Development of Deep Learning Techniques for Image Retrieval

Farzad Sabahi

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy (Electrical and Computer Engineering) at

Concordia University

Montréal, Québec, Canada

July 2023

CONCORDIA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Farzad Sabahi

Entitled: Development of Deep Learning Techniques for Image Retrieval

and submitted in partial fulfillment of the requirements for the degree of

Doctor Of Philosophy (Electrical and Computer Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Onur Kuzgunkaya

_____ External Examiner
Dr. Panajotis Agathoklis

_____ Examiner
Dr. Chunyan Wang

_____ Examiner
Dr. Wei-Ping Zhu

_____ Examiner, External to Program
Dr. Chun-Yi Su

_____ Thesis Co-Supervisor
Dr. M. Omair Ahmad

_____ Thesis Co-Supervisor
Dr. M.N.S. Swamy

Approved by _____
Dr. Jun Cai, Graduate Program Director

8/23/2023 _____
Dr. Mourad Debbabi, Dean
Gina Cody School of Engineering and Computer Science

Abstract

Development of Deep Learning Techniques for Image Retrieval

Farzad Sabahi, Ph.D.

Concordia University, 2023

Images are used in many real-world applications, ranging from personal photo repositories to medical imaging systems. Image retrieval is a process in which the images in the database are first ranked in terms their similarities with respect to a query image, then a certain number of the images are retrieved from the ranked list that are most similar to the query image. The performance of an image retrieval algorithm is measured in terms of mean average precision. There are numerous applications of image retrieval. For example, face retrieval can help identify a person for security purposes, medical image retrieval can help doctors make more informed medical diagnoses, and commodity image retrieval can help customers find desired commodities. In recent years, image retrieval has gained more popularity in view of the emergence of large-capacity storage devices and the availability of low-cost image acquisition equipment. On the other hand, with the size and diversity of image databases continuously growing, the task of image retrieval has become increasingly more complex. Recent image retrieval techniques have focused on using deep learning techniques because of their exceptional feature extraction capability. However, deep image retrieval networks often employ very complex networks to achieve a desired performance, thus limiting their practicability in applications with limited storage and power capacity. The objective of this thesis is to design high-performance, low complexity deep networks for the task of image retrieval. This objective is achieved by developing three different low-complexity strategies for generating rich sets of discriminating features.

Spatial information contained in images is crucial for providing detailed information about the positioning and interrelation of various elements within an image and thus, it plays an important role in distinguishing different images. As a result, designing a network to extract features that characterize this spatial information within an image is beneficial for the task of image retrieval. In the light of the importance of spatial information, in our first strategy, we develop two deep convolutional neural networks capable of extracting features with a focus on the spatial information. For the design of the first network, multi-scale dilated convolution operations are used to

extract spatial information, whereas in the design of the second network, fusion of feature maps obtained from different hierarchical levels are employed to extract spatial information.

Textural, structural, and edge information is very important for distinguishing images, and therefore, a network capable of extracting features characterizing this type of information about the images could be very useful for the task of image retrieval. Hence, in our second strategy, we develop a deep convolutional neural network that is guided to extract textural, structural, and edge information contained in an image. Since morphological operations process the texture and structure of the objects within an image based on their geometrical properties and edges are fundamental features of an image, we use morphological operations to guide the network in extracting textural and structural information, and a novel pooling operation for extracting the edge information in an image.

Most of the researchers in the area of image retrieval have focused on developing algorithms aimed at yielding good retrieval performance at low computational complexity by outputting a list of certain number of images ranked in a decreasing order of similarity with respect to the query image. However, there are other researchers who have adopted a course of improving the results of an already existing image retrieval algorithm through a process of a re-ranking technique. A re-ranking scheme for image retrieval accesses the list of the images retrieved by an image retrieval algorithm and re-ranks them so that the re-ranked list at the output the scheme has a mean average precision value higher than that of the originally retrieved list.

A re-ranking scheme is an overhead to the process of image retrieval, and therefore, its complexity should be as small as possible. Most of the re-ranking schemes in the literature aim to boost the retrieval performance at the expense of a very high computational complexity. Therefore, in our third strategy, we develop a computationally efficient re-ranking scheme for image retrieval, whose performance is superior to that of the existing re-ranking schemes. Since image hashing offers the dual benefits of computational efficiency and the ability to generate versatile image representation, we adopt it in the proposed re-ranking scheme.

Extensive experiments are performed, in this thesis, using benchmark datasets, to demonstrate the effectiveness of the proposed new strategies in designing low-complexity deep networks for image retrieval.

Acknowledgments

I would like to express my profound gratitude to my PhD supervisors, Prof. M. Omair Ahmad and Prof. M.N.S. Swamy. Their mentorship throughout the development and presentation of my research, along with their unwavering support during every phase of my PhD journey, has been invaluable. I am truly indebted to them for their continuous guidance and for availing the essential research facilities that ensured the success of my research.

I extend my sincere appreciation to Concordia University and the Department of Electrical and Computer Engineering. The nurturing academic environment they provided has been pivotal in aiding me to realize my academic aspirations and achieve my research goals.

On a personal note, I want to express heartfelt gratitude to my beloved wife, Fahimeh. Her endless support and patience are unique and immeasurable. I also owe thanks to my sister Farzaneh for her kindness and always being available when I needed her, and I thank my kind and genius sister, Dr. Farnaz, for her unfailing support. I would like to thank my mother-in-law, Ashraf, and father-in-law, Gholamreza, for their boundless kindness and their help from the very first moment that I met them. I want to thank my sister-in-law, Zahra, whose understanding and support have helped me and my family cope with challenges. Finally, I want to express my deep love for my daughter, Elsa. Her presence has always motivated me to push toward my goals.

I dedicate this thesis to my late mother, a paragon of strength and honesty. Her memory remains a beacon guiding me, despite her absence. My heart aches at the lost opportunity to see her one last time, to hold her hand, to say, "I love you, Maman".

Contents

List of Figures

List of Tables

List of Symbols

$\mathcal{S}$	Spatial Feature Extraction Module
$\mathcal{H}$	Hierarchical Feature Extraction Module
$\mathcal{C}_{\mathrm{i}}$	$i-th$ Convolution Layer
$\mathcal{I}$	Input Feature Tensor
O	Output Feature Tensor
$i_c[m,n]$	(m,n)-th Element of the c-th Channel of Input Tensor of $\mathcal{I}$
$o_c[m,n]$	(m,n)-th Element of the c-th Channel of Output Tensor of O
$\mathcal{N}_{a\times b}^{(m,n)}$	Index Matrix Positioned at (m,n) with a Neighborhood Size of $a \times b$
$\mathcal{P}$	Pooling Function
$\oplus$	Morphological Dilation Operation
$\ominus$	Morphological Erosion Operation

$\odot$	Canonical Multiplication Operation
$Tanh$	Hyperbolic Tangent Activation Function
H	Shannon Entropy Function
θ	Set of Orientations for Image Rotation
α	Block Number for DCT-based Hash Code Generation
β	Block Number for DWT-based Hash Code Generation
W_b	Weight Matrix of Block b
B_b	Influence Vector for Block b
f_b^N	Feature Vector of Block b of Image N of Training Set
$\hat{f}_b^N$	Modified version of the Feature Vector f_b^N
f_b^M	Feature Vector of Block b of Image M of Test Set
$\hat{f}_b^M$	Modified version of the Feature Vector f_b^M
Γ	Initial Retrieval List
K	Number of images in the initial retrieval list Γ
$\bar{0}$	Sequence of 128 Zeros
$\bar{1}$	Sequence of 128 Ones
A_b	Average of Coefficient Values for Block b
$I_{\theta,\alpha}^i$	Block α of Image i Rotated θ Degrees
$\Phi_{\theta,\alpha}^i$	DCT Coefficients of Block α of Image i Rotated θ Degrees

$\mu_{\theta,\alpha}^{i}$	Average of DCT Coefficients of Block α of Image i, Rotated by θ Degrees
$h_{\theta,\alpha}^{i}$	Hash bit for Block α of Image i Rotated θ Degrees, based on DCT
m_{θ}^{i}	Median of Average Values of Coefficients based on DCT for Image i Rotated by θ Degrees
h^{i}	Hash Code for Image i based on DCT
$\Psi_{\theta,\beta}^{i}$	DWT Coefficient of block β of Image i Rotated θ Degrees
$\widehat{m}_{\theta}^{i}$	Median of Average Values of Coefficients based on DWT for Image i Rotated by θ Degrees
$\hat{\mu}_{\theta,\beta}^{i}$	Average of DWT Coefficients of block β of Image i, Rotated by θ Degrees
$\hat{h}_{\theta,\beta}^{i}$	Hash bit for block β of Image i Rotated θ Degrees, based on DWT
$\hat{h}^{i}$	Hash Code for Image i, based on DWT
L^{i}	Left Half of the Hash Code $\hat{h}^{i}$
R^{i}	Right Half of the Hash Code $\hat{h}^{i}$
$\mathcal{T}_{\mathrm{L}}$	Tree Constructed using L^{i} for all i images
$\mathcal{T}_{\mathcal{R}}$	Tree Constructed using R^{i} for all i images
$\mathcal{T}_{\mathrm{h}}$	Tree Constructed using h^{i} for all i images
L^{q}	Left Half of the DWT-based hash Code for the Query Image
R^{q}	Right Half of the DWT-based hash Code for the Query Image

h^q	DCT-based Hash Code for the Query Image
δ_1	Candidate List Obtained from Searching for L^q in the Tree $\mathcal{T}_L$
δ_2	Candidate List Obtained from Searching for R^q in the Tree $\mathcal{T}_R$
δ_3	Candidate List Obtained from Searching for h^q in the Tree $\mathcal{T}_h$
δ_{123}	Intersection of δ_1, δ_2,and δ_3
δ_{12}	Intersection of $\delta_1$and δ_2 Excluding δ_{123}
δ_{13}	Intersection of $\delta_1$and δ_3 Excluding δ_{123}
δ_{23}	Intersection of δ_2 and δ_3 Excluding δ_{123}
$sign$	Sign Function
ℓ_2	Euclidean Norm
$\mathcal{O}$	Big-O Notation
$max(X)$	Maximum Element in Pooling Window X
$min(X)$	Minimum Element in Pooling Window X
$P(k)$	Precision at Rank k
$R(k)$	Recall at Rank k

List of Abbreviations

DL	Deep Learning
CNN	Deep Convolutional Neural Network
CBIR	Content-based Image Retrieval
AP	Average Precision
mAP	Mean Average Precision
HNN	Hopfield Neural Network
FLOP	Floating Point Operation
DCT	Discrete Cosine Transform
DWT	Discrete Wavelet Transform
BST	Binary Search Tree
BBST	Balanced Binary Search Tree

PConv.	pointwise convolution
$ReLU$	Rectified Linear Unit
$LReLU$	Leaky Rectified Linear Unit
SGD	Stochastic Gradient Descent
PW Conv.	Point-wise Convolution
DW Pool.	Depth-wise Pooling
Conv.	Convolutional Layer
$CONCAT$	Concatenation
D.Conv.	Dilated Convolution
MoF	Morphological Feature Generating Residual Block
Max-m-Min	Maximum minus Minimum Pooling
BHIR	Block-wise Hopfield Network-based Image Retrieval
MSFR	Multi-source Spatial Feature Generating Residual Block
HMSR	Hierarchical and Multi-scale Spatial Feature Generating Residual Block
MorIRNet	Morphological Feature generation-based Image Retrieval Network
SIFT	Scale Invariant Feature Transform
DQE	Discriminative Query Expansion
TXEBFSR	Texture Expansion based on Feature Selection Retrieval

Chapter 1

Introduction

1.1 Overview of Image Retrieval

"A picture is worth one thousand words." This proverb comes from Confucius - a philosopher who lived more than 2500 years ago. Humans have often used drawings to convey information. Throughout history, mankind often uses visual representations as a medium for information convey. Evidence of this is the prehistoric cave paintings depicting the perilous hunting experiences as well as several paintings from Pharaohs' ritual practice wall paintings found in Egypt. Today, we are surrounded by visual information almost everywhere. With the influence of the digital world, a substantial amount of information is available only in digital formats, including images. Nowadays, digital images are gaining the main visual medium across various platforms, from personal photo libraries to medical imaging. With the rise of computational power and decreasing storage costs, Images are playing an increasingly important role in people's daily lives. We cannot access or make use of a large collection of images unless it is organized to allow efficient browsing, searching, and retrieval processes. Image retrieval involves retrieving images similar to a user-

