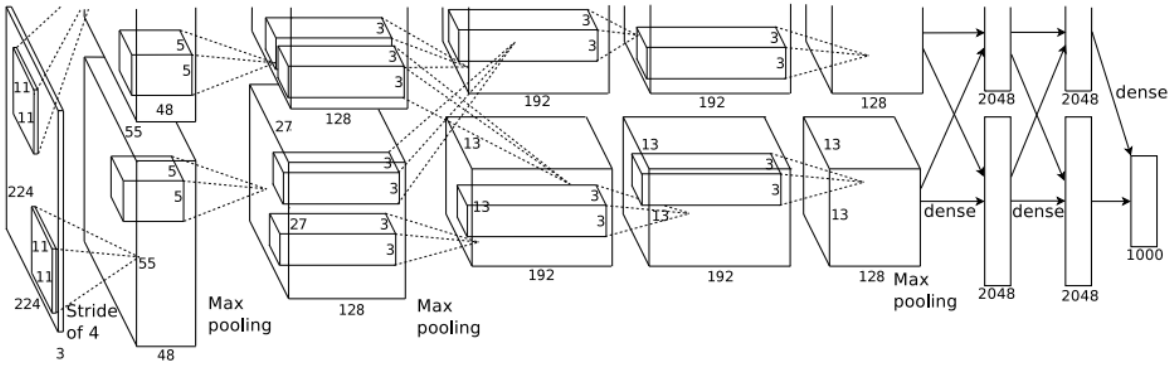


Figure 4.4: AlexNet Structure. [4]

Fine-tuned VGGNet

Using multiple consecutive smaller convolutional kernels instead of more giant convolutional kernels is an improvement of VGG16 [9] over AlexNet [4]. In particular, VGG employs three 3x3 convolutional kernels in place of 7x7, and two 3x3 convolutional kernels in place of 5*5 convolutional kernels. This substitution primarily intends to increase the network's depth while maintaining the same perceptual field. Consequently, more sophisticated learning is assured at a lower cost. VGG16 and VGG19 are the two typically utilised hierarchies for VGGNet. In the experiments comparing AlexNet and VGGNet, this thesis use VGG16 with 16 layers of depth. The model hierarchy of VGGNet is illustrated in Figure 4.5.

In this thesis, we fine-tune the training of VGGNet using the Sfm120k [21] dataset to obtain fine-tuned VGGNet (VGG).

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 4.5: VGGNet Structure. [5]

Fine-tuned ResNet

ResNet [5] is based on the VGG19 network, with modifications and the addition of residual units via a short-circuiting mechanism. The network architecture of the 34-layer ResNet is depicted in Figure 4.6. An essential ResNet design idea is that the number of feature maps is doubled when the feature map size is halved, preserving the network layers' complexity. ResNet adds a short-circuiting mechanism between two layers, resulting in residual learning, as shown in Figure 4.6.

We further analyse the design of residual units in ResNet, which uses two types of residual units, as shown in Figure 4.7. When the input and output dimensions are the same, the input can be directly added to the output for short-circuit connections. There are two strategies when the dimensions do not match: (1) perform a downsample and then use zero-padding to expand the dimensions so that the parameters remain the same; (2) employ a new mapping (shortcut projection), which increases the parameters as well as the computational cost.

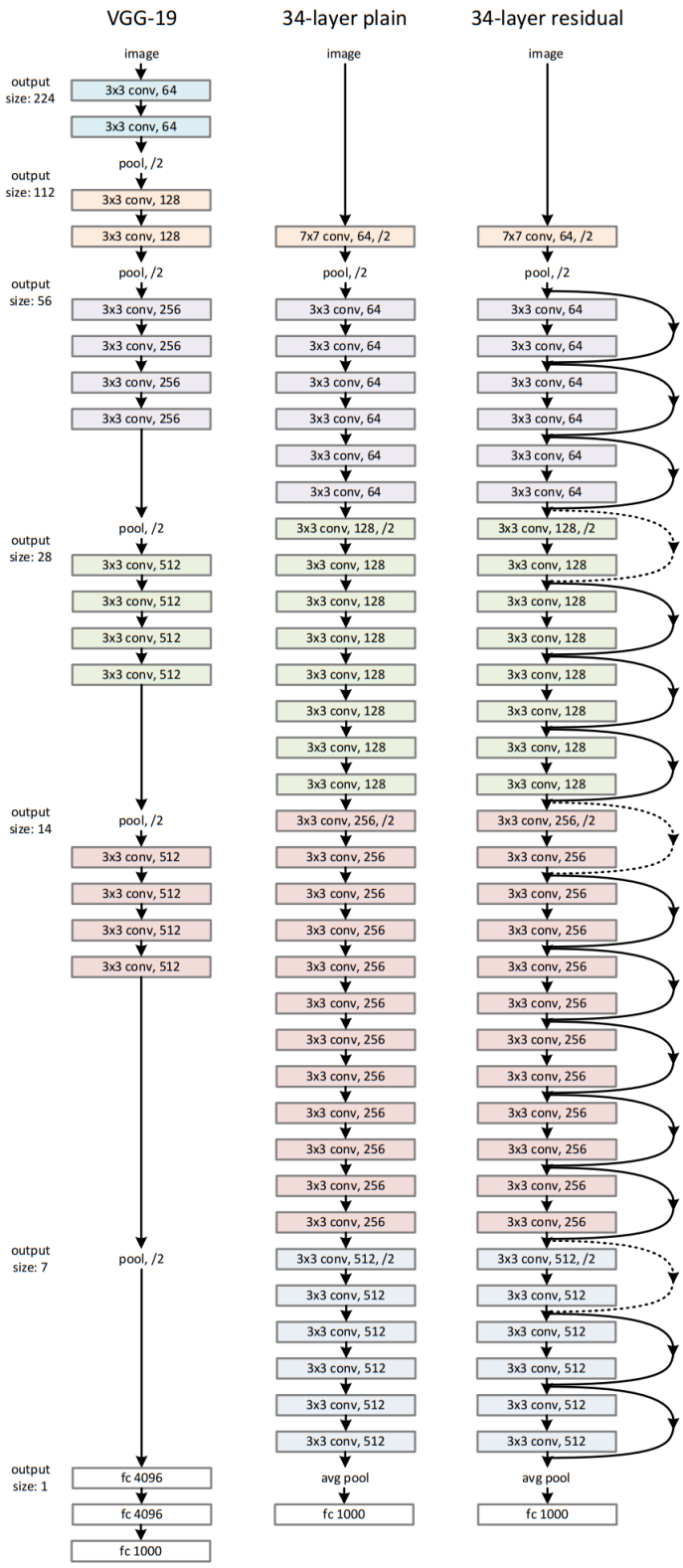


Figure 4.6: ResNet34 Structure. [5]

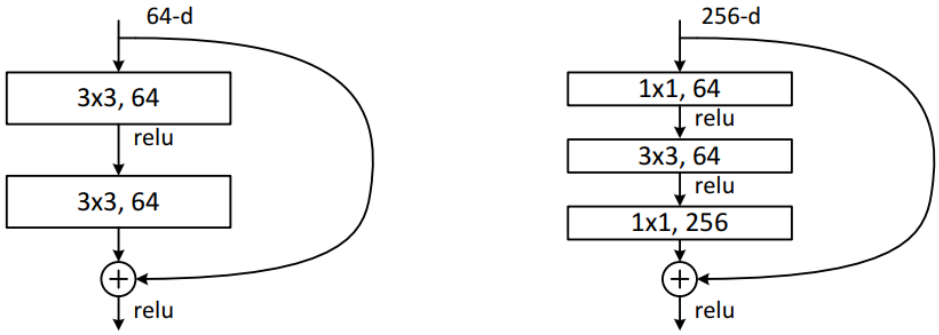


Figure 4.7: Two types of residual units for ResNet. The left shows the shallow network residual unit and the right shows the deep network residual unit.[5]

ResNet supports a range of standard depths, and in comparative studies with AlexNet and VGGNet, we employ ResNet101 with 101 depth layers. In this thesis, we fine-tune the training of 101 layers ResNet using the Sfm120k [21] dataset to obtain fine-tuned ResNet (Res).

As indicated in Section 4.3, we trained the models with Alex, VGG, and Res as backbones, respectively. In order to streamline the training process and increase training speed and efficiency, post-whitening was omitted from this comparison experiment. Similarly, to increase the experiment's efficiency, the feature output dimension was reduced to 512 to retrieve the findings more quickly. The lower dimension features may be slightly less representative than the higher dimensional features. However, the goal of this experiment is to examine the most appropriate base model structure as opposed to the final model configuration. Thus this lack of precision should be disregarded. The default settings of the general model settings were utilised for all other hyperparameters, and Table 4.3 details the individual experimental settings.

Table 4.3: Elemental set-up for comparative validation experiments of different CNN network models.

Experimental elements	**Setting**
Backbone	Alex/Res/VGG
Pooling Method	GeM
Whitening Method	None
Output dim	512
Loss function	Contrastive
Loss Margin	0.85
Optimizer	Adam
Learning rate	1e-6

We evaluated Alex, VGG and Res on four test datasets, Oxford5k, Paris6k, ROxford5k, and RParis6k, and achieved the accuracy and speed results presented in Tables 4.4 and 4.5.

Table 4.4: The mAP performance comparison of methods for different CNN backbones. Evaluation is performed on Oxford5k Paris6k, ROxford5k, and RParis6k datasets.

Method	mAP					
	Oxford	*Paris*	*ROxford*		*RParis*	
			M	H	M	H
Alex	61.7	71.5	43.3	17.1	58	29.7
VGG	80.3	83.8	**56.2**	**28.9**	69.2	46.9
Res	**85.2**	**88.8**	55.8	28.1	**69.7**	**47.0**

Table 4.5: The speed performance comparison of methods for different CNN backbones. Evaluation is performed on Oxford5k Paris6k, ROxford5k, and RParis6k datasets.

Method	Extract time per query image (T_e/s)			
	Oxford	*Paris*	*ROxford*	*RParis*
Alex	0.0423	0.0399	0.0465	0.456
VGG	0.0526	0.0515	0.0526	0.513
Res	0.0537	0.0535	0.0537	0.535

From the data above, it is evident that both ResNet101CNN and VGG16CNN demonstrate a significant increase in accuracy compared to AlexCNN, with this gain reaching approximately 20% on Oxford5k and Paris6k. On the Revisited Oxford-5k dataset, the VGG16CNN outperformed ResNet101, while on all other datasets, it was the ResNet101CNN that outperformed. Comparing the feature extraction times of ResNet101CNN and VGG16CNN on the Oxford5k and Paris6k datasets reveals a 0.001-second improvement for VGG16CNN. Overall, it was determined that ResNet101 performed the best in this trial.

4.2.1.2 Layer Depth Experiment

Having established ResNet as the backbone architecture, we want to investigate the effect of different layer depths on its performance. Therefore further comparison experiments were implemented for 50, 101 and 150 layers of ResNet. The experimental parameters were set as Table 4.6.

The results in terms of accuracy and speed are shown in Tables 4.7 and 4.8, where we conducted comparative tests on four test datasets, Oxford5k, Paris6k, ROxford5k and RParis6k.

In image retrieval, we ideally prefer to attain both high precision and excellent efficiency. However, it is evident from Tables 4.7 that a deeper hierarchy yields more

accurate retrieval results at the expense of increased time and decreased efficiency. Therefore, an effective retrieval system aims to strike a trade-off between precision and speed. Res101 obtains an mAP of 85.2 on Oxford5k and 88.8 on Paris6k in 0.0537s and 0.0535s, respectively. The combined performance of Res101 in terms of accuracy and speed led us to choose it as the model backbone.

Table 4.6: Elemental set-up for comparative validation experiments of different ResNet Layers.

Experimental elements	**Setting**
Backbone	Res50/Res101/Res150
Pooling Method	GeM
Whitening Method	None
Output dim	512
Loss function	Contrastive
Loss Margin	0.85
Optimizer	Adam
Learning rate	1e-6

Table 4.7: The mAP performance comparison of methods for different layer depths. Evaluation is performed on Oxford5k Paris6k, ROxford5k, and RParis6k datasets.

Method	mAP					
	Oxford	*Paris*	*ROxford*		*RParis*	
			M	H	M	H
Res50	82.3	85.9	52.7	25.4	66.6	45.4
Res101	85.2	88.8	55.8	28.1	69.7	47.0
Res150	88.8	89.2	57.1	29.5	70.1	48.9

Table 4.8: The speed performance comparison of methods for different layer depths. Evaluation is performed on Oxford5k Paris6k, ROxford5k, and RParis6k datasets.

Method	Extract time per query image (T_e/s)			
	Oxford	*Paris*	*ROxford*	*RParis*
Res50	0.0498	0.0478	0.0499	0.0479
Res101	0.0537	0.0535	0.0537	0.0535
Res150	0.0634	0.0613	0.0648	0.0613

Table 4.9: Elemental set-up for comparative validation experiments of different pooling methods.

Experimental elements	**Setting**
Backbone	Res101
Pooling Method	SPoc/RMac/GeM
Whitening Method	None
Output dim	512
Loss function	Contrastive
Loss Margin	0.85
Optimizer	Adam
Learning rate	1e-6

4.2.2 Comparison Validation of Pooling Methods

As noted in Section 3.2, this thesis employs various pooling strategies to increase the identifiability of features by aggregating them. In this subsection, control variable experiments are performed on the various pooling strategies to compare their performance.

4.2.2.1 Diverse Pooling Methods

We train the model in accordance with Section 4.3. On Resnet101, which did well in the network structure comparison tests, the three pooling methods GeM [21], SPoC [28], and RMac [29] were trained. In order to streamline the training process and increase training speed and efficiency, post-whitening was omitted from this comparison experiment. This design also provides a more objective evaluation of the effect of the pooling approach on the training effect and removes distracting elements. In order to save training time, the feature output dimension was set to 512 in this experiment, as same as the prior experiments. The default settings of the general model settings were utilised for all other hyperparameters, and Table 4.9 displays the specific experimental settings.

We named the models based on the three different pooling methods as Res-GeM, Res-SPoc and Res-RMac and tested them on four test datasets, Oxford-5k, Paris-6k, ROxford-5k and RParis-6k, and obtained the results in terms of precision and speed as shown in Tables 4.10 and 4.11.

Res-GeM performed the best in this experiment when both accuracy and speed are considered combined.

4.2.2.2 Pooling Parameter Experiment

Having established GeM as the model's pooling method, the next step's objective was to investigate the pooling parameter p on the model's properties; thus, additional compar-

Table 4.10: The mAP performance comparison of methods for different pooling methods. Evaluation is performed on Oxford-5k, Paris-6k, ROxford-5k, and RParis-6k datasets.

Method	mAP					
	Oxford	*Paris*	*ROxford*		*RParis*	
			M	H	M	H
Res-SPoc	82.9	85.9	54.7	26.3	68.4	43.9
Res-RMac	84.8	87.8	**56.5**	28.0	**70.1**	**48.1**
Res-GeM	**85.2**	**88.8**	55.8	**28.1**	69.7	47.0

Table 4.11: The speed performance comparison of methods for different pooling methods. Evaluation is performed on Oxford-5k, Paris-6k, ROxford-5k, and RParis-6k datasets.

Method	Extract time per query image (T_e/s)			
	Oxford	*Paris*	*ROxford*	*RParis*
Res-SPoc	0.0591	0.0542	0.0593	0.0547
Res-RMac	0.0568	0.0566	0.0536	0.0565
Res-GeM	0.0537	0.0535	0.0537	0.0535

ison experiments were further conducted with other pooling parameters. The following parameters were set for the experiment:

Table 4.12: Elemental set-up for comparative validation experiments of different pooling parameters.

Experimental elements	**Setting**
Backbone	Res101
Pooling Method	GeM Pooling
Whitening Method	None
Output dim	512
Loss function	Contrastive
Loss Margin	0.85
Optimizer	Adam
Learning rate	1e-6

Tables 4.13 illustrates the comparison tests performed on four test sets: Oxford5k, Paris6k, ROxford5k, and ROxford6k.

On the results of Oxford-5k and Paris-6k datasets, we observed that the pooling parameter p produced consistently good results around the starting value $p = 3$, with mAPs of 67.9 and 74.8, respectively. Meanwhile, a shared fixed parameter and a

shared learning parameter performed similarly, with the latter slightly outperforming the former. Thus for the remainder of the experiments, we used the shared learning parameter p and manually set the initial value at 3.

Table 4.13: The mAP performance comparison of methods for different pooling parameters. Evaluation is performed on Oxford-5k, Paris-6k, ROxford-5k, and RParis-6k datasets.

Method		mAP					
Initial p	Learned p	Oxford	Paris	ROxford		RParis	
				M	H	M	H
3	-	**67.9**	**74.8**	45.8	18.5	69.7	47
3	2.35	67.7	73.5	**46.1**	**19.1**	**69.9**	**48**
2,5	-	66.8	74.1	45.3	18.1	69.0	46
1,10	-	65.6	72.2	45.1	18.1	69.2	46.5
1,10	1.5-8.7	65.3	71.4	45.1	17.8	69.0	46.8

4.2.3 Comparison Validation of Whitening Methods

This thesis compares three whitening techniques: standard PCA whitening [20], discriminative learnt whitening [21], and end-to-end whitening [30]. The theoretical principles for these strategies is outlined in Section 3.3, and more experimental details are provided in this section.

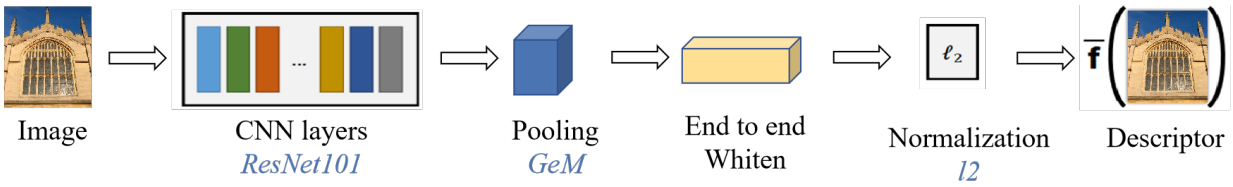


Figure 4.8: The process of learned whitening implementation.

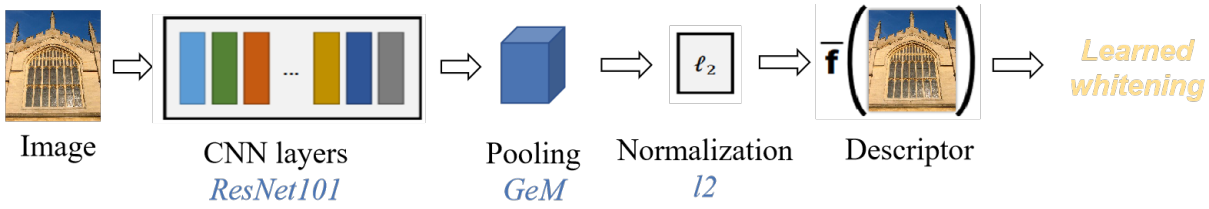


Figure 4.9: The process of end-to-end whitening implementation.

Whitening is often learned unsupervised using generative models using PCA on independent datasets. In the experiments, discriminative whitening was learned via the Sfm30k dataset, which employed the identical 3D training data gathering approach

as the training set. Figure 4.8 illustrates that discriminative learned whitening was conducted as a post-processing step after the global descriptors were extracted.