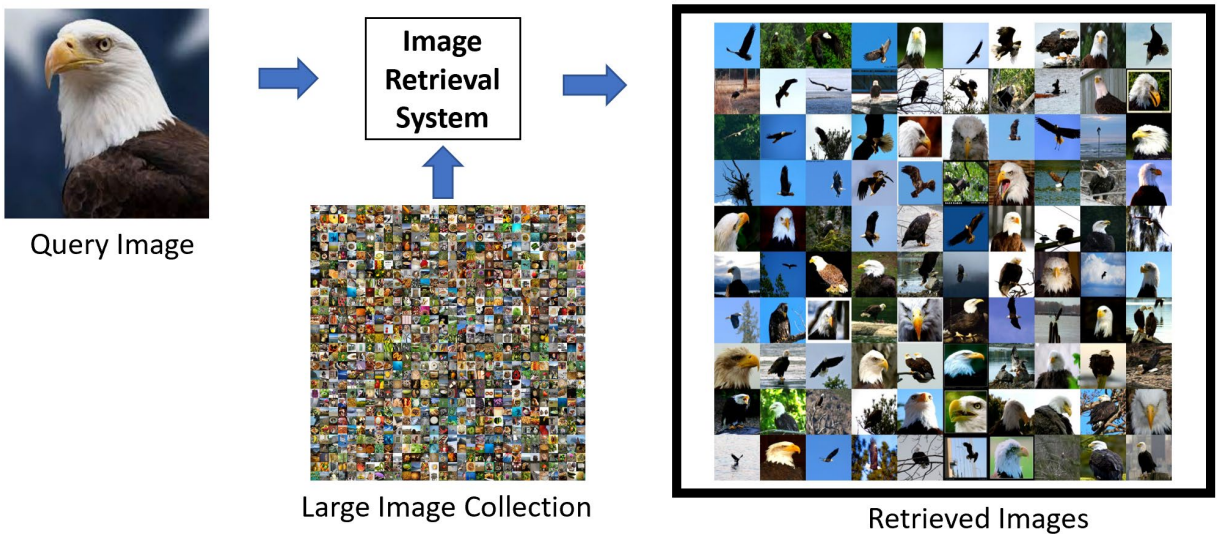

Figure 1.1: Illustration of the task of image retrieval.

specified textual or pictorial object (query) from the images of a database. An illustration of the image retrieval task in shown in Figure 1.1.

With the rapid growth of digital image repositories and the increasing demand for efficient and accurate retrieval systems, image retrieval has emerged as a crucial research area. The task of image retrieval has evolved significantly over the years, advancing from low-level feature-based approaches to deep learning-based methods. Low-level feature-based image retrieval methods focus on extracting and comparing features such as color, texture, and shape from images.

In recent years, image retrieval has gained more attention in light of the emergence of applications employing computer vision and artificial intelligence techniques. Image retrieval processes have also become more complex owing to the exponential growth of the size and diversity of image databases resulting from the availability of low-cost digital acquisition equipment and the emergence of large-scale storage devices.

A typical image retrieval method performs the task by processing the visual information contained in an image and creating a feature vector based on the image content. Any subsequent query operations take place solely within the generated feature vectors, not the raw image data. In fact, every image in the image database, including the query image, is analyzed, and a compact representation of it is stored as a feature vector. This feature vector acts as a unique signature for the image that represents it during similarity matching. The key processes conducted in a typical image

retrieval method are shown in Figure 1.2.The units shown in the figure are explained further as follows:

- **Feature extraction:** This process is the core of any image retrieval method. Every image must be transformed into numerical values (a feature vector) in order to make the visual information understandable by a computer. The representable ability of the generated feature vectors is crucial to obtain high retrieval performance.
- **Feature database:** all the feature vectors are stored in the feature database. The algorithm that is used to generate this database is the same for processing all the images, including the query image.
- **Feature matching:** This step involves comparing the feature vector of the query image to each feature vector in the feature database. A suitable similarity measurement method is used to calculate the distance between the feature vectors.
- **Sorting:** During this final stage, images are ranked based on the similarity scores obtained from the feature-matching process. The images corresponding to the feature vectors that are closest to the query image are expected to have the highest scores and will be reported as the top matches.

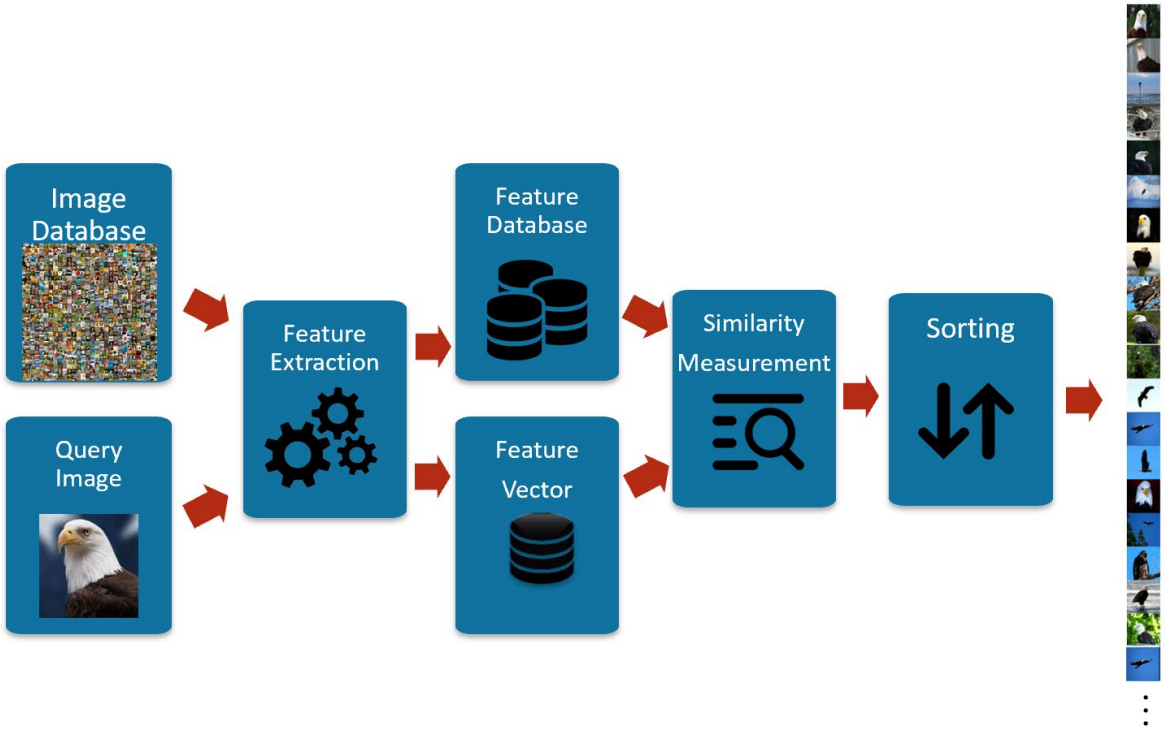

Figure 1.2: Details of an image retrieval system.

1.2 Importance of Image Retrieval

Image retrieval plays a vital role in a wide range of applications ranging from medical diagnostics to e-commerce. In medical diagnostics, image retrieval systems can efficiently locate relevant medical images, such as X-rays, from available medical image databases [1]. Similarly, the e-commerce industry exploits image retrieval systems to offer visually similar product recommendations [2]. Several online retail stores have integrated image retrieval into their platforms, facilitating product discovery. Given the ongoing advancements in image retrieval techniques, the scope of its application continues to broaden across various industries. This trend necessitates more efficient and effective image retrieval methods capable of analyzing the ever-increasing volume and complexity of image databases for real-world applications. Consequently, the study and improvement of image retrieval techniques are not only relevant but crucial in our progressively digital world.

1.3 Brief Literature Review

The quality of features extracted from images is crucial in achieving high retrieval performance. Therefore, it is not surprising that a significant portion of research in the field of image retrieval has been done to improve the representational capacity of the obtained feature vectors. The progression can be divided into three distinct eras, each characterized by differences in how raw pixel values of images are transformed into feature vectors.

In the first era, the focus was on extracting global low-level features to describe the characteristics of an image in the form of a single feature vector. An example is the Query By Image Content (QBIC) system [3], which uses global characteristics like color and texture to represent images. However, these global features frequently failed to discriminate between the visual contents of different images effectively. Due to this deficiency in representation, researchers shifted their focus towards methods based on local low-level features, which was the start of the second era. In this era, methods involve identifying salient patches within a given image and extracting the feature vector of each patch to represent the image as a combination of several such vectors [4], [5]. An example is scale-invariant feature transform (SIFT) [4] which is a method designed to extract

distinctive invariant features from images, which can be used to perform reliable matching between different views of an object or scene. Despite these advances, a common drawback of these early low-level feature-based methods is their struggle with understanding high-level visual content in images [6]. As a result, traditional image retrieval systems that relied on low-level features often underperformed due to the inherent limitations associated with low-level features [7].

The advent of deep Convolutional Neural Networks (CNNs) has initiated a new era for image retrieval, fundamentally revolutionizing the field with their remarkable feature extraction capabilities. Unlike handcrafted features, CNNs automatically learn discriminative and robust features directly from raw pixel values, significantly outperforming the low-level-based image retrieval methods. The strength of CNNs lies in their architecture, which consists of multiple convolutional layers, each learning increasingly complex features from the outputs of the previous layer through several convolution operations followed by nonlinear activation functions. The stack of several convolutional layers forms a hierarchical structure, with low-level features (such as edges and textures) learned in the early shallow layers and high-level features (such as objects) identified in the deeper layers. This hierarchical feature extraction process enables CNNs to integrate critical visual characteristics within an image into high-level features, resulting in feature sets that are highly representative and discriminative, allowing for superior image representation and, thereby, enhancing the retrieval performance.

Early deep learning-based techniques primarily utilized pre-trained networks to obtain features [8]–[17]. For instance, in [8], the authors investigated the use of feature maps from the top fully connected layer of a large CNN for image retrieval, transforming a given image into a single vector. The lack of spatial information in the fully connected layers has led to utilizing feature maps available in the convolutional layers, as proposed in [15]. Convolutional layers contain spatial information, which is crucial for applications where the location of objects is helpful for generating more informative features. However, the discriminative power of these methods is inherently limited by the pre-trained deep networks from which the feature vectors are obtained. To address this issue, some works have been developed to combine deep features and high-end low-level features. For example, in [16], the authors proposed a technique to combine features obtained from a deep network with and that of SIFT features in order to exploit the strengths of both types of features. While some improvements were achieved by combining low- and high-level features, the

performance degraded when there is a high degree of inter-class similarity. To mitigate this issue, in [17], the feature maps of multiple deep networks can be fused in order to improve the representational ability of the feature vectors. While these methods provide a good performance, usage of multiple deep networks renders them resource-intensive and impractical for real-world applications. It is seen from the above discussion that the main focus of image retrieval methods using pre-trained networks is on designing effective mechanisms to extract the best possible features. However, the performance of these techniques is constrained by the design of the utilized pre-trained networks, which are not specifically designed for image retrieval tasks, thereby resulting in suboptimal performance [18]. This paves the way for designing new architectures, which may offer more tailored solutions for image retrieval tasks.

Apart from improving representational capacity to enhance retrieval performance, another promising research direction is to re-rank an initial retrieval list obtained from a retrieval process in such a way that images similar to a given query image are ranked higher in the list. Some re-ranking methods have been proposed to enhance retrieval performance [20-34]. For example, in a query expansion method proposed in [19], a new feature vector for the query image is constructed by averaging the feature vectors of the top-ranked images from the initial retrieval list. This revised feature vector is then used to search the image dataset again. However, query expansion requires performing an entirely new search, which can be particularly resource-intensive for large-scale datasets. Using k-nearest neighbor is another technique that is used to exploit the similarity relationship between top-ranked images for re-ranking the initial retrieval results. Similarly, in [20]–[23], the k-reciprocal nearest neighbor is used where two images are defined as k-reciprocal nearest neighbors if both appear in each other top-k list when one image serves as the query image. Discriminative Query Expansion (DQE) [24] proposes a re-ranking method based on the support vector machine model. Building upon the idea of feature-specific expansion, methods like Texture Expansion based on Feature Selection Retrieval (TXEBFSR) [25] are proposed. This method creates a new feature vector for the query image by averaging texture features calculated on top-ranked images to perform a new image retrieval process. Recently, a new re-ranking scheme for image retrieval is proposed [26]. This method transforms an initial retrieval list into a correlation matrix. This matrix is then used to train a CNN to learn the semantic relevance among the images. After the network has learned these relationships, the images in the list are re-ranked based on

their relevance to the original query. However, a significant limitation of this method is its dependency on the training of a CNN and the requirement for a ground-truth relevance matrix. The latter can be particularly challenging to obtain, especially when dealing with large datasets or in scenarios where the relationships between images are not well-defined or known in advance. Very recently, a new re-ranking method referred to as RbQE has been proposed [27]. RbQE has two search stages: a rapid search and a final search. In the rapid search, using feature vector of the query images, retrieval process is done in each class and by calculating mean values of deep features of top-ranked images from each class, a feature vector for each class is computed. In the final search, among the computed feature vectors, the one that is most similar to the original query is used to re-query the database. While the aforementioned re-ranking techniques demonstrate promising performance, they suffer from high computational complexity, leading to slow and resource-intensive operations and, thereby, making them impractical for real-world applications. This necessitates the development of a computationally efficient re-ranking approach that effectively improves retrieval performance.