As depicted in Figure 4.9, end-to-end whitening has a different training pattern than discriminative learned whitening. This approach whitens using a fully connected layer that can be learned (FC Layer). The FC Layer is connected after the Global Pooling Layer, and its initialization is the outcome of supervised whitening learning with training data and predefined features.

In order to highlight the dimensionality reduction of the high-dimensional global features by the whitening process, the feature output dimension in this experiment was set to 2048, and two distinct pooling methods were used to lessen the effect of the pooling layer structure. The default settings of the general model were utilized for all other hyperparameters, and Table 4.14 details the individual experimental settings.

Table 4.14: Elemental set-up for comparative validation experiments of different whitening methods.

Experimental elements	**Setting**
Backbone	Res101
Pooling Method	GeM Pooling
Whitening Method	PCA/Learned whitening/End-to-end whitening
Output dim	2048
Loss function	Triplet
Loss Margin	0.85
Optimizer	Adam
Learning rate	1e-6

We named the models based on the three different whitening methods GeM-PCA, GeM-lw and GeM-e2e, with GeM-0 being the control group without any whitening treatment. Comparative tests were performed on four test sets, Oxford-5k, Paris-6k, ROxford-5k and ROxford-6k, and the results in terms of accuracy and speed were obtained as shown in Tables 4.15 and 4.16.

Table 4.15: The mAP performance comparison of methods for different whitening methods. Evaluation is performed on Oxford-5k, Paris-6k, ROxford-5k, and RParis-6k datasets.

Method	mAP					
	Oxford	*Paris*	*ROxford*		*RParis*	
			M	H	M	H
GeM-0	66.0	76.1	45.8	18.5	69.7	47
GeM-PCA	58.1	60.2	42.9	16.3	59.9	42.4
GeM-lw	**82.1**	**89.7**	62.78	**37.5**	**77.0**	55.3
GeM-e2e	**83.1**	**89.9**	**63.4**	**37.6**	**77.0**	**56.2**

Table 4.16: The speed performance comparison of methods for different whitening methods. Evaluation is performed on Oxford-5k, Paris-6k, ROxford-5k, and RParis-6k datasets.

Method	Extract time per query image (T_e/s)			
	Oxford	*Paris*	*ROxford*	*RParis*
GeM-0	0.1034	0.0791	0.1029	0.0793
GeM-PCA	0.0784	0.0781	0.0791	0.0775
GeM-lw	0.0550	0.0534	0.0550	0.0533
GeM-e2e	0.0638	0.0621	0.0634	0.0632

Table 4.17: The pros and cons of different whitening methods. Minus signs indicate weakening and plus signs indicate enhancement.

Methods	mAP	Speed
GeM	baseline	baseline
PCA whitening	−	+
Discriminative learning whitening	+	=/+
End-to-end whitening	++	=

Observing the experimental data, it is evident that adding the whitening step considerably improves Gem's accuracy. Table 4.17 summarises the pros and cons of the various whitening techniques regarding precision and speed.

Both discriminative learning whitening and end-to-end whitening performed unquestionably better than the situation without whitening. However, local PCA whitening diminished the model's precision and speed while decreasing its dimensionality. On the Oxford-5k and Paris-6k datasets, global end-to-end whitening enhances mAP more significantly, achieving mAP 83.1 and 89.9. Notably, despite obtaining comparable performance, discriminative learning whitening requires additional post-processing of descriptors at test time, whereas end-to-end integrated models are more convenient to test. In light of these considerations, the decision was made to implement an end-to-end whitening strategy.

4.3 Training Implementation of Baseline Model

After identifying the precise methods that will be implemented for each module of the model, the specific training implementation process of the baseline model is presented. First, the entire training procedure is described, followed by the selection and mining of training pairs and the formulation of the loss function. After numerous trials, the ideal hyperparameter parameters for the model are reported.

4.3.1 Implementation Process

The Baseline model consists of a CNN model, a pooling layer for enhancing features, and a whitening procedure for dimensionality reduction. Each training tuple has one original image, M positive and N negative images. First, a training image is preprocessed and then put into the Res101 network, where the feature maps are retrieved using convolution. The feature maps are further enhanced by a GeM [21] pooling layer. $D \times H \times W$ feature maps are pooled into a single $D \times 1$ global descriptor and processed by a whitening layer to generate the final descriptor. Each training session has a fixed amount of training tuples, and the descriptors in a training tuple are computed, learned via the loss function, and then fed back to the model during the training round. Figure 4.10 depicts the experiment's training implementation.

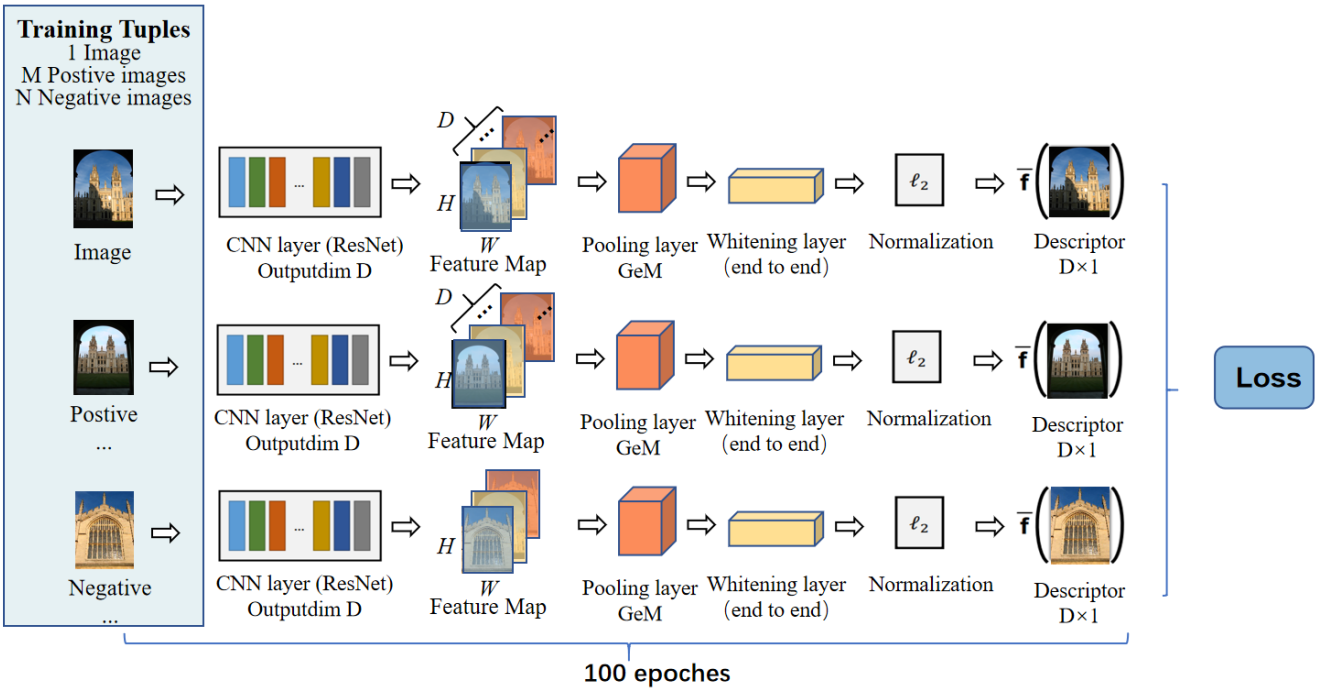


Figure 4.10: The process of baseline model training implementation.

4.3.1.1 Selection of Training Pairs

In our experiments, 551 building models were chosen for training, whereas 162 building models were chosen for validation. All of these training data are provided to the model as training tuples. How are the training tuples chosen and concatenated in the massive training dataset? The selection of training pairs is dependent on the loss function's design. Contrastive and triplet are pairwise loss functions. Therefore they require a pair of positive and negative examples. Each training and validation tuple consists of one query image, M positive images, and N negative images, where M and N are set as 1 and 5, respectively. For each epoch round, the tuples' positive and negative images are reselected randomly from the dataset. Figures 4.11 depict the components of a

training tuple.

In the case of AP Loss [40], it is a type of ranking loss and does not rely on the distance between positive and negative training pairs for learning and iteration. Instead of generating a training tuple, thousands of images are analysed simultaneously during each training session, and the AP scores between them are directly learned.

Figure 4.11: Composition of a typical training tuple.

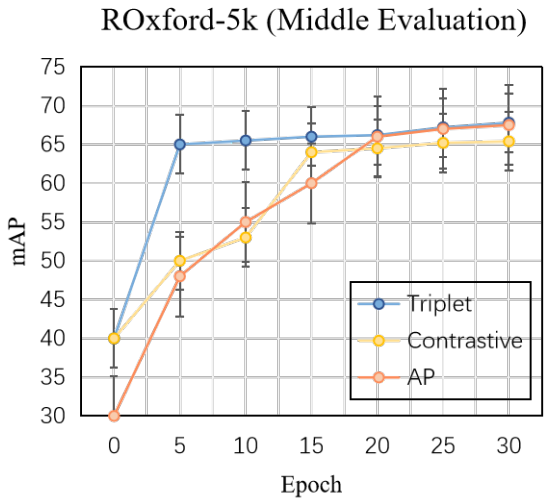
4.3.1.2 Loss Performances

Different loss functions employ distinct strategies for learning and dealing with descriptions. This thesis tries pairwise loss represented by contrastive and triplet as well as ranking loss represented by AP loss, respectively. We select ROxford-5k and RParis-6k as test cases to demonstrate their performance during training.

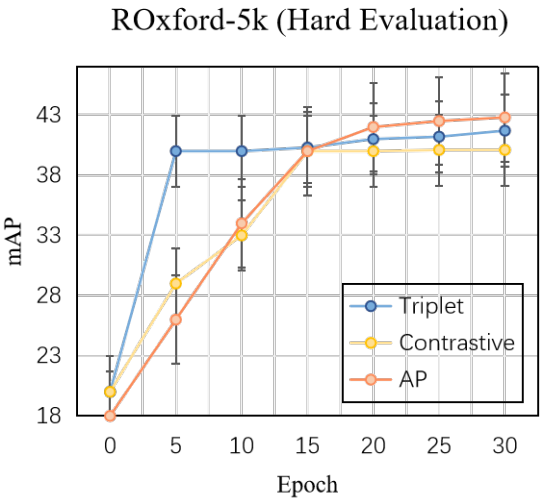
In this experiment, the feature output dimension was set to 2048, and all other hyperparameters were set using the default settings of the general model, as shown in Table 4.18. Using ROxford-5k and RParis-6k as the test set, the variation curves of mAP with the number of training rounds of the model using different loss functions are plotted, as shown in Figure 4.12.

Table 4.18: Elemental set-up for different loss functions in baseline model.

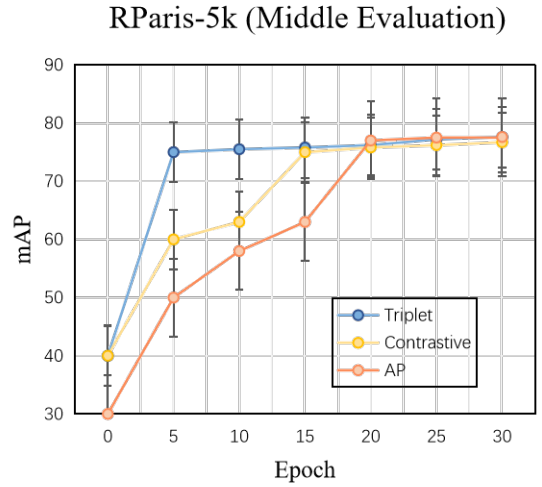
Experimental elements	**Setting**
Backbone	Res101
Pooling Method	GeM Pooling
Whitening Method	End-to-end whitening
Output dim	2048
Loss function	Contrastive/Triplet/AP
Loss Margin	0.85
Optimizer	Adam
Learning rate	1e-6



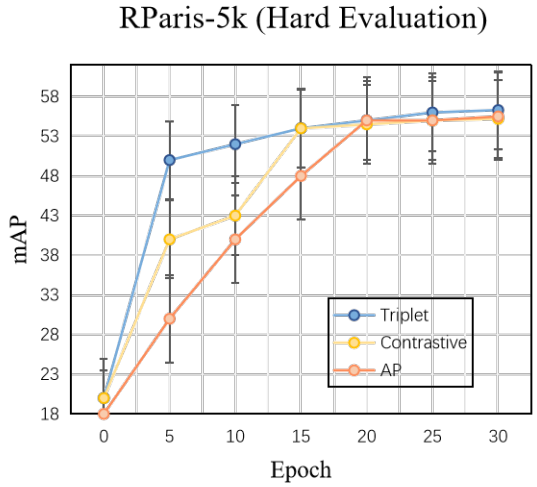
(a) Performance on ROxford-5k (Middle Evaluation)



(b) Performance on ROxford-5k (Hard Evaluation)



(c) Performance on RParis-6k (Middle Evaluation)



(d) Performance on RParis-6k (Hard Evaluation)

Figure 4.12: Performance comparison of different loss functions. The evaluation was performed with ResNet101 GeM on the ROxford-5k and RParis-6k datasets in middle and hard modesrespectively shown as Figures a-d). These figures show the variation of mAP with the number of training epochs.

Figure 4.12 demonstrates that Triplet loss has a faster convergence rate and already exhibits consistent accuracy between the $5th$ and $10th$ epoch. Contrarily, the contrastive loss is inferior to the triplet loss in the final mAP achieved, beginning to settle around the 15th epoch. AP Loss performed comparably to the previous two in terms of mAP, was somewhat superior to triplet loss, and was slower than the previous two in network convergence. This weakness may be owing to the training process, which involves inputting hundreds of training photos at each epoch, resulting in a slower learning rate and an iterative effect. In light of the preceding performance, the final model in this thesis employs triplet loss for model learning.

4.3.2 Hyperparameters

In machine learning, hyperparameter optimization seeks to identify the hyperparameters that allow machine learning algorithms to perform optimally on validation datasets. In contrast to general model parameters, hyperparameters are predetermined prior to training. This model's hyperparameters consist of the learning rate, query size, batch size, etc. Hyperparameters optimization is a combinatorial optimization problem that cannot be solved by gradient descent. In addition, the time cost of evaluating a set of hyperparameter configurations is very high, making it challenging to use some optimization techniques (e.g. evolutionary algorithms) for hyperparameter optimization. Each hyperparameter tweak must be retrained to evaluate its effect, which is inefficient for large models such as the one used in this thesis. Consequently, grid search and random search are frequently utilized for hyperparameter settings.

Grid search is a technique that identifies an acceptable set of hyperparameter configurations by attempting all possible hyperparameter combinations. Suppose there are k total hyperparameters, and the kth hyperparameter can take on m_k different values. Consequently, there are $k \times m_k$ configurations in total. If the hyperparameters are continuous, they can be discretized by selecting a few "empirical values," such as setting the learning rate to $1e-6$. We cannot discretize continuous hyperparameters at equal intervals but must instead discretize them according to their unique features. The grid search trains a model based on several combinations of these hyperparameters, then tests the performance of these models on the development set and selects the configuration with the greatest performance.

Table 4.19: Hyperparameter configuration of the optimal baseline model trained on the Sfm120k dataset.

Baseline Model(*Sfm120-Res*)		
Superparameters	**Description**	**Value**
p	Initial value of pooling parameter	3
loss margin	Decision margins for triplet loss functions	0.5
lr	Learning rate of the modle	5e-7
neg-num	Number of negative images per training tuple	5
query size	Number of randomly selected queries per epoch	2000
pool size	Size of the pool for hard negative examples mining	20000
batch size	Number of training tuples in a mini-batch	5
image size	Artificially specified size of training images	1024
epoch	Total number of training epochs	100

Table 4.20: Hyperparameter configuration of the optimal baseline model trained on the Google Landmarks v2 dataset.

Baseline Model(*GL-Res*)		
Superparameters	**Description**	**Value**
p	Initial value of pooling parameter	3
loss margin	Decision margins for triplet loss functions	0.75
lr	Learning rate of the modle	5e-7
neg-num	Number of negative images per training tuple	5
query size	Number of randomly selected queries per epoch	2000
pool size	Size of the pool for hard negative examples mining	20000
batch size	Number of training tuples in a mini-batch	8
image size	Artificially specified size of training images	1024
epoch	Total number of training epochs	100

However, there are a huge number of hyperparameters for the model provided in this thesis, and the impact of different hyperparameters on the model's performance varies significantly. Some hyperparameters (such as query size) have a modest effect on model performance, whilst others (such as learning rate) have a relatively substantial effect. In this instance, employing grid search would be a futile attempt on unimportant hyperparameters that would consume too much time. For hyperparameter optimization, this thesis uses a stochastic search approach. In stochastic search, a random combination of hyperparameters is generated, and the configuration with the highest performance is chosen.

Two optimization models were developed for this thesis using Sfm120k and Google-Landmark v2 as training sets for optimization training. After multiple hyperparameter optimization trials, Tables 4.19 and 4.20 indicate the ideal hyperparameter combinations for the two models. The accuracy of the two optimization models at various sizes is displayed in Table 4.21 and 4.22.

Table 4.21: The mAP performance of *Sfm120-Res* Model with different scales. Evaluation is performed on Oxford-5k, Paris-6k, ROxford-5k, and RParis-6k datasets.

Scales	mAP of *Sfm120-Res*					
	Oxford	*Paris*	*ROxford*		*RParis*	
			M	H	M	H
[1]	86.2	90.6	66.9	76.6	39.8	55.2
$[1, \sqrt{1/2}, 1/2]$	87.8	92.7	67.8	77.6	41.7	56.2

Table 4.22: The mAP performance of *GL-Res* Model with different scales. Evaluation is performed on Oxford-5k, Paris-6k, ROxford-5k, and RParis-6k datasets.

Scales	mAP of *GL-Res*					
	Oxford	*Paris*	*ROxford*		*RParis*	
			M	H	M	H
[1]	87.5	91.6	64.6	78.0	40.9	57.5
$[1, \sqrt{2}, \sqrt{1/2}]$	89.9	93.7	67.3	80.6	44.3	61.5

4.4 Training Implementation of Fine-tuned SOI Model

Following the training in Section 4.3, we have obtained a Baseline model with excellent performance. However, we are not stopped with this, and the additional application of the SOI module on top of the baseline model provides us with a further optimised model with improved performance. This section discusses the training procedure and specifics of the SOI model's optimisation.