1.4 Motivation and Objective

It is seen from the literature review in Section 1.2 that the design of efficient image retrieval methods is very crucial in many real-world computer vision applications, and the performance of any image retrieval method largely depends on the quality of representation of the feature vectors. Many existing image retrieval methods, although providing respectable performance, may fall short when faced with the continuously growing size and complexity of image datasets. Moreover, they achieve their performance at the expense of high computational complexity, making them impractical for real-world applications. Furthermore, there is a noticeable scarcity in the design of new deep architectures that focus on improving the representational capacity of the network. The incorporation of crucial information for obtaining high retrieval performance for the task of image retrieval, such as spatial and structural information, remains largely unexplored, despite the potential for these tailored architectures to significantly enhance the performance of a deep image retrieval network.

The objective of the thesis is twofold. First, we propose several novel residual blocks focusing on extracting textural and structural information to be used in convolutional neural networks to enhance the representational capacity of the networks for image retrieval tasks. The proposed residual blocks include various novel modules, including hierarchical spatial feature extraction, multi-scale feature extraction, multi-source spatial feature extraction, edge feature extraction, and morphological feature extraction. By employing these new modules in residual framework, various deep convolutional neural networks are proposed that learn rich sets of features to enhance the retrieval performance significantly. The second objective is to improve the retrieval performance of an initial retrieval list through a novel re-ranking method, utilizing the speedy and proficient nature of image hashing techniques for image representation. The proposed method aims to improve the retrieval performance of an initial retrieval list with minimal computational overhead.

1.5 Organization of the Thesis

The organization of this thesis is as follows. Chapter 3 presents two designs of residual blocks and their applications in deep convolutional neural networks in order to improve the representational capacity of the deep networks. These designs focus on the integration of spatial information into feature maps to enhance the representational capacity of a deep convolutional neural network for image retrieval. Chapter 4 explores the application of the idea of guided feature generation in deep networks. The chapter proposes a new method designed to enhance the representational ability of feature vectors obtained from a deep network by guiding the network to incorporate textural and structural information using processes such as morphological operations and edge feature extraction. In addition, a new pooling operation is presented, which focuses on producing rich sets of edge features to further improve the network learning quality. Chapter 5 proposes a novel hashing-based re-ranking technique aimed at enhancing the performance of any image retrieval technique. This innovative technique explores how image hashing can be utilized to refine the initial retrieval list, hence boosting the retrieval performance of an image retrieval method through efficient post-processing of the results. Finally, in Chapter 6, the thesis concludes with a summary of the significant findings and contributions, along with identifying potential research directions for future investigations.

Chapter 2

Background Material

In this chapter, we provide a brief review of background material that is useful for understanding the work presented in this thesis. We start with a brief overview of Hopfield Neural Networks, presenting the basic principles of this recurrent neural network architecture recognized for its associative memory capabilities. This is followed by a discussion on image hashing, a technique used for efficient and effective image representation. Next, we present the idea of re-ranking, a method designed to refine an initial retrieval list to improve the retrieval performance further. Recognizing the importance of fast image search, we then review the balanced binary search tree, a data structure designed for efficient data storage and retrieval. Further, the pooling operation in deep convolutional neural networks is discussed, an essential process in deep networks that reduces dimensionality while preserving crucial information. Finally, we present an overview of morphological operations and image processing techniques proficient in processing the images based on their shapes and structural information.

2.1 Hopfield Neural Network

A Hopfield network is a specific type of recurrent neural network designed to model associative memory. Associative memory is characterized by its ability to store and recall relationships between patterns stored within the network [28]–[30]. Due to this fundamental attribute, Hopfield networks have been utilized in various computer vision tasks, including image retrieval [31], [32]. The goal of using Hopfield networks is to memorize $\mathcal{P}$ different patterns $X_i^{\mathcal{K}}$,$\mathcal{K} = 1, \dots, \mathcal{P}$, $i = 1, \dots, n$. Each pattern consists of n components. The network is defined by a weight matrix, $\mathcal{W}$, an $n \times n$ matrix in which element $\mathcal{W}_{ij}$ equals the weight attached to the connection between node i and node j in the network. The weight matrix is computed as follows:

$$\mathcal{W}_{ij} = \frac{1}{\mathcal{P}} \sum_{K=1}^{\mathcal{P}} X_i^{\mathcal{K}} X_j^{\mathcal{K}} \qquad (2.1)$$

$$i = 1, \dots, n \quad , \quad j = 1, \dots, n \quad , \quad \mathcal{W}_{ij} = \mathcal{W}_{ji}$$

where $\mathcal{P}$ is the number of patterns that we aim to store in the network. Given a Hopfield network holding some patterns, the retrieval process begins by providing an initial pattern, denoted as q. The network then retrieves the closest pattern to the initial pattern using the weight matrix, $\mathcal{W}$. The retrieval process starts by using q as the initial pattern of the network. The pattern at time $t + 1$, is a function of the weight matrix multiplied by the pattern at time t or $v(t)$, as

$$v_i(t + 1) = sign\left[\sum_{j=1}^{\mathcal{N}} W_{ij}\, v_j(t)\right] \qquad (2.2)$$

Here, $sign$ function is defined as

$$sign(x) = \begin{cases} 1 & if\ input \geq 0 \\ -1 & if\ input < 0 \end{cases} \qquad (2.3)$$

Given $v(0) = q$, after all components of $v(t + 1)$ have been calculated, $v(t + 1)$ will be entered into the network to obtain another pattern until the state becomes stable. The state $v(t)$ is

considered stable if the difference $v(t+1) - v(t)$ is less than a predetermined threshold. Upon the completion of the stabilization process, $v(t)$ is the retrieved output.

Some image retrieval methods have been proposed that take advantage of the capabilities of Hopfield networks. For instance, in [31], a Hopfield network is used for content-based image retrieval, where the network learns to identify and retrieve images that are most similar to the query image. In [32], the Hopfield network is employed to create an initial retrieval list, and a re-ranking technique is employed to improve the retrieval performance.

2.2 Image Hashing

Image hashing is a computational technique used in image processing to generate a concise and unique representation, known as a hash code, for a given image. The fundamental aim of image hashing is to formulate a compact, fixed-length binary code that embodies the visual attributes of an image [33]–[37]. This process typically necessitates the extraction and transformation of features of an image. These features, once transformed into a hash code, enable more efficient storage of image data and also improve resource efficiency during any image processing process. Therefore, by encapsulating the visual content of images into hash codes, image hashing can effectively complement other techniques, thereby enhancing efficiency in applications such as image authentication, duplication detection, and digital watermarking. For example, in [38], image hashing is employed for image authentication, a process that seeks to verify the integrity and authenticity of digital images. The work in [39] utilizes image hashing for copy detection to identify image duplication or plagiarism. Lastly, the work in [40] proposes a method incorporating image hashing with image watermarking. This method involves transforming a digital signature into a binary hash vector and embedding it within the transformed coefficients of the original image, thereby improving the robustness of the image authentication process.

2.3 Re-ranking

Re-ranking is a post-processing task of refining and initially ranked list of images obtained from an image retrieval technique for a given query image, with the goal of enhancing retrieval

performance in an efficient manner. By employing various methods such as query expansion and k-nearest neighbor algorithms, the re-ranking process focuses on re-arranging the retrieved images to bring the ones most similar to the query image to the forefront of the list in order to improve the overall retrieval performance.

One common technique utilized in re-ranking is query expansion [27], which aims to improve the representational capacity of the feature vector of the query image by incorporating additional information into it. This expanded query is then used to retrieve a new list of images, providing more accurate results. This expansion is typically achieved by using techniques such as pseudo-relevance feedback, which employs the top-ranked images from the initial retrieval as a basis for expanding the query representation.

Another widely used approach in re-ranking is based on k-nearest neighbor algorithms [41]. In this method, the similarity between the query image and the retrieved images is reassessed using a suitable distance metric in order to identify the nearest neighbors. The re-ranking process reorders the initial list by considering the similarities between the query and these nearest neighbors.

2.4 Balanced Binary Search Tree

One of the pillars of efficient search algorithms in computer science is the Binary Search Tree (BST). A BST is a tree data structure in which each node contains a key and corresponding value. A distinguishing characteristic of BSTs is that the key of any node is always larger than all keys in its left subtree and smaller than all keys in its right subtree. This property allows BSTs to have an average time complexity of $\mathcal{O}(\log n)$ for operations such as search [42]. However, in certain scenarios, the absence of a height constraint can cause a BST to become skewed, which can deteriorate the time complexity of these operations to $\mathcal{O}(n)$. To overcome the skewness issue of BST, balanced binary search trees (BBST) can be used. BBSTs are a variant of BSTs that maintain balance by ensuring that the height difference between the left and right subtrees of any node never exceeds one [43]. This condition helps prevent the tree from becoming excessively skewed and thus guarantees more predictable time cost and efficient operations. Figure 2.1 shows an illustrative comparison, demonstrating that a search operation in a BBST would require fewer comparisons than in an unbalanced BST. For example, searching for the node marked as ‘7’ requires only

two comparisons in the BBST, whereas, in the unbalanced BST, the number of comparisons rises to six.

Binary search trees have been used in some image processing applications. For example, in [44], a method is proposed that utilizes Binary Search Trees for approximate nearest neighbour searches in high-dimensional binary vectors. In [45], a geometrically motivated approach is introduced that effectively compresses binary search trees for more efficient nearest-neighbor searches.

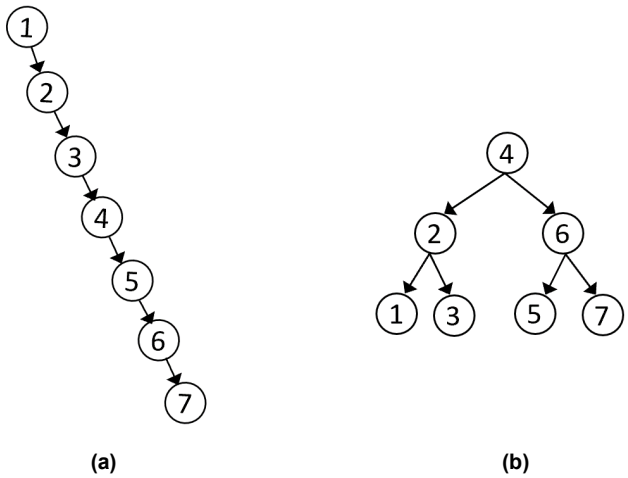

Figure 2.1: Unbalanced (a) vs. balanced (b) binary search trees.

2.5 Pooling Operation

A pooling operation can be considered as a mapping of a set of input feature maps (also called channels) $\mathcal{I}$ to a set of output feature maps $\mathcal{O}$ given by

$$\mathcal{O} = \mathcal{P}(\mathcal{I}) \tag{2.4}$$

where $\mathcal{P}$ is the pooling operation. Let $i_c[m,n]$ represent the c-th two-dimensional channel of the input feature tensor $\mathcal{I}$. In practice, the pooling operation consists of placing at the (m,n)-th position of the c-th feature map, a window of size $(a+1, b+1)$ and performing

$$o_c[m,n] = \underset{x,y \in \mathcal{N}_{a\times b}^{(m,n)}}{\mathcal{F}} i_c[x,y] \tag{2.5}$$

where $\mathcal{N}_{a\times b}^{(m,n)}$ is a set of indices given by

$$\mathcal{N}_{a\times b}^{(m,n)} = \begin{bmatrix} m,n & m+1,n & \cdots & m+a-1,n \\ m,n+1 & m+1,n+1 & \cdots & m+a-1,n+1 \\ \vdots & \vdots & \vdots & \vdots \\ m,n+b-1 & m+1,n+b-1 & \cdots & m+a-1,n+b-1 \end{bmatrix} \tag{2.6}$$

and $\mathcal{F}$ is a pooling operation performed on the elements of $i_c[m,n]$ that fall within the window. For example, using the summation operation as the pooling operation $\mathcal{F}$, Equation (2.5) becomes

$$o_c[m,n] = \sum_{x,y\in\mathcal{N}_{a\times b}^{(m,n)}} i_c[x,y] \tag{2.7}$$

2.5.1 Average and Max Pooling

Average pooling and max pooling are two of the most common pooling operations used in convolutional neural networks. Average pooling performs the pooling operation as:

$$o_c[m,n] = \frac{1}{ab} \sum_{x,y\in\mathcal{N}_{a\times b}^{(m,n)}} i_c[x,y] \tag{2.8}$$

Average pooling diminishes critical information by smoothing the extreme values in the feature maps, thus lowering the performance of a network [46].