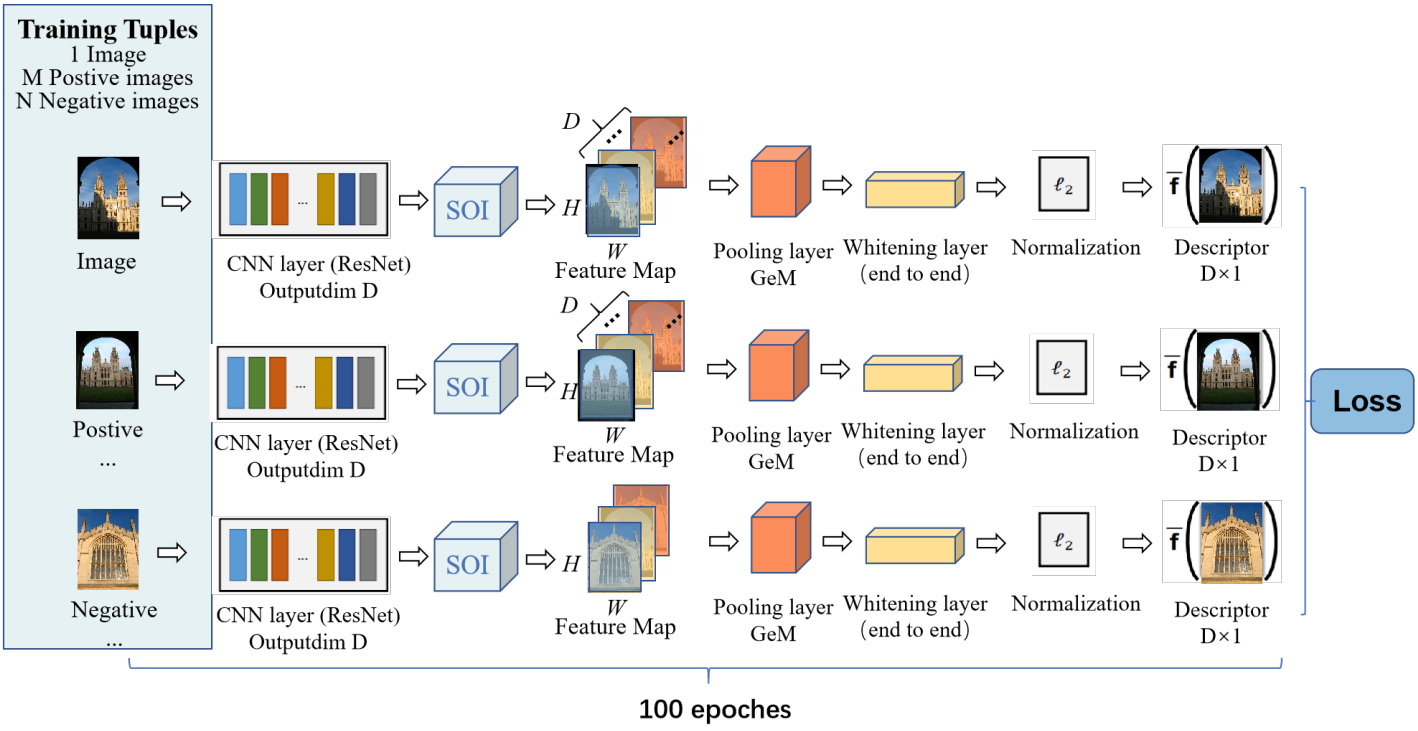


Figure 4.13: The process of SOI model training implementation.

4.4.1 Implementation Process

The optimised SOI model comprises a CNN model, an SOI module for feature map optimisation, a pooling layer for feature aggregation, and a whitening procedure for dimensionality reduction. Images are also processed in groups of tuples during the

training phase, with each training tuple including one original image, M positive images, and N negative images. First, a training image is preprocessed and then put into the ResNet101 network, where a feature map is produced using convolution. The feature map is processed through the GeM [21] pooling and whitening layers to acquire the final descriptors. Afterwards, it is transferred through the SOI module to generate a new feature map with increased local spatial information utilisation. In our experiments, 100 such epochs are used to train a model. Each training session includes a predetermined number of training tuples selected and configured in the same manner as the baseline model. Figure 4.13 depicts the experiment's training implementation.

Similarly, this thesis conducts experiments comparing the performance of different loss functions for the fine-tuned SOI model. Pairwise loss, represented by contrastive and triplet, and an additional second-order similarity loss function (SOS) are applied. Their performance during training is shown as examples in *ROxford5k* and *RParis6k* datasets.

In this experiment, the feature output dimension was set to 2048, and all other hyperparameters were set using the default settings as general, as shown in Table 4.23. Using *ROxford5k* and *RParis6k* as the test set, the variation curves of mAP with the number of training rounds of the model using different loss functions are plotted, as shown in Figure 4.14.

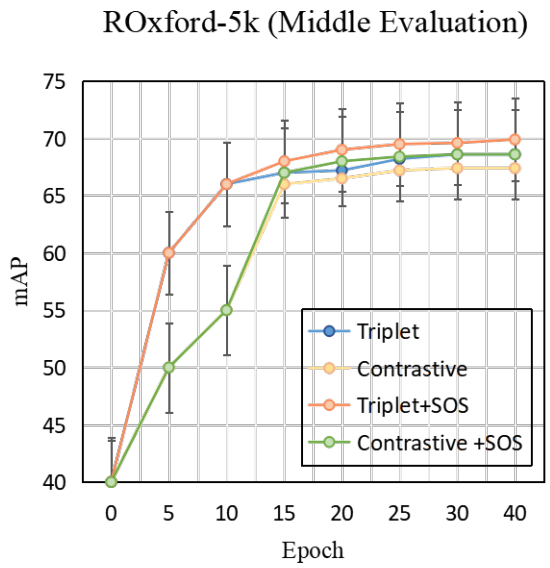
Table 4.23: Elemental set-up for different loss functions in fine-tuned model.

Experimental elements	**Setting**
Backbone	Res101
Pooling Method	GeM Pooling
Whitening Method	End-to-end whitening
Output dim	2048
Loss function	Contrastive/Triplet/AP
Loss Margin	0.85
Optimizer	Adam
Learning rate	1e-6

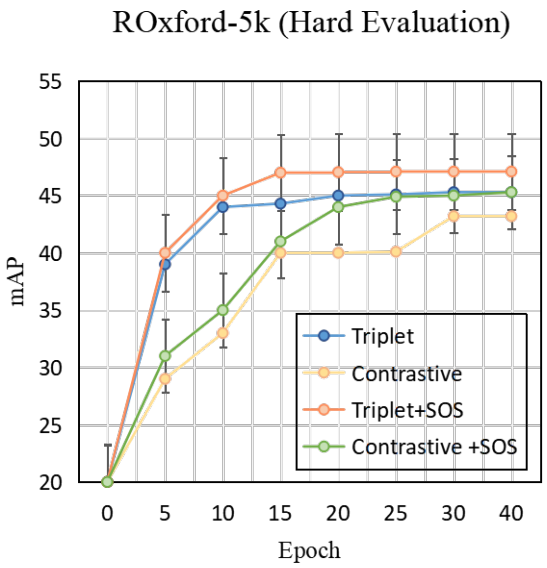
Figure 4.14 demonstrates that the single Triplet loss has a faster convergence rate and is already accurate at the 10th epoch. On the other hand, the single Contrastive Loss is marginally inferior to the Triplet loss and begins to settle around the 15th epoch. And after combining with SOS loss, both converge with a minor delay, but both demonstrate evident improved precision (mAP). In this thesis, the final model employed $triplet + SOS$ loss for fine-tuned model learning.

4.4.2 Hyperparameters

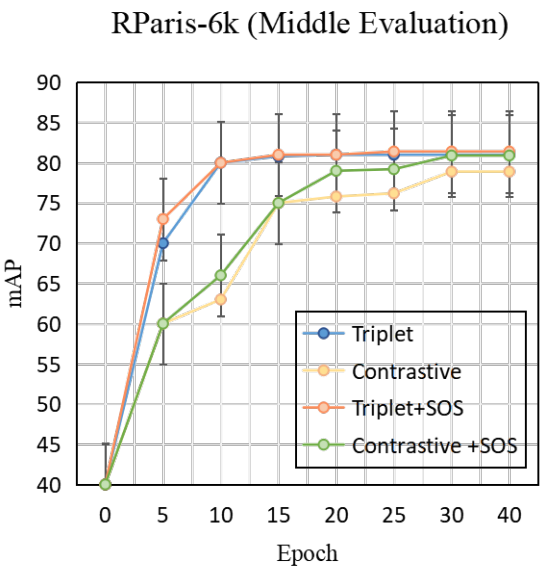
This part also employs a random search technique for optimizing hyperparameters. In random search, hyperparameters are generated randomly, and the configuration with



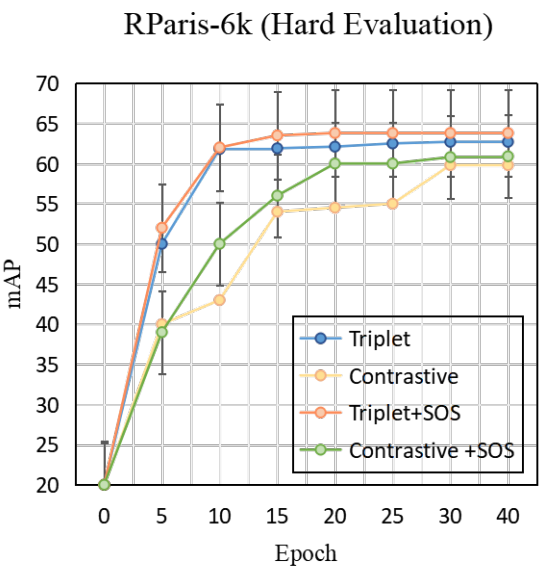
(a) Performance on ROxford-5k(Middle Evaluation)



(b) Performance on ROxford-5k(Hard Evaluation)



(c) Performance on RParis-6k(Middle Evaluation)



(d) Performance on RParis-6k(Hard Evaluation)

Figure 4.14: Performance comparison of different loss functions. The evaluation was performed with Res101 GeM on the ROxford-5k and RParis-6k datasets in middle and hard modesrespectively shown as Figures a-d). These figures show the variation of mAP with the number of training epochs.

the highest performance is selected. In this thesis, the optimization fine-tuned SOI model was obtained using the GoogleLandmark v2 training set. After multiple hyperparameter optimization trials, Table 4.24 illustrates the best hyperparameter settings for this model. The accuracy performance of this optimization fine-tuned SOI model

at various scales is displayed in Table 4.25.

Table 4.24: Hyperparameter configuration of the optimal SOI model trained on the Google Landmarks v2 dataset.

Fine-tuned Model(*GL-Res-SOI*)		
Superparameters	Description	Value
p	Initial value of pooling parameter	3
loss margin	Decision margins for triplet loss functions	1.25
lr	Learning rate of the modle	1e-6
ld	Decay rate for learning rate	1e-2
lambda	SOS loss term weight	10
neg-num	Number of negative images per training tuple	5
query size	Number of randomly selected queries per epoch	2000
pool size	Size of the pool for hard negative examples mining	20000
batch size	Number of training tuples in a mini-batch	8
image size	Artificially specified size of training images	1024
epoch	Total number of training epochs	100

Table 4.25: The mAP performance of *GL-Res-SOI* Model with different scales. Evaluation is performed on Oxford-5k, Paris-6k, ROxford-5k, and RParis-6k datasets.

Scales	mAP of *GL-Res-SOI*					
	Oxford	*Paris*	*ROxford*		*RParis*	
			M	H	M	H
[1]	89.5	91.6	66.6	80.7	45.3	62.1
$[1, \sqrt{2}, \sqrt{1/2}]$	90.9	94.7	69.9	81.6	47.9	64.5

4.5 Generalization Tests

4.5.1 Datasets

To further test the model's generalisation ability, datasets other than Oxford5k, Paris6k and its revisited version were used. The composition of these datasets is described in this subsection.

1M Distractors

This data comprises nearly one million high-resolution (1024×768) photographs crawled from Flickr's 145 most popular tags. In practice, 100k interference images are frequently utilised alongside existing datasets to evaluate the model's generalisation and noise immunity. This thesis employs the ROxford105k and RParis106k datasets, including 100k overlapping images for extensive testing.

Custom

This is a proprietary dataset developed by this thesis, and it consists of 20 objects and around 1,000 digital historical photos. They are obtained via Google and several museum websites. The content includes statues, art, architecture, maps, antiquities and other typical digital history content. Each retrieved object group consists of 10 to 15 photographs of the same object taken from various angles and lighting conditions. Each image in the custom dataset is independent of the sfm120k dataset and the Google Landmarks v2 dataset utilised during the prior training.

4.5.2 Results and discussions

Our results are extensively compared to the state-of-the-art. Table 4.26 summarises the results of the *GL-Res* model and the *GL-Res-SOI* model provided in this paper, as well as previously published data.

Table 4.26: Large-scale image retrieval results of our proposed models in comparison to state-of-the-art methods on the ROxf-RPar, R1M-distractors, and Custom datasets. We analyse using the mAP and mP@10 metrics against the Medium and Hard evaluation protocols. The best results in the table are marked in bold in royal blue, the second best results are in light blue.

Method	ROxford-5K				RParis-6K				ROxford+1M				RParis+1M				Custom	
	M		H		M		H		M		H		M		H			
	mAP	mp@10	mAP	mp@10	mAP	mp@10	mAP	mp@10	mAP	mp@10	mAP	mp@10	mAP	mp@10	mAP	mp@10	mAP	mp@10
Local Feature based descriptor																		
HesAff-rSIFT-ASMK [46]	60.4	85.6	36.4	56.7	61.2	97.9	34.5	80.6	45.0	76.0	25.7	42.1	42.0	**95.3**	16.5	63.4	89.9	97.9
HesAff-rSIFT-ASMK-SP	60.6	86.1	36.7	57.0	61.4	97.9	35.0	81.7	-	-	-	-	-	-	-	-	90.2	98.0
DELF-D2R-R-ASMK [17]	**69.9**	**89.0**	45.6	61.9	78.7	99.0	57.7	**93.0**	-	-	-	-	-	-	-	-	90.5	98.5
CNN based Global descriptor																		
AlexNet-GeM [21]	43.3	62.1	17.1	26.2	58	91.6	29.7	67.6	24.2	42.8	9.4	11.9	29.9	84.6	8.4	39.6	85.1	92.0
VGG16-GeM [21]	61.9	82.7	33.7	51.0	69.3	97.9	44.3	83.7	42.6	68.1	19.0	29.4	45.4	94.1	19.1	64.9	88.4	95.5
ResNet101-RMac [29]	60.9	78.1	32.4	50.0	78.9	96.9	59.4	86.1	39.3	62.1	12.5	24.9	54.8	93.9	28.0	70.0	88.2	95.8
ResNet1010-GeM [O]	45.8	66.2	18.1	31.3	69.7	97.6	47.0	84.9	25.6	45.1	4.7	13.4	46.2	94.0	20.3	70.4	85.1	92.0
ResNet101-GeM-AP [40]	67.5	-			80.1	-	60.5	-	47.5	-	23.2	-	52.5	-	25.1	-	89.9	98.1
ResNet101-GeM-DAME [47]	65.3	85.0	40.3	56.3	77.1	98.0	56.0	88.0	44.7	70.1	22.8	35.6	50.3	94.6	22.0	69.0	88.9	98.0
Our Models																		
GL-Res(baseline)	67.8	87.6	41.7	56.7	77.6	**98.1**	56.3	89.1	46.3	72.3	19.9	35.9	54.3	**95.3**	24.7	73.3	89.97	98.1
GL-Res-SOI(fine-tuned)	**69.9**	86.7	**47.9**	**63**	**81.6**	97.1	**64.5**	**93.0**	**53.5**	**76.7**	**29.9**	**48.9**	**59.2**	94.9	**33.4**	**81.6**	**90.97**	**98.97**

By incorporating SOI blocks and SOS loss functions, our refined model *GL-Res-SOI* delivers state-of-the-art mAP and mP@10 performance for both medium and hard protocols. *GL-Res-SOI* surpasses the baseline model *GL-Res* in ROxford-5k and RParis-6k for mAP with significant increases of 2.1% and 7.2%, respectively, in the most difficult demanding protocols. Also, for the mAP of ROxford-5k-Medium, our method outperformed the state-of-the-art local aggregation method *DELF-D2R-R-ASMK* by 0.3%, RParis-6k-Medium by 2.9% and RParis-6k-Hard by 6.8%.

GL-Res-SOI also performs the most advanced global no-rearrangement single-channel approach for R-1M. For R-1M, it also achieves state-of-the-art performance across global single-pass approaches, exceeding baseline in mAP by 9.2% on ROxford-1M-Medium, 10% on ROxford-1M-Hard, and by 4.9% on RParis-1M-Medium, 9.7% on RParis-1M-Hard.

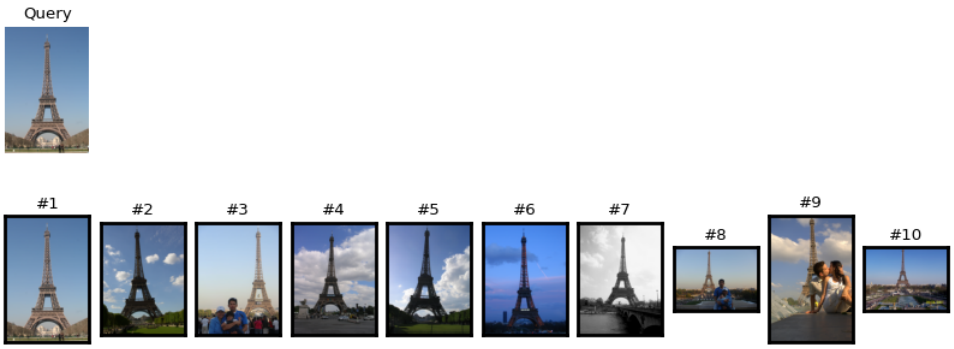
And on our custom-built dataset, *GL-Res-SOI* again performed exceptionally well, achieving 90.97% mAP. This outcome demonstrates the success of our model proposed in this thesis for retrieving of digital historical images.

4.6 Performance of Final Integration Retrieval System

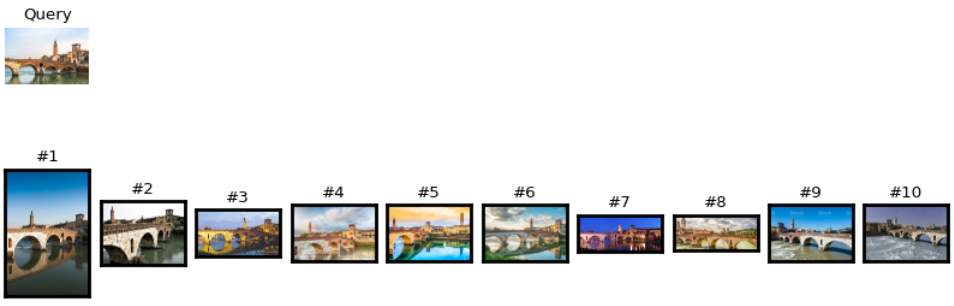
The model for feature extraction proposed in this thesis is a component of the entire system for image retrieval. In practical application, the *GL-Res-SOI* model presented in this thesis is integrated with the index matching module and reranking module developed by team members to produce a digital historical image search system.

The following visualisations derive from search tests conducted with this search method. In each search, the top ten matched images are displayed.

Furthermore, in order to enhance this search system's functionality and make it more user-friendly, it is necessary to equip it with more efficient tools. The interface for the web-based version of the search engine is depicted in Figure 4.19a. The user uploads an image of interest locally and searches for it, and the system delivers the thirty most comparable photographs in the database, as illustrated in Figure 4.19b.