The max pooling operation involves obtaining the maximum value for each pooling window as follows:

$$o_c[m,n] = \underset{x,y\in\mathcal{N}_{a\times b}^{(m,n)}}{\boldsymbol{max}} i_c[x,y] \tag{2.9}$$

The shortcoming of max pooling is that it ignores everything but the maximum value. Therefore, this method misses all crucial information except the most significant value, thus making it difficult to consider other informative features that exist in the input signal.

Without evaluating these methods in a network on a given dataset, it is difficult to predict effectively whether max pooling or average pooling will yield higher performance [47]. For example, max pooling is not robust to scenes with clutter, as it produces the maximum response at clutter locations rather than object locations [48]. In such scenarios, average pooling is more

effective. The optimal choice depends on the image dataset, and one should apply both methods to see which one provides better results. However, in addition to the greater computational requirements involved, investigating both pooling operations is not feasible for real-world applications, as the data used in these applications change over time.

2.5.2 Other Pooling Methods

Some variants of average pooling and max pooling have been developed to address the abovementioned shortcomings. One variant of average pooling is Lp pooling [49]. In this pooling operation, a weighted average of the pooling window is calculated as

$$o_c[m,n] = \left(\frac{1}{ab} \sum_{x,y \in \mathcal{N}_{a\times b}^{(m,n)}} (i_c[x,y])^p \times G(x,y) \right)^{\frac{1}{p}} \tag{2.10}$$

where $G(x,y)$ is a Gaussian kernel. Rank-based average pooling [50] is another variant of average pooling that attempts to address the issue that standard average pooling decreases the importance of extreme values. This method utilizes an average of the top t highest values in the pooling window and calculates the output as follows:

$$o_c[m,n] = \frac{1}{t} \sum_{x,y \in \mathcal{N}_{a\times b}^{(m,n)}, r_{(x,y)} \leq t} i_c[x,y] \tag{2.11}$$

where $r_{(x,y)}$ is the rank of the value located at (x,y) in the pooling window. However, this method cannot be generalized effectively, as the value of t has to be selected empirically based on the database used.

Mixed pooling [51] is a combination of the average and max pooling operations defined by

$$o_c[m,n] = \alpha \max_{x,y \in \mathcal{N}_{a\times b}^{(m,n)}} i_c[x,y] + (1-\alpha) \frac{1}{ab} \sum_{x,y \in \mathcal{N}_{a\times b}^{(m,n)}} i_c[x,y] \tag{2.12}$$

where α is an activation parameter, which for a given window is set randomly to a value of either 0 or 1, indicating the choice of using the max pooling or average pooling. The proposed method changes the pooling regulation scheme in a stochastic manner. However, this method suffers from

possible overfitting, as randomness may cause unpredictable biased training. A method similar to the mixed pooling is the stochastic pooling [47]. Stochastic pooling is less vulnerable to overfitting because the activation parameter is chosen for each pooling layer based on a multinomial distribution. This method is superior to the mixed pooling and the L_p pooling, but it suffers from a higher computational complexity. Another variant of max pooling was introduced in [52] that applies max pooling several times on the same pooling window but with different pooling window sizes. The pooled features are then fused to form the final feature map. Even though this pooling method improves the representational capacity of the generated feature maps, it adds many operations to the pooling layer that significantly decrease the learning speed.

Another pooling operation is the spectral pooling proposed in [53], which has been shown to outperform max pooling since it preserves more information for the same output dimensionality by applying linear low-pass filtering. While spectral pooling offers a fast pooling operation, the features learned by the network fail to preserve the image's visual content effectively, as the resulting feature maps are the output of a blurred version of the input.

2.6 Morphological Operations

Morphological operations are commonly employed in image processing [54]. These operations can be used to develop methods to analyze the shape and form of the objects in images. In view of these characteristics of morphological operations, they have been extensively used in conventional schemes for image retrieval to extract features. They also have been used in convolutional neural networks, but with the exception of work in [55], only in applications other than image retrieval. For example, morphological operations have been utilized in CNN for the purpose of edge detection in [56] and for the enhancement of image super-resolution in [57]. The work in [55] is the only work in which an image retrieval deep network has been designed in which morphological operations have been utilized for automatic feature extraction. This network has used the basic morphological operations of erosion and dilation. In view of the importance of morphological operations, it would be worthwhile to explore the use of other morphological operations in the design of deep networks for the task of image retrieval.

2.7 Evaluation Metric

Mean Average Precision (mAP) is a standard metric for evaluating the performance of image retrieval techniques. This metric takes into account the order of appearance of similar images in the ranked list. The first step in calculating mAP is to compute the precision at rank k, denoted as $P(k)$. It is defined as:

$$P(k) = \frac{TP(k)}{TP(k) + FP(k)} \tag{2.13}$$

where $P(k)$ is the precision until position k in the retrieval ranked list for a given query. $TP(k)$ is number of true positives up to position k, indicating the number of similar images that have been correctly retrieved among the top k results. $FP(k)$ is the number of false positives up to position k, indicating the number of dissimilar images that have been retrieved among the top k results.

Building upon precision at rank k, average precision for a query q, denoted as AP_q, is calculated as follows:

$$AP_q = \frac{1}{R_q} \sum_{k=1}^{n} P(k) \times S@k \tag{2.14}$$

where R_q is the total number of relevant documents for query q. $P(k)$ is the precision at rank k for query q. $S@k$ is an indicator function that equals 1 if the item at rank k is a similar image to the query and 0 otherwise. n is the total number of retrieved images for query q.

Finally, mAP is calculated as the mean of AP_q values across all queries as follows:

$$mAP = \frac{1}{Q} \sum_{q=1}^{Q} AP_q \tag{2.15}$$

where Q is the total number of queries. The mAP score ranges from 0 to 1; a score close to 0 suggests that similar images are mostly ranked lower in the ranked list, whereas a score close to 1 indicates that similar images appear predominantly at the top of the list.

2.8 Summary

This chapter has reviewed two primary groups of methods for achieving high retrieval performance in image retrieval: enhancing the representational capacity of the features in deep networks and refining the initial retrieval list through re-ranking techniques. The review discusses the usage of deep learning to obtain representative and discriminative features. Additionally, the chapter reviews some processes that can be employed to enhance the representational capacity of deep networks. The chapter also reviews various pooling operations and highlights the importance of pooling in the learning quality of the network. Regarding the re-ranking technique, the review reveals the importance of computational efficiency. Since generating the initial retrieval list can be a resource-intensive task, utilizing high computationally expensive re-ranking techniques can result in slow, resource-intensive operations, making them impractical for real-world applications. The review highlights the importance of developing computationally efficient re-ranking approaches that effectively improve retrieval performance.

Chapter 3

Deep Image Retrieval Networks using Residual Blocks Focusing on Spatial Information

The performance of deep image retrieval networks significantly depends on the quality of the feature vectors that these networks generate. Deep convolutional neural networks are excellent at feature extraction but may compromise a portion of the spatial information during the convolution and pooling operations. This is because they are designed to be invariant to translations which makes them excellent for classification tasks where the position of the object in the image is irrelevant to some extent. Such loss of spatial information can potentially impact the performance of a

deep image retrieval network, where retention of spatial information, with its ability to provide cues about the location and interrelations of objects within an image, is crucial in obtaining highly discriminative feature vectors. In this chapter, we develop two distinct residual blocks with a focus on incorporating spatial information obtained from different scales and levels of abstraction in order to enhance the representational capacity of deep convolutional neural networks for the task of image retrieval. Two spatial information acquisition techniques, namely multi-scale spatial features and multi-source spatial features, are developed to improve the representational capacity of deep networks for image retrieval [58], [59].

3.1 Improving Deep Features for Image Retrieval using Multi-Source Spatial Information

The representational quality of the generated feature vectors for images is essential for image retrieval models to achieve high performance. Spatial information is crucial in obtaining highly representative feature vectors for image retrieval, and deep convolutional neural networks provide an excellent framework to generate such features. Deep convolutional neural networks include spatial information in the feature maps through convolutional operations. However, most available architectures cannot include adequate spatial details in the feature maps helpful for obtaining high-performance image retrieval. Deep residual networks are deep networks capable of including useful information through residual learning. This section presents a novel residual block to generate feature maps by focusing on spatial information. The proposed residual block comprises three modules: a spatial feature extraction module, a hierarchical feature extraction module, and a feature fusion module. The first module includes spatial information in the feature maps at different levels of abstraction, while the second module includes spatial information using conventional convolution hierarchy. The third model fuses the outputs of the first two modules to provide a very rich set of feature maps.

Figure 3.1 shows the architecture of the proposed residual block. Figure 3.1(a) shows a high-level view of the proposed residual block, which consists of two distinct feature extraction modules, one feature fusion module, and a skip connection. The figure shows that the input feature tensor $\boldsymbol{x}$ is passed through two parallel pathways, each of which extracts specific features from the

input tensor $\boldsymbol{x}$. The upper pathway extracts spatial features, and the lower pathway extracts hierarchical features. The feature maps **V** and **U** resulting from these two modules are given by

$$\begin{aligned} \boldsymbol{V} &= \mathcal{S}(\boldsymbol{x}) \\ \boldsymbol{U} &= \mathcal{H}(\boldsymbol{x}) \end{aligned} \tag{3.1}$$

where $\mathcal{S}(.)$ and $\mathcal{H}(.)$ denote the processes of the spatial feature extraction module and the hierarchical feature extraction module, respectively.

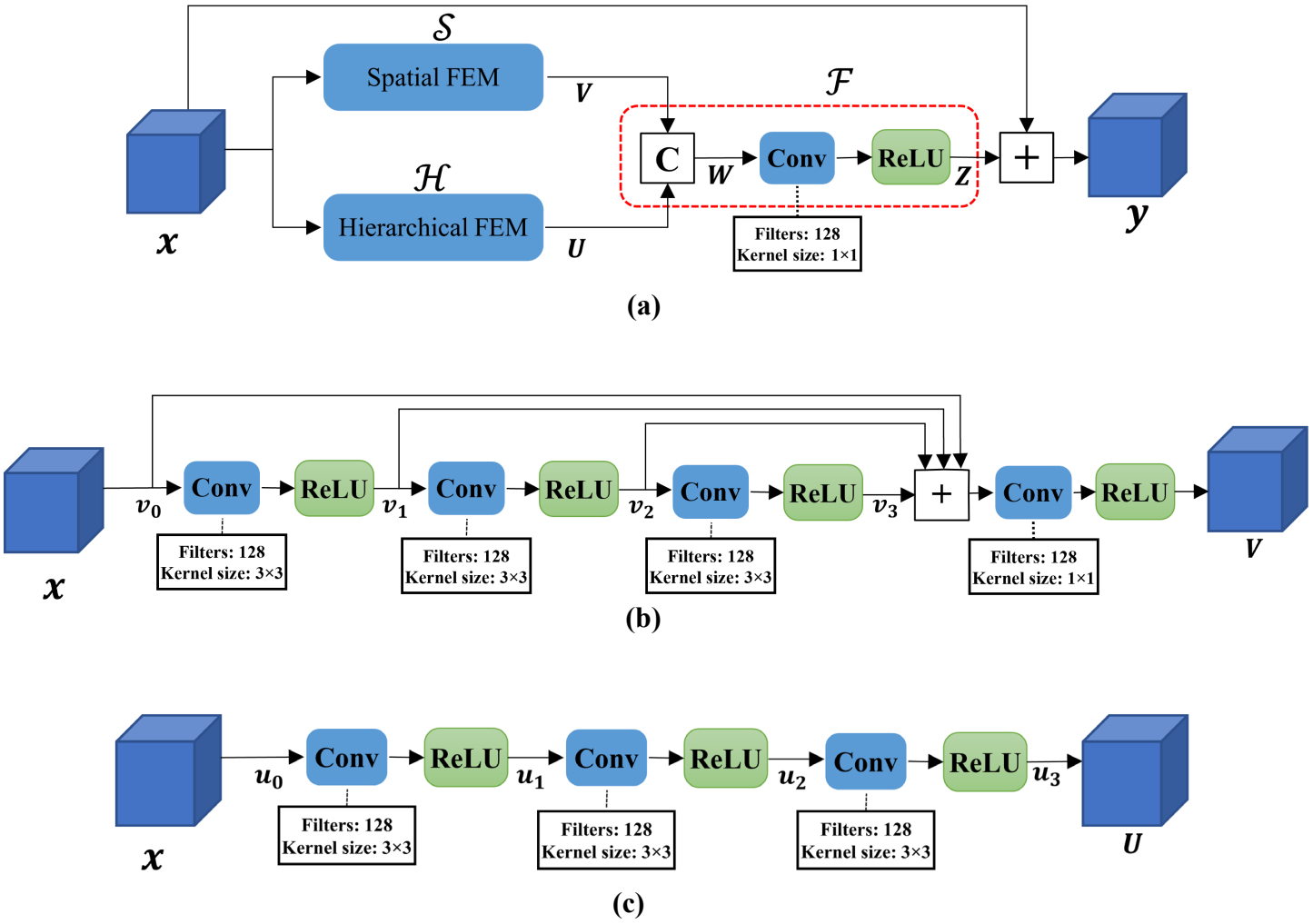

Figure 3.1: Architecture of (a) the proposed residual block, (b) the spatial feature extraction module, and (c) the hierarchical feature extraction module. Conv. denotes the convolution. "C" is a symbol representing concatenation and summation operations. Spatial FEM and Hierarchical FEM denote "spatial feature extraction module" and "hierarchical feature extraction module," respectively.

Figure 3.1(b) shows the architecture of the spatial feature extraction module used to produce feature tensor $\boldsymbol{V}$. In this module, the input feature tensor $\boldsymbol{x}$ undergoes a cascade of three sets of convolutional operations followed by ReLU activation functions to produce feature tensors $\boldsymbol{v}_i$ as

$$\boldsymbol{v}_i = ReLU\big(\mathcal{C}_i(\boldsymbol{v}_{i-1})\big)\ i = 1,2,3 \quad \text{and} \quad \boldsymbol{v}_0 = \boldsymbol{x} \tag{3.2}$$

where $\mathcal{C}_i$ represents the convolution operations employing 128 filters with a kernel size of 3×3. The output feature tensor of this module is generated as

$$\boldsymbol{V} = ReLU\left(\mathcal{C}_4\left(\sum_{i=0}^{3}\boldsymbol{v}_i\right)\right) \tag{3.3}$$

where $\mathcal{C}_4$ is a convolution operation with 128 filters, each with a size of 3×3.

In the hierarchical feature extraction module shown in Figure 3.1(c), the input feature tensor $\boldsymbol{x}$ is fed into a sequence of convolution operations, each of which is followed by a ReLU activation function. The output of this module is the feature tensor $\boldsymbol{U}$, which is given by

$$\boldsymbol{U} = ReLU\left(\mathcal{C}_j\left(\boldsymbol{u}_{j-1}\right)\right)\ j = 1,2,3 \quad \text{and} \quad \boldsymbol{u}_0 = \boldsymbol{x} \tag{3.4}$$

where $\mathcal{C}_j$ denotes the convolution operation with 128 filters with a kernel size of 3×3.

In the feature fusion module shown in Figure 3.1(a), the two feature tensors $\boldsymbol{U}$ and $\boldsymbol{V}$ are concatenated as

$$\boldsymbol{W} = CONC(\boldsymbol{U}, \boldsymbol{V}) \tag{3.5}$$

where $CONC(.)$ is the concatenation operation. The feature tensor $\boldsymbol{W}$ is then passed through a convolution operation, followed by the ReLU activation function, to produce the feature tensor $\boldsymbol{Z}$ as follows:

$$\boldsymbol{Z} = \mathcal{F}(\boldsymbol{U}, \boldsymbol{V}) = ReLU\big(\mathcal{C}(\boldsymbol{W})\big) \tag{3.6}$$

where $\mathcal{F}$ (.) denotes the feature fusion process, and $\mathcal{C}$ is the convolution operation with 128 filters with a kernel size of 3×3. Next, the obtained feature tensor $\boldsymbol{Z}$ is added to the feature tensor $\boldsymbol{x}$ input to the block through a skip connection to produce the output of the block as

$$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{Z} \tag{3.7}$$

The output feature tensor $\boldsymbol{y}$ of the residual block contains a rich set of features enhanced by spatial features and hierarchical features that significantly enhance the representational capacity of the network to improve the retrieval performance of the deep image retrieval network.

The deep network employing the proposed residual block is derived from standard AlexNet [60]. Standard AlexNet contains five convolutional layers and three fully connected layers. The first two convolutional layers are followed by max pooling layers, while the third and fourth convolutional layers are directly connected to the next convolutional layer. We have modified the standard AlexNet to employ our proposed residual block and use it as a feature extractor. The major modifications are listed below:

1. The feature tensors obtained from the fourth convolutional layer are fed to a sequence of eight units of the proposed residual block.
2. Unlike the standard AlexNet, where the last convolutional layer is connected to a max pooling layer, our modified version does not have a pooling layer after the last convolutional layer to ensure the spatial information of the last convolutional layer remains intact.

The network employing the proposed residual block is trained as a classifier on the chosen dataset. Depending on the dataset used for a given experiment, the size of the last fully connected layer is changed to match the number of classes present in the dataset. This fully connected layer later provides the probability for every image passing through the network through a SoftMax activation function. Once the training is done, the top layers, including all the fully connected layers and the SoftMax activation function, are removed, and the network is converted into a feature extractor. Feature extraction is performed by inputting a given image into the network and capturing the network's output as the corresponding feature vector for the image. Unlike most existing deep image retrieval networks, we do not obtain feature vectors from one of the fully connected layers since these layers distort spatial information. We refer to our proposed residual

block as a **M**ulti-source **S**patial **F**eature generating **R**esidual block (MSFR) and the network using the proposed MSFR as MSFRNet.

The network is trained for 200 epochs with a batch size of 32. The sizes of the datasets are artificially expanded using the idea of data augmentation for training purposes to improve the learning performance of the network. The utilized augmentation techniques are random rotation, random zoom, and random brightness methods, using techniques in [61]. The network is optimized using the stochastic gradient descent technique. The learning process starts with a learning rate of 0.01, and the decay is set to 5×10^{-6}. The deep network employing the proposed residual block is implemented using TensorFlow [62] and Keras [63]. An Intel Core i7 @3.2 GHz machine with an Nvidia GeForce GTX 3060 GPU is used to train and test the proposed network. The metric used is mean average precision (mAP), which is the mean value of the average precisions computed for all query images [66].

3.2 Development of a Deep Image Retrieval Network using Hierarchical and Multi-scale Spatial Features

Deep convolutional neural networks provide an excellent tool for obtaining highly representative feature vectors from images which is a crucial aspect in improving the performance of image retrieval methods. Among various available deep architectures, residual networks have shown superior flexibility and performance over other architectures, as they can be designed to incorporate valuable information into the feature vectors through residual learning [57]. Their superiority can be attributed mainly to their ability to embed additional information into the feature vector through residual learning, which can be accomplished by designing suitable operations within the block. One operation is to include spatial information obtained at different scales and levels of abstraction in the deep network. In this section, we develop a novel residual block to include spatial information obtained at different abstraction levels and scales to enhance a deep network's ability to generate very rich sets of features for image retrieval. The proposed residual block consists of three modules: a hierarchical spatial feature extraction module focusing on spatial information at different abstraction levels, a multi-scale feature extraction module that generates features at three different scales, and a feature fusion module.

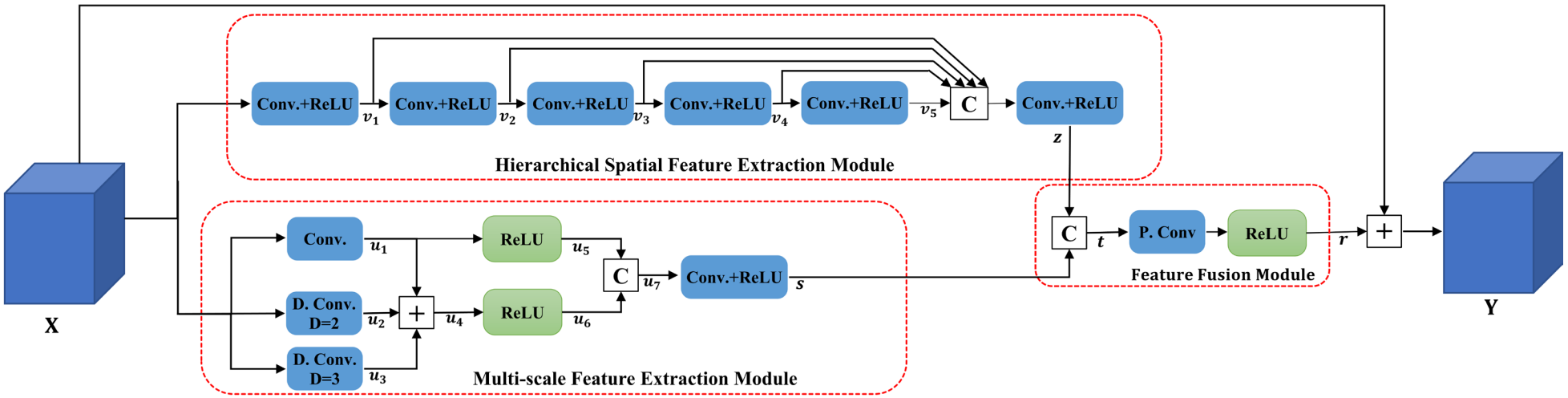

Figure 3.2: Architecture of the proposed residual block. Conv., D.Conv. and P.Conv. represent the convolution, dilated convolution and point-wise convolution operations, respectively.

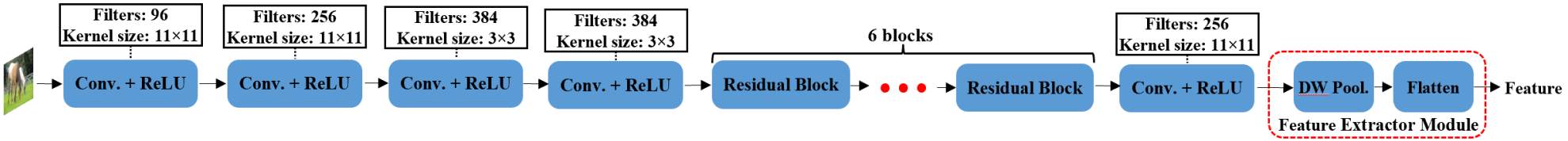

Figure 3.3: Modified variant of AlexNet architecture to employ the proposed residual block. DW Pool. denotes depth-wise pooling. Max pooling layers are not shown.

The design of the proposed residual block is shown in Figure 3.2. As the figure shows, the proposed residual block consists of three modules: a hierarchical spatial feature extraction module, a multi-scale feature extraction module, and a feature fusion module. The input feature tensor $\boldsymbol{X}$ is simultaneously fed into two feature extraction modules. In the hierarchical spatial feature extraction module, the input feature tensor $\boldsymbol{X}$ undergoes a sequence of convolution operations, followed by ReLU activation functions as

$$\boldsymbol{v}_i = ReLU\big(W_i(\boldsymbol{u}_i)\big), i = 1, \dots, 5 \text{ and } \boldsymbol{v}_0 = \boldsymbol{X} \tag{3.8}$$

where W_i represents the convolution operations, each of which has 64 filters with spatial support of 3×3. The outcome of each of $\boldsymbol{v}_i$ incorporates spatial information into the corresponding feature tensor at different levels of abstraction. Concatenating these feature tensors, namely $\boldsymbol{v}_i$, into feature tensor $\boldsymbol{z}$ provides rich feature sets enriched by strong spatial information. The spatial information improves the capability of the residual block to generate a rich set of feature vectors to enhance retrieval performance.