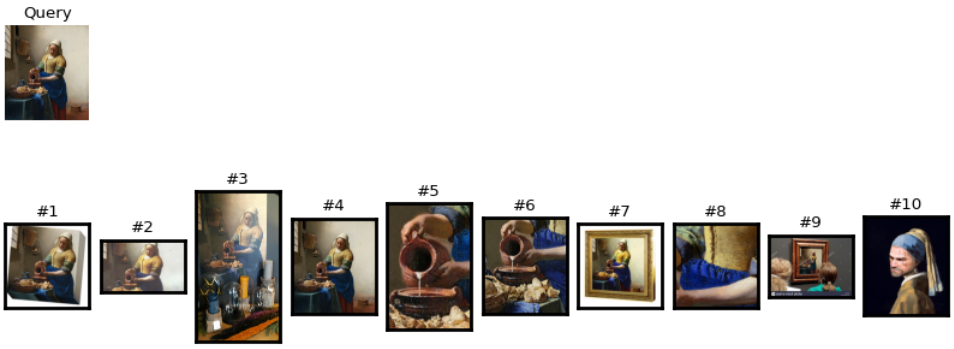
(a) Easy Example.



(b) Hard Example.

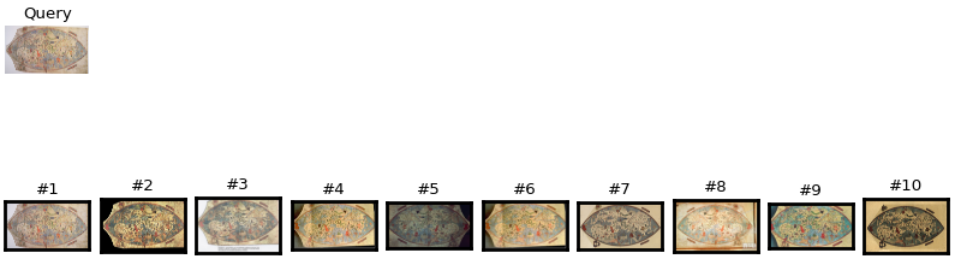
Figure 4.15: Visual search results for examples of historic buildings.

(a) Easy Example.

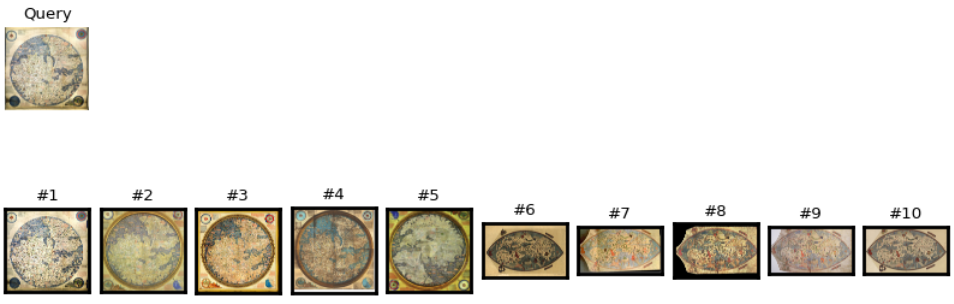


(b) Hard Example.

Figure 4.16: Visual search results for examples of painting arts.

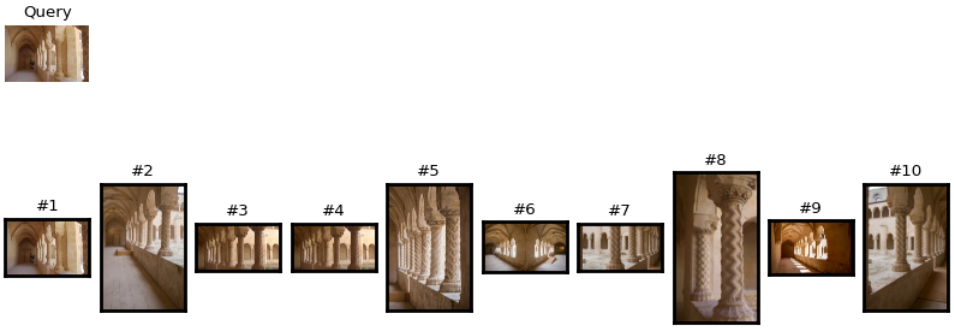


(a) Easy Example.



(b) Hard Example.

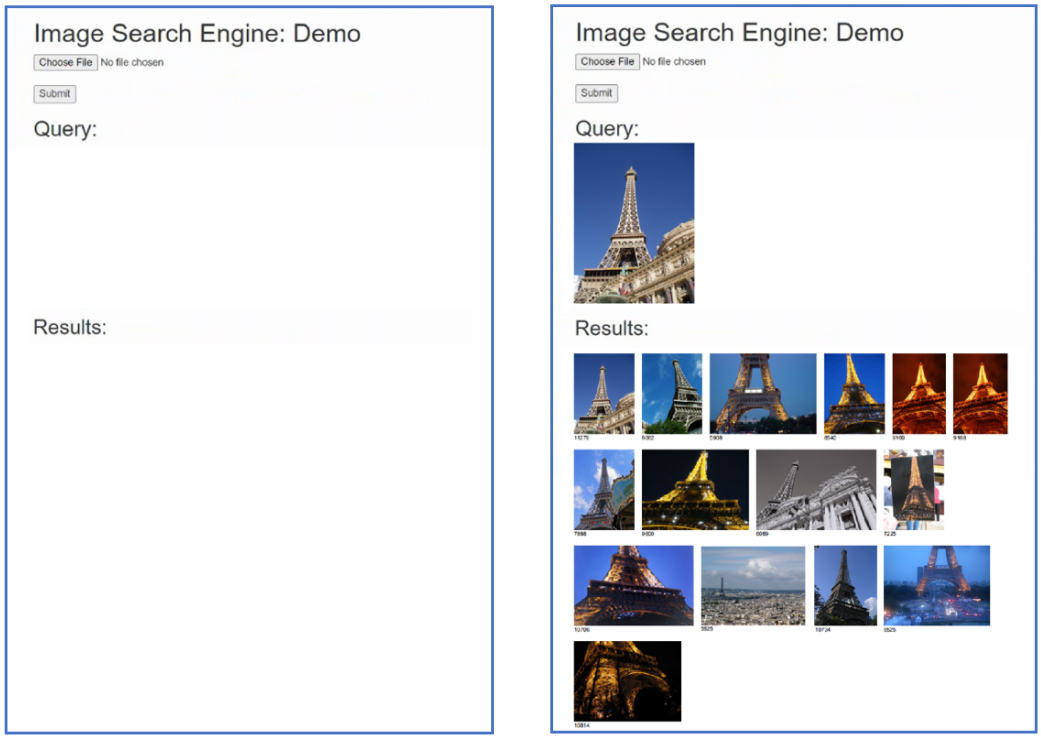
Figure 4.18: Visual search results for examples of maps.



(a) Easy Example.

(b) Hard Example.

Figure 4.17: Visual search results for examples of the sculptures.



(a) Retrieve interface.

(b) Display interface.

Figure 4.19: Digital historical image search engine.

5 Conclusion

In this chapter, we discuss and summarize the work presented in this thesis. We also recommend potential improvements as well as future work that can be investigated further.

5.1 Summary

In recent years, the expansion of the Internet has brought an explosion of visual information, including social media, medical photographs, and digital history. This massive amount of visual content generation and sharing presents new challenges, especially when searching for similar images in databases — Content-Based Image Retrieval (CBIR). How to make a CBIR system acquire visual image features rapidly and precisely is an intense scientific interesting topic.

In this project, we investigated the extraction and optimization of image features with several deep learning based methodologies. We constructed multiple neural network models to enable content-based feature extraction. The model involved in this task is the CNN, the most straightforward and effective model to perform the image retrieval task. We finally proposed two new feature extraction models in this thesis as main contributions: the baseline model *GL-Res* and the fine-tuned model *GL-Res-SOI*.

Our extraction models are fed with images as input and construct high-dimensional global uniform descriptors at their output. Specifically, fine-tuned CNNs were involved in the extraction process, with fine-tuned ResNet101 demonstrating to be the most appropriate network backbone for the task following a series of comparative validation trials. The $H \times W \times D$ feature maps outputted by ResNet can be more representative and discriminative by feature aggregation approaches. During the process of feature aggregation, Generalized Mean Pooling(GeM) [21], Sum-Pooled Convolution(SPoC) [28], and Regional Maximum Activation of Convolutions(RMaC) [29] were selected and validated, with GeM achieving the best performance. The aggregated high-dimensional global features were dimensionally reduced by whitening methods to improve the model's efficiency further. Principal Components Analysis (PCA) [8], learned discriminative whitening [29] post-processing of descriptors, and end-to-end whitening layer [21] were all considered. The end-to-end whitening layer was experimentally demonstrated to reduce dimensionality while maximising precision without excessive loss. Fine-tuned ResNet Backbone, GeM pooling layer, and end-to-end whitening layer consist the baseline model *GL-Res*. Based on the well-performing baseline model, we also add a secondary attention block between the CNN and pooling layers to enhance the interaction information between features. We innovatively enhance the utilisation of second-order information by simplifying its structure and optimizing the cooperating position. We name this fine-tuned extraction model with better performance as

GL-Res-SOI.

In our experiments, two evaluation metrics were utilised to evaluate the outcomes of image retrieval in Oxford, Paris, and their revisited datasets to assess the performance of various models. Mean average precision (mAP) and extract time per query (T_e) evaluate the precision and speed of the models, respectively. According to the current common standards in image retrieval, we used two different mAP evaluation protocols, medium(M) and hard(H).

From the results, the fine-tuned model *GL-Res-SOI* presented in this paper outperformed the state-of-art methods on the most popular test datasets. On the common benchmark RParis6k datasets, *GL-Res-SOI* outperforms the state-of-the-art method *DELF-D2R-R-ASMK* by 2.9%(M) and 6.8%(H). For the challenging datasets with 1M distractors, our *GL-Res-SOI* also achieves outstanding mAP performance. It attains 53.5 mAP(M) and 29.9 mAP(H) on ROxford+1M, as well as 59.2 mAP(M) and 33.4 mAP(H) on RParis+1M. On our self-made historical dataset Custom, *GL-Res-SOI* attains the excellent 90.97 mAP and 98.97 mP@10.