The second module, namely the multi-scale feature extraction module, is designed to extract features based on different scales, as the module contains dilated convolution operations performed on different sizes of receptive fields. This process broadens the view of the convolution operation on the input feature map to obtain different spatial information. In this module, the input feature tensor $\boldsymbol{X}$ undergoes three parallel coevolution operations as follows:

$$\begin{aligned} \boldsymbol{u}_1 &= W_6(\boldsymbol{X}) \\ \boldsymbol{u}_2 &= W_7(\boldsymbol{X}) \\ \boldsymbol{u}_3 &= W_8(\boldsymbol{X}) \end{aligned} \tag{3.9}$$

where W_6 is the convolution operation employing 64 filters with a kernel size of 3×3, and W_7 and W_8 are dilated convolution operations, each with 64 filters with kernel size 3×3 and a dilation rate of 2 and 3, respectively.

However, dilated convolution operations have a gridding artifact that degrades their performance [57],[64]. The negative effect of the artifact is mitigated by adding the feature tensor $\boldsymbol{u}_1$, which is not generated employing a dilated convolution operation, to the other two feature tensors obtained from dilated convolution operations (namely, $\boldsymbol{u}_2$ and $\boldsymbol{u}_3$) to produce feature tensor $\boldsymbol{u}_4$ as

$$\boldsymbol{u}_4 = \underbrace{W_6(\boldsymbol{X})}_{\boldsymbol{u}_1} + \underbrace{W_7(\boldsymbol{X})}_{\boldsymbol{u}_2} + \underbrace{W_8(\boldsymbol{X})}_{\boldsymbol{u}_3} \tag{3.10}$$

Then, the feature tensor $\boldsymbol{u}_7$ is generated by fusing the feature tensors obtained from the application of the ReLU activation function on $\boldsymbol{u}_1$ and $\boldsymbol{u}_4$ as

$$\boldsymbol{u}_7 = Conc\left(\underbrace{ReLU(\boldsymbol{u}_1)}_{\boldsymbol{u}_5} + \underbrace{ReLU(\boldsymbol{u}_4)}_{\boldsymbol{u}_6}\right) \tag{3.11}$$

where $Conc(.)$ is the concatenation operation. The feature tensor $\boldsymbol{u}_7$ is then passed through a convolution operation followed by a ReLU activation function to produce feature tensor $\boldsymbol{s}$.

In the feature fusion module, the feature tensors $\boldsymbol{s}$ and $\boldsymbol{z}$ obtained from the two feature extraction modules are concatenated. Then, the resulting feature maps undergo a convolution operation and a ReLU function to generate a rich set of residual feature maps of the block as

$$\boldsymbol{r} = ReLU\left(W_9\big(Conc(\boldsymbol{s}, \boldsymbol{z})\big)\right) \tag{3.12}$$

where the operation W_9 represents a convolution operation using 64 filters with a kernel of size 3×3. The residual feature tensor $\boldsymbol{r}$ is added to the feature tensor $\boldsymbol{X}$ input into the block to produce the block's output $\boldsymbol{Y}$ as

$$\boldsymbol{Y} = \boldsymbol{X} + \boldsymbol{r} \tag{3.13}$$

We refer to the proposed residual block as the **H**ierarchical and **M**ulti-scale **S**patial feature generating **R**esidual block (HMSR), and the network employing the proposed block is called HMSRNet.

The network architecture of the proposed deep image retrieval network is shown in Figure 3.3. The backbone of the deep image retrieval network employing the proposed residua block is AlexNet [60], with some modifications. The original architecture of the AlexNet consists of five convolutional layers and three fully connected layers. The first two layers contain a sequence of two sets of convolution operations followed by a ReLU activation function. The following (third and fourth) convolutional layers are connected to their succeeding convolutional layers without interleaving by any activation functions. In the original AlexNet, the following (fifth) layer contains a convolution layer and a max pooling layer. However, in our proposed HMSRNet, the output of the fourth convolutional layer is cascaded with six units of the proposed residual block, namely HMSR, followed by the fifth convolution layer, which is directly connected to the first layer of a group of three fully connected layers. A SoftMax activation function follows the fully connected layers to train the model as an image classifier.

Once the network is trained, the fully connected layers and the SoftMax activation function are removed and replaced by the feature extraction module highlighted in Figure 3.3 without further refining the weights. The feature extraction module contains a depth-wise convolution operation and a flattening layer. The output of the feature extraction module for a given image is the feature vector of that image, which is used for retrieval.

The network is trained for up to 200 epochs. The best-performing network configuration is then chosen and employed for retrieval tasks. The size of the batch utilized during the training process is 64. Optimization is carried out using the stochastic gradient descent technique. The learning process starts with a learning rate of 0.01, and it is decayed by the learning rate divided by the number of epochs. The proposed HMSRNet is trained and tested on an Intel Core i7 @3.2 GHz machine with an Nvidia GeForce GTX 3060-12GB GPU. The implementation is done using the Keras library [63] and the TensorFlow package [62].

Table 3.1: Comparison between various image retrieval networks.

Network	Backbone	Dataset				FLOPs*
		Cifar10	Cinic10	Cifar100	Animals	
CSCFM [65]	ResNeXt-50	0.8351	0.7143	**0.8351**	**0.6087**	>15.3
BIMCNN [66]	VGG16	**0.8382**	0.7343	**0.8295**	–	>15.3
DELG [67]	ResNet101	**0.8334**	**0.7482**	0.8211	**0.6351**	>7.6
MRDL [68]	VGG19	0.7782	0.7336	0.7556	0.6036	>19.6
ADFSDH [69]	VGG16+VGG19	0.7933	**0.7372**	0.7743	–	>27.2
UDPH [70]	VGG16	0.7754	0.5990	0.7598	0.4235	>15.3
SIRS-IR [71]	Inception-ResNet-V2	0.6945	0.7343	0.6627	0.5832	>17.0
MSFRNet [58]	AlexNet	**0.8431**	**0.7487**	**0.8348**	**0.6346**	9.3

BOLD, ***ITALIC***, and **underlined** fonts indicate the best, second-best, and third-best performance, respectively.
* FLOPs refer to floating point operations on a scale of billions.

3.3 Experimental Results

3.3.1 Experimental Results of MSFRNet

Table 3.1 compares the results obtained from the proposed MSFRNet [58] with those obtained from state-of-the-art deep image retrieval networks. As the table shows, our proposed MSFRNet demonstrates the highest or second-highest accuracy for all the datasets, even though it uses AlexNet as a relatively lightweight deep network, whereas other methods utilize complex and heavy networks. The performance of the proposed MSFRNet demonstrates that the proposed residual block improves the performance of AlexNet such that it is comparable with the performance of other (much more complex) networks.

Table 3.2: Comparison of the results obtained by the network depending on whether the proposed residual block is used.

Dataset	With	Without	Reduction
Cifar100	0.8348	0.7918	5.4%
Cinic10	0.7487	0.7058	6.0%
Animals	0.6346	0.6047	15.2%

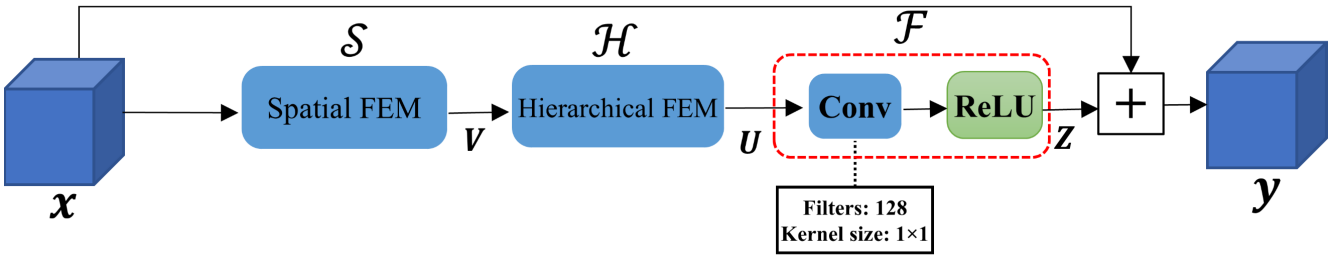

Figure 3.4: Design of the proposed residual block using a serial scheme.

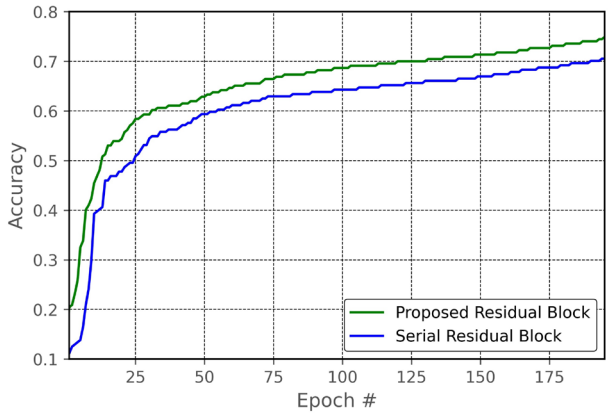

Figure 3.5: Learning curves for two design schemes of the proposed residual block on Animals dataset.

Table 3.3: mAP values for the serial configuration of the residual block and the proposed residual block.

Dataset	Proposed	Serial Scheme	% Reduction
Cifar100	0.8348	0.7445	12.1
Cinic10	0.7487	0.6501	15.2
Animals	0.6346	0.5565	14.0

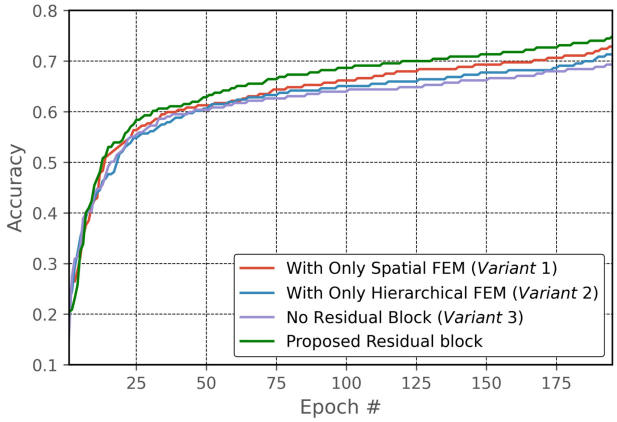

Figure 3.6: Learning curves for the proposed residual block and its variants on Animals dataset.

Table 3.4: mAP values for the proposed block and its variants.

Modules	Variant 1	Variant 2	Variant 3	Proposed
Hierarchical Feature Extraction	-	✓	-	✓
Spatial Feature Extraction	✓	-	-	✓
mAP (Cifar100)	0.8045	0.7761	0.7918	**0.8348**
mAP (Cinic10)	0.7299	0.7087	0.7058	**0.7487**
mAP (Animals)	0.6243	0.6032	0.6047	**0.6346**

The values in the **BOLD** fonts indicate the best.

We now investigate the effectiveness of the proposed residual block by excluding it from the MSFRNet, thus degenerating it to the standard AlexNet. The results are shown in Table 3.2. The table clearly shows that the modified AlexNet employing the proposed residual block performs significantly better than the standard AlexNet for all datasets.

We now study the design of the proposed residual block. As shown in Figure 3.1(a), the proposed residual block contains two modules in a parallel mode. Unlike a serial scheme, this parallel arrangement enables the modules to extract features based on their design without mutual distortion if the feature tensor $\boldsymbol{x}$ is input into two modules simultaneously. We carry out an experiment to show the superiority of our proposed method over the serial scheme. The design of the serial configuration of the proposed residual block model is shown in Figure 3.4. The serial residual

block has a major change compared to the proposed residual block. In the block shown in Figure 3.1(a), both modules use the input feature tensor $\boldsymbol{x}$. Differently, in the serial residual block shown in Figure 3.4, the input of the hierarchical feature extraction module is the feature tensor $\boldsymbol{V}$ which is provided by the spatial feature extraction module. The learning curves are shown in Figure 3.5. As Figure 3.5 shows, the network employing the serial residual block has significantly lower learning quality than the proposed residual block. Table 3.3 shows the mAP of the networks using the proposed residual block and using the serial configuration of the block. The table clearly shows that the parallel configuration (the proposed method) is superior to the serial configuration in terms of mAP values.

We now investigate the impact of each of the two modules of the proposed residual block—namely, the hierarchical feature extraction module and edge feature extraction module—on the network performance individually. Three variants of the proposed residual block are formed; Variant 1, Variant 2, and Variant 3. Each of these variants is formed by individually employing either module or removing the entire residual block (Variant 3). The learning curves for the proposed deep image retrieval network with its variants are shown in Figure 3.6. As seen from the figure, the learning curve for the network employing the proposed residual block shows a consistent, favorable learning curve. Furthermore, the curves are close for those variants when only a hierarchical feature extraction module is used and no residual block is employed. The slight improvement in the block with only a hierarchical feature extraction module is because of the extra convolution operations added to the network by the module. However, the network employing the residual block with only the spatial feature extraction module shows an apparent improvement gain compared to Variants 2 and 3. Results demonstrate that spatial information is the key to improving the retrieval performance of a deep image retrieval network. A quantitative comparison between the proposed residual block and its variants is shown in Table 3.4. The table confirms the proposed network's superior retrieval performance for all the datasets.

3.3.2 Experimental Results of HMSRNet

The performance of the proposed network is compared with that of several other conventional and state-of-the-art deep image retrieval networks. The results are given in Table 3.5. The table shows that HMSRNet [59] performs best in three of the four datasets while maintaining much lower

complexity than the others. A noteworthy observation is that the proposed HMSR residual block allows a comparatively lightweight network like AlexNet to be comparable or superior to methods that employ much more complex networks.

We now compare HMSRNet with our previously presented work, namely MSFRNet. Table 3.5 clearly demonstrates the significant superiority of HMSRNet. A key reason for this superiority is the utilization of dilated convolutional operations in the multi-scale module used in the proposed residual block. Unlike traditional convolutional operations, in which their receptive fields are limited to only capture information from immediate, neighboring pixels, utilizing dilated convolution operations in HMSRNet enables it to generate a richer set of features, thereby significantly enhancing the retrieval performance.

Table 3.5: Comparison between various image retrieval networks in terms of mAP.

Network	Backbone	Dataset				FLOPs*
		Cifar10	Cinic10	Cifar100	Animals	
CSCFM [65]	ResNeXt-50	0.8351	0.7143	**0.8351**	0.6087	>15.3
BIMCNN [66]	VGG16	**0.8382**	0.7343	0.8295	–	>15.3
DELG [67]	ResNet101	0.8334	**0.7482**	0.8211	**0.6351**	>7.6
MRDL [68]	VGG19	0.7782	0.7336	0.7556	0.6036	>19.6
ADFSDH [69]	VGG16+VGG19	0.7933	0.7372	0.7743	–	>27.2
UDPH [70]	VGG16	0.7754	0.5990	0.7598	0.4235	>15.3
SIRS-IR [71]	Inception-ResNet-V2	0.6945	0.7343	0.6627	0.5832	>17.0
MSFRNet [58]	AlexNet	**0.8431**	**0.7487**	**0.8348**	**0.6346**	9.3
HMSRNet [59]	AlexNet	**0.8543**	**0.7390**	**0.8507**	**0.6405**	9.3

BOLD, ***ITALIC***, and **underlined** fonts indicate the best, second-best, and third-best performance, respectively.
*FLOPs refer to floating point operations, shown here in billions (GFLOPs).

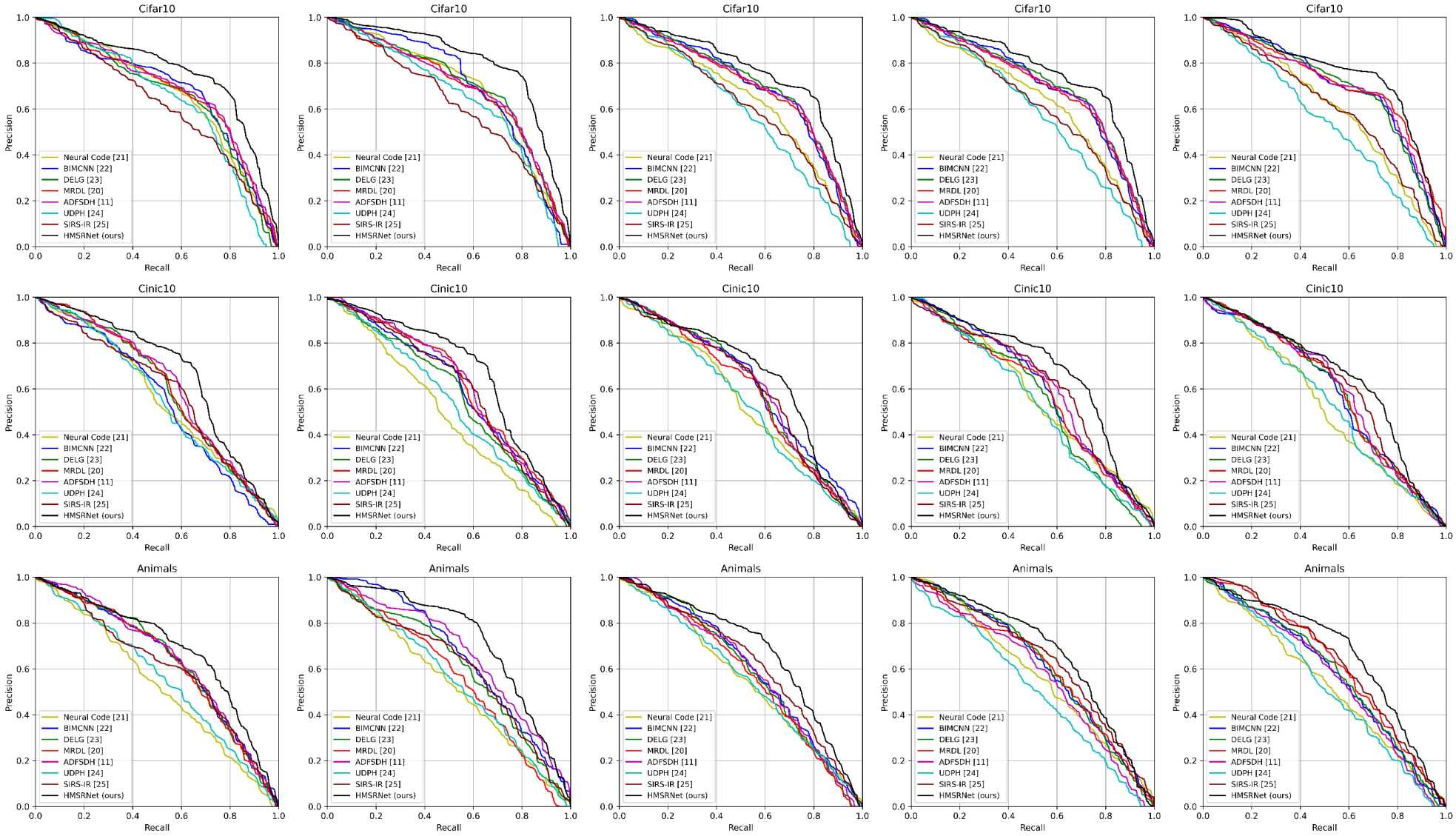

Figure 3.7: Precision-recall curves obtained from the proposed HMSRNet on Cifar10, Cinic10, and Animals datasets.

Table 3.6: Performance comparison of the proposed method.

Dataset	With	Without	% Reduction
Cifar10	0.8643	0.7718	6.46
Cinic10	0.7382	0.6701	2.51
Animals	0.6365	0.6031	5.53

Table 3.7: Results of ablation study.

Modules	Variant 1	Variant 2	Variant 3	HMSRNet
Hierarchical Spatial Feature Extraction	✓	-	-	✓
Multi-scale Feature Extraction	-	✓	-	✓
mAP (Cifar10)	0.8145	0.7801	0.7718	**0.8643**
mAP (Cinic10)	0.7283	0.6984	0.6701	**0.7382**
mAP (Animals)	0.6123	0.5967	0.6031	**0.6365**

The values in the **red** fonts indicate the best.

To further investigate the effectiveness of the proposed residual block, we exclude the residual block and re-train the network before observing the performance. The results are summarized in Table 3.6. As the table indicates, removing the proposed residual block from the retrieval network reduces the retrieval performance by at least 2.51% for Cinic10. The performance degradation is more severe for the network tested without the residual block on Cifar10 and Animals datasets, which exhibited performance reductions of 6.46% and 5.53%, respectively.

Figure 3.7 presents examples of precision-recall curves generated based on the testing results of our proposed retrieval network and several other networks on three datasets. As the figure shows, HMSRNet demonstrates better retrieval performance than the other methods for all datasets.

We now study the impact of two modules of the proposed residual block—i.e., the hierarchical spatial feature extraction module and multi-scale feature extraction module—on the performance by forming three distinct variants. *Variant 1* is formed by including only the spatial feature

extraction module. *Variant 2* includes only the multi-scale feature module. Finally, *Variant 3* is formed by removing the entire residual block from the network. The results using the proposed residual block and its three variants are summarized in Table 3.7. it is seen from the table that removing either module from the network significantly degrades the retrieval performance. The degradation in performance intensifies when the spatial feature extraction module is removed from the network (*Variant 2*). This outcome is not surprising, as spatial information is the most important information for locating the objects in the images in order to match the content of the images. As this module incorporates spatial information from different levels of abstraction in the feature maps, excluding such information significantly reduces the representational capacity of the feature vectors generated by the block. Meanwhile, the multi-scale feature extraction module produces more general features from the input feature maps than the others. *Variant 3* is the worst-performing variant, as this network relies on the non-residual network; the feature maps produced by such a network have less spatial information than networks employing the proposed residual block.

Figure 3.8 shows the learning curves for the above-discussed variants of the proposed residual blocks for 200 epochs on the Animals dataset. The learning quality of the network employing the proposed residual block is higher than that of the network employing either module individually. Therefore, the proposed block is able to enhance the representational capacity of the feature vectors.

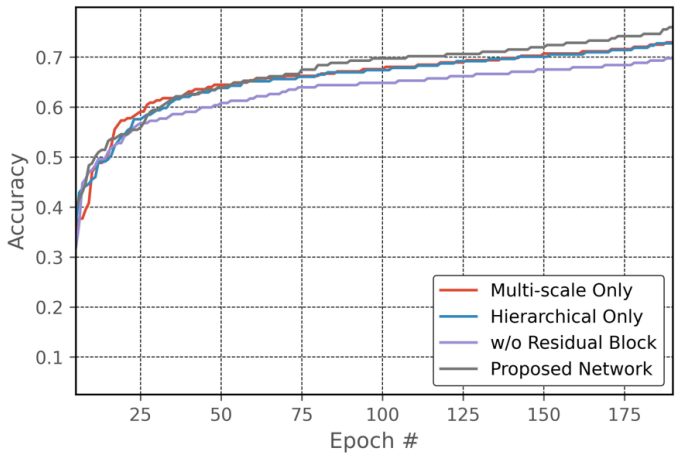

Figure 3.8: Learning curves of the proposed network and the variants on Animals dataset.

3.4 Summary

In this chapter, we have proposed two novel techniques aiming to guide deep networks to generate rich feature sets by extracting spatial information. Specifically, we have used two distinct techniques to extract spatial information, namely, the multi-source spatial information extraction feature technique and the multi-scale feature extraction technique. We have developed two deep convolutional neural networks capable of extracting features with a focus on spatial information. The extensive experimental results have confirmed that the new designs of residual blocks focusing on spatial information result in high-performance, low-complexity deep networks for the task of image retrieval.

Chapter 4

Deep Image Retrieval Network with Guided Feature Generation

Textural, structural, and edge information is very important for differentiating images. As such, a network designed to capture these specific attributes could significantly enhance the effectiveness of image retrieval tasks. In [55], we have shown that the textural and structural information derived from images significantly contributes to the generation of highly discriminative features. The work presented in [55] highlights the potential of morphological operations, which are capable of providing such information effectively to a deep network. Building upon this foundational work, this chapter proposes a novel residual block that utilizes morphological operations and a new pooling operation that emphasizes edge features. The aim is to design a deep convolutional neural network capable of generating rich sets of features for image retrieval tasks [72].