5.2 Contributions

The main contributions of this thesis are as follows:

1. Fine-tune the pre-trained CNNs with medium size datasets to get new CNN models to fit into our task. Fine-tuned CNN highly improve the final mAP.

2.Investigate and optimizes the incorporation of second order attention information in feature map and propose the method SOI. SOI optimizes the embedding position and simplify the structure of SOA without sacrificing accuracy.

3. Combine and compare different feature aggregation methods and whitening methods. Explores how to combine all strengths to make features more compact, discriminative and representative.

4. Propose two new CNN based model for feature extraction. This proposed model beats the state-of-art model and achieves 90.67% mAP on Custom dataset.

5.Design and achieve a fast and user-friendly search engine based on the proposed model. Its respond time per query is around 0.1s. A demo of this search engine has also been posted on GitHub(https://github.com/yanan-huu/Image-Search-Engine-for-Historical-Research)

5.3 Recommendation

Numerous ideas could be integrated with this digital historical image search platform to enhance the work of this thesis. Due to time constraints, some ideas with high workloads could not materialize, and others were of complexity beyond the author's current knowledge. Research directions worthy of further exploration to extend this thesis work are

1) The combination of local and global features of one image

This thesis focuses primarily on the image's global high-level features, which has the advantage of yielding a more abstract and invariant single feature descriptor. In contrast to global features, however, local features are low-dimensional, high robustnesss, and simple to process and compute. Even though the fine-tuned model in this thesis employs a secondary attention module to improve the usage of local spatial information in the feature map extraction step, local features are still not truly integrated. In a multi-layer model, retrieval accuracy would be improved if low-dimensional local features extracted from the bottom layer and high-dimensional global features derived from the top layer were utilised simultaneously.

2) Video Retrieval

In addition to numerous visuals such as photographs, digital resources often include video clips. Theoretically, video content retrieval is possible so long as standard image feature extraction and retrieval methods can be carried out on the primary frames. However, because of the enormous number of frames, feature extraction speed requirements for video material retrieval are substantially greater. Therefore, rapid feature extraction from a large number of still images is a vital study direction.

Bibliography

[1] W. Zhou, H. Li, and Q. Tian, "Recent advance in content-based image retrieval: A literature survey," *arXiv preprint arXiv:1706.06064*, 2017.

[2] S. R. Dubey, "A decade survey of content based image retrieval using deep learning," *IEEE Transactions on Circuits and Systems for Video Technology*, 2021.

[3] W. Chen, Y. Liu, W. Wang, E. M. Bakker, T. Georgiou, P. Fieguth, L. Liu, and M. Lew, "Deep image retrieval: A survey," *ArXiv*, 2021.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[6] X. Li, J. Yang, and J. Ma, "Recent developments of content-based image retrieval (cbir)," *Neurocomputing*, vol. 452.

[7] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[8] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *Computer Vision, IEEE International Conference on*, vol. 3, pp. 1470–1470, IEEE Computer Society, 2003.

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

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[11] L. Zheng, Y. Yang, and Q. Tian, "Sift meets cnn: A decade survey of instance retrieval," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 5, pp. 1224–1244, 2017.

[12] Y. Kalantidis, C. Mellina, and S. Osindero, "Cross-dimensional weighting for aggregated deep convolutional features," in *European conference on computer vision*, pp. 685–701, Springer, 2016.

[13] G. Tolias, R. Sicre, and H. Jégou, "Particular object retrieval with integral max-pooling of cnn activations," *arXiv preprint arXiv:1511.05879*, 2015.

[14] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," *Advances in neural information processing systems*, vol. 27, 2014.

[15] A. S. Razavian, J. Sullivan, S. Carlsson, and A. Maki, "Visual instance retrieval with deep convolutional networks," *ITE Transactions on Media Technology and Applications*, vol. 4, no. 3, pp. 251–258, 2016.

[16] L. Zheng, S. Wang, J. Wang, and Q. Tian, "Accurate image search with multi-scale contextual evidences," *International Journal of Computer Vision*, vol. 120, no. 1, pp. 1–13, 2016.

[17] H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han, "Large-scale image retrieval with attentive deep local features," in *Proceedings of the IEEE international conference on computer vision*, pp. 3456–3465, 2017.

[18] Y. Lv, W. Zhou, Q. Tian, S. Sun, and H. Li, "Retrieval oriented deep feature learning with complementary supervision mining," *IEEE Transactions on Image Processing*, vol. 27, no. 10, pp. 4945–4957, 2018.

[19] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: Cnn architecture for weakly supervised place recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5297–5307, 2016.

[20] A. Gordo, J. Almazan, J. Revaud, and D. Larlus, "End-to-end learning of deep visual representations for image retrieval," *International Journal of Computer Vision*, vol. 124, no. 2, pp. 237–254, 2017.

[21] F. Radenović, G. Tolias, and O. Chum, "Fine-tuning cnn image retrieval with no human annotation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 7, pp. 1655–1668, 2018.

[22] F. Radenović, G. Tolias, and O. Chum, "Cnn image retrieval learns from bow: Unsupervised fine-tuning with hard examples," in *European conference on computer vision*, pp. 3–20, Springer, 2016.

[23] J. Xu, C. Wang, C. Qi, C. Shi, and B. Xiao, "Iterative manifold embedding layer learned by incomplete data for large-scale image retrieval," *IEEE Transactions on Multimedia*, vol. 21, no. 6, pp. 1551–1562, 2018.

[24] Y. Zhao, L. Wang, L. Zhou, Y. Shi, and Y. Gao, "Modelling diffusion process by deep neural networks for image retrieval.," in *BMVC*, p. 161, 2018.

[25] A. S. Razavian, J. Sullivan, S. Carlsson, and A. Maki, "Visual instance retrieval with deep convolutional networks," *ITE Transactions on Media Technology and Applications*, vol. 4, no. 3, pp. 251–258, 2016.

[26] S. Pang, J. Xue, J. Zhu, L. Zhu, and Q. Tian, "Unifying sum and weighted aggregations for efficient yet effective image representation computation," *IEEE Transactions on Image Processing*, vol. 28, no. 2, pp. 841–852, 2018.

[27] T.-T. Do, T. Hoang, D.-K. L. Tan, H. Le, T. V. Nguyen, and N.-M. Cheung, "From selective deep convolutional features to compact binary representations for image retrieval," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 15, no. 2, pp. 1–22, 2019.

[28] A. Babenko and V. Lempitsky, "Aggregating local deep features for image retrieval," in *Proceedings of the IEEE international conference on computer vision*, pp. 1269–1277, 2015.

[29] G. Tolias, R. Sicre, and H. Jégou, "Particular object retrieval with integral max-pooling of cnn activations," *arXiv preprint arXiv:1511.05879*, 2015.

[30] A. Gordo, J. Almazán, J. Revaud, and D. Larlus, "Deep image retrieval: Learning global representations for image search," in *European conference on computer vision*, pp. 241–257, Springer, 2016.

[31] S. Pang, J. Ma, J. Xue, J. Zhu, and V. Ordonez, "Deep feature aggregation and image re-ranking with heat diffusion for image retrieval," *IEEE Transactions on Multimedia*, vol. 21, no. 6, pp. 1513–1523, 2018.

[32] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 3304–3311, IEEE, 2010.

[33] F. Perronnin and C. Dance, "Fisher kernels on visual vocabularies for image categorization," in *2007 IEEE conference on computer vision and pattern recognition*, pp. 1–8, IEEE, 2007.

[34] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 806–813, 2014.

[35] A. Babenko and V. Lempitsky, "Aggregating deep convolutional features for image retrieval," *arXiv preprint arXiv:1510.07493*, 2015.

[36] K. Mikolajczyk and J. Matas, "Improving descriptors for fast tree matching by optimal linear projection," in *2007 IEEE 11th International Conference on Computer Vision*, pp. 1–8, IEEE, 2007.

[37] T. Ng, V. Balntas, Y. Tian, and K. Mikolajczyk, "Solar: second-order loss and attention for image retrieval," in *European Conference on Computer Vision*, pp. 253–270, Springer, 2020.

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

[39] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

[40] J. Revaud, J. Almazán, R. S. Rezende, and C. R. d. Souza, "Learning with average precision: Training image retrieval with a listwise loss," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5107–5116, 2019.

[41] Y. Tian, X. Yu, B. Fan, F. Wu, H. Heijnen, and V. Balntas, "Sosnet: Second order similarity regularization for local descriptor learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11016–11025, 2019.

[42] T. Weyand, A. Araujo, B. Cao, and J. Sim, "Google landmarks dataset v2-a large-scale benchmark for instance-level recognition and retrieval," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2575–2584, 2020.

[43] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *2007 IEEE conference on computer vision and pattern recognition*, pp. 1–8, IEEE, 2007.

[44] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Lost in quantization: Improving particular object retrieval in large scale image databases," in *2008 IEEE conference on computer vision and pattern recognition*, pp. 1–8, IEEE, 2008.

[45] F. Radenović, A. Iscen, G. Tolias, Y. Avrithis, and O. Chum, "Revisiting oxford and paris: Large-scale image retrieval benchmarking," in *CVPR*, 2018.

[46] G. Tolias, Y. Avrithis, and H. Jégou, "Image search with selective match kernels: aggregation across single and multiple images," *International Journal of Computer Vision*, vol. 116, no. 3, pp. 247–261, 2016.

[47] T.-Y. Yang, D. K. Nguyen, H. Heijnen, and V. Balntas, "Dame web: Dynamic mean with whitening ensemble binarization for landmark retrieval without human annotation," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pp. 2913–2922, 2019.