4.1 Max-m-Min Pooling

Let us define a pooling operation, referred to as *maximum minus minimum pooling* (*Max-m-Min*), as

$$o_c[m,n] = \underset{x,y \in \mathcal{N}_{a\times b}^{(m,n)}}{\mathbf{max}} i_c[x,y] - \underset{x,y \in \mathcal{N}_{a\times b}^{(m,n)}}{\mathbf{min}} i_c[x,y] \tag{4.1}$$

In order to illustrate this pooling operation, for the sake of simplicity, we consider a pooling window of size of 2×2 with its set of indices given by $\mathcal{N}_{2\times 2}^{(m,n)}$. The output signal $o_c[m,n]$ obtained by applying *Max-m-Min* pooling operation to the input signal $i_c[m,n]$ is

$$o_c[m,n] = \underset{x,y \in \mathcal{N}_{2\times 2}^{(m,n)}}{\mathbf{max}} i_c[x,y] - \underset{x,y \in \mathcal{N}_{2\times 2}^{(m,n)}}{\mathbf{min}} i_c[x,y] \tag{4.2}$$

Let the values of the features in the window corresponding to these indices be $i_c\,[m,n] = \{a_1, a_2, a_3, a_4\}$, as shown in Figure 4.1 (a). The minimum (min) and maximum (max) values in the pooling window required to calculate $o_c[m,n]$ can be found in four possible locations, as shown in Figure 4.1 (b)-(e). Taking Figure 4.1(b) as an example (in which the maximum value is a_1), the three possibilities presented below arise:

$$o_c[m,n] = \begin{cases} G_1 = a_1 - a_2 & if \quad min = a_2 \\ G_2 = a_1 - a_3 & if \quad min = a_3 \\ G_3 = a_1 - a_4 & if \quad min = a_4 \end{cases} \tag{4.3}$$

where G_1, G_2, and G_3 represent gradient approximations of two adjacent pixels located at indices represented by $\mathcal{N}_{2\times 2}^{(m,n)}$. The above discussion can be easily generalized to the other three scenarios depicted in Figure 4.1(c)-(e). Note that a pooling window in an image represents a very small neighborhood of the image in which, generally a pair of adjacent pixels does not have a significant difference in their pixel values unless the pixel (m,n) is an edge pixel. Therefore, after applying the proposed *Max-m-Min* pooling operation, most of the pixel values in the resulting image will

have very small values, and only the pixel positions on the image edge will have large pixel values. Therefore, by applying the proposed pooling operation, one can expect the resulting image to represent an edge map of the image. It is because of this reason; one can also expect to improve the representational capacity of the network if the proposed pooling operation is utilized.

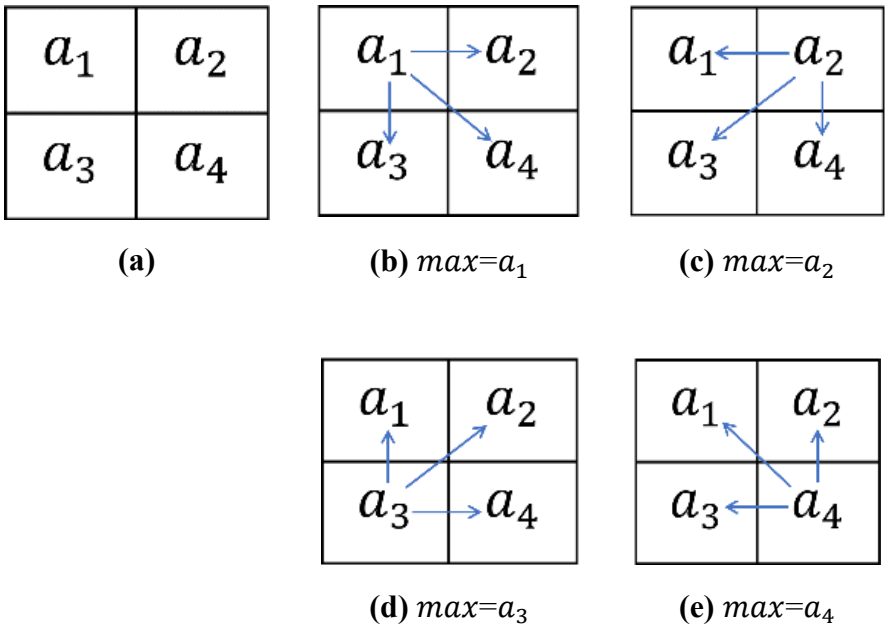

Figure 4.1: (**a**) An example illustrating a pooling window of size 2×2. (**b**)-(**e**) represent four templates of the possible locations of minimum and maximum values required for *Max-m-Min* pooling on this window.

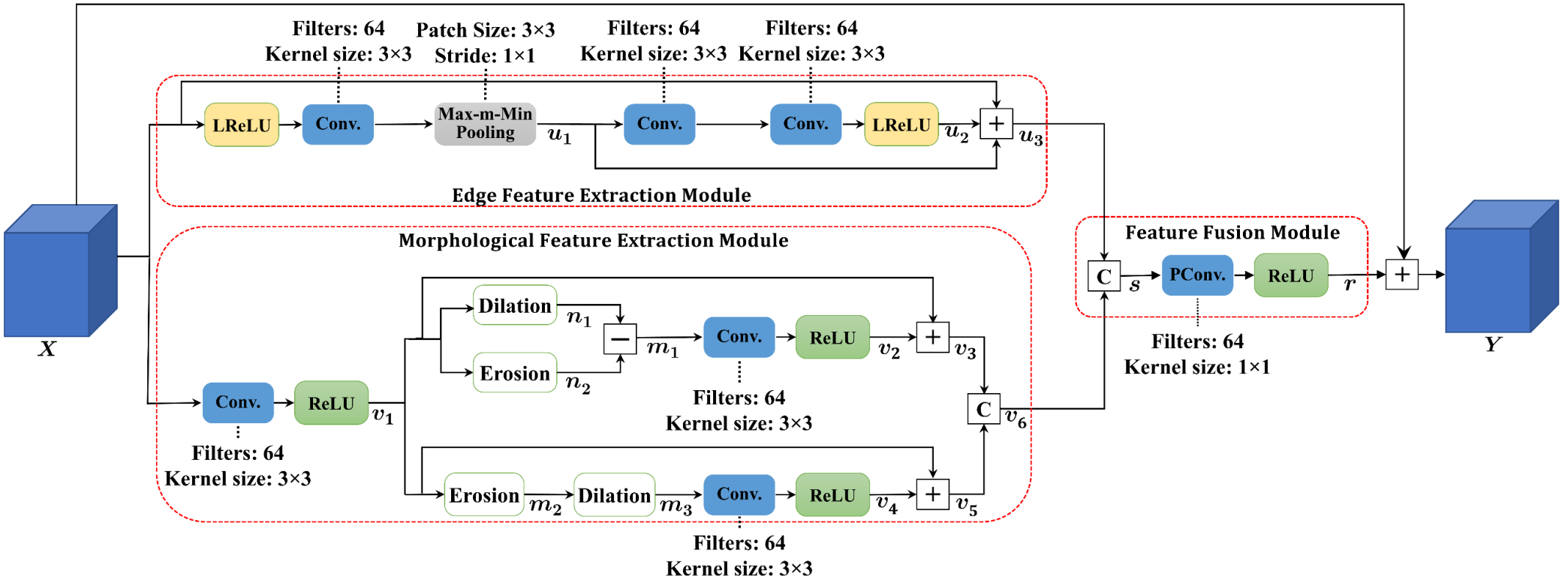

Figure 4.2: Detailed architecture of the proposed residual block. LReLU, Conv., and PConv. are Leaky-ReLU, convolution, and pointwise convolution, respectively. The symbols '+' and '-' represent tensor addition and tensor subtraction, respectively. The symbol 'c' represents the concatenation operation.

4.2 Proposed Residual Block

Figure 4.2 shows the architecture of the proposed residual block, which consists of three modules: an edge feature extraction module, a morphological feature extraction module, and a feature fusion module. In the edge feature extraction module, the feature tensor $\boldsymbol{X}$ input into the proposed residual block undergoes an edge feature extraction operation to obtain feature maps that consist of high-frequency components. The edge feature extraction module consists of two groups of convolution operations, which are interleaved with one *Max-m-Min* pooling layer and are encapsulated by two Leaky-ReLU (LReLU) activation functions. Specifically, the feature tensor obtained from the *Max-m-Min* pooling operation (i.e., $\boldsymbol{u}_1$) is calculated as

$$\boldsymbol{u}_1 = \textit{Max-m-Min}\Big(Conv\big(LReLU(\boldsymbol{X})\big)\Big) \tag{4.4}$$

where $Conv$ is a convolution operation with 64 filters and a kernel size of 3×3. Then, $\boldsymbol{u}_1$ undergoes two convolution operations, followed by LReLU activation to obtain the feature tensor $\boldsymbol{u}_2$ as

$$\boldsymbol{u}_2 = LReLU\left(Conv\big(Conv(\boldsymbol{u}_1)\big)\right) \tag{4.5}$$

Finally, the residual feature tensors $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ are added to the input tensor $\boldsymbol{X}$ to produce the edge feature extraction module's output $\boldsymbol{u}_3$ given by

$$\boldsymbol{u}_3 = \boldsymbol{u}_1 + \boldsymbol{u}_2 + \boldsymbol{X} \tag{4.6}$$

In the edge feature extraction module, feature maps are learned solely through conventional convolution operations with the assistance of the proposed *Max-m-Min* pooling operation. Similarly, in the morphological feature extraction module, feature vectors are learned through convolution operations but with the guidance of nonlinear morphological operations. This guidance provides the network with rich textural and structural information; thus, the network using this residual block would learn a richer set of features.

In the morphological feature extraction module, the feature tensor $\boldsymbol{X}$ first undergoes convolution operation to produce feature map $\boldsymbol{v_1}$ as

$$\boldsymbol{v}_1 = ReLU\big(Conv(\boldsymbol{X})\big) \tag{4.7}$$

The feature tensor $\boldsymbol{v}_1$ is then processed in parallel through two branches that consist of morphological operations to produce morphologically guided features $\boldsymbol{m}_1$ and $\boldsymbol{m}_3$. Specifically, let $\boldsymbol{v}_1^k$ represent the k^{th} channel of the tensor $\boldsymbol{v}_1$, respectively. In the upper branch, the k^{th} channel of the feature tensor $\boldsymbol{n}_1$ resulting from the dilation operation is given by

$$\boldsymbol{n}_1^k[m,n] = \big(\boldsymbol{v}_1^k \oplus b\big)[m,n] = \max_{(x,y)\epsilon \boldsymbol{B}} \boldsymbol{v}_1^k[m+x, n+y] \tag{4.8}$$

where b is the structuring element defined over a neighborhood $\boldsymbol{B}$ of size 2×2 around $[m,n]$ and $\oplus$ is the dilation operation. The k^{th} channel of the feature tensor $\boldsymbol{n}_2$ resulting from the erosion operation over the same neighborhood $\boldsymbol{B}$ is obtained as

$$\boldsymbol{n}_2^k[m,n] = \big(\boldsymbol{v}_1^k \ominus b\big)[m,n] = \min_{(m,n)\epsilon \boldsymbol{B}} \boldsymbol{v}_1^k[m+x, n+y] \tag{4.9}$$

where $\boldsymbol{v}_1^k[m,n]$ is the two-dimensional signal representing the k^{th} channel of tensor $\boldsymbol{v}_1$ at pixel position $[m,n]$ and $\ominus$ is the erosion operation. The k^{th} channel of the output feature tensor $\boldsymbol{m}_1$ is the difference between the feature tensors $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ (so-called morphological gradient operation), and is given by

$$\boldsymbol{m}_1^k[m,n] = \boldsymbol{n}_1^k[m,n] - \boldsymbol{n}_2^k[m,n] \tag{4.10}$$

To learn the features of $\boldsymbol{m}_1$, a convolution operation followed by a ReLU activation function is carried out on the feature tensor $\boldsymbol{m}_1$, yielding $\boldsymbol{v}_2$ given by

$$\boldsymbol{v}_2 = ReLU\big(Conv(\boldsymbol{m}_1)\big). \tag{4.11}$$

The feature tensor $\boldsymbol{v}_2$ is then added to the input feature tensor $\boldsymbol{v}_1$ to obtain the first set of residual feature maps $\boldsymbol{v}_3$ as

$$\boldsymbol{v}_3 = \boldsymbol{v}_1 + \boldsymbol{v}_2 \tag{4.12}$$

In the lower branch, the feature tensor m_3 is obtained by letting the feature tensor $\boldsymbol{v}_1$ to undergo the erosion operation followed by the dilation operation. The tensor $\boldsymbol{m}_3$ is given by

$$\boldsymbol{m}_3^k[m,n] = \big(\boldsymbol{m}_2^k \oplus b\big)[m,n] = \max_{(x,y)\epsilon \boldsymbol{B}} \boldsymbol{m}_2^k[m+x, n+y] \tag{4.13}$$

where $\boldsymbol{m}_2$ is given by

$$\boldsymbol{m}_2^k[m,n] = \big(\boldsymbol{v}_1^k \ominus b\big)[m,n] = \min_{(x,y)\epsilon \boldsymbol{B}} \boldsymbol{v}_1^k[m+x, n+y) \tag{4.14}$$

The tensor feature $\boldsymbol{m}_3$ is passed through a convolution operation and a ReLU activation to yield $\boldsymbol{v}_4$ as

$$\boldsymbol{v}_4 = ReLU\big(Conv(\boldsymbol{m}_3)\big) \tag{4.15}$$

The feature tensor $\boldsymbol{v}_4$ is then added to the input feature tensor $\boldsymbol{v}$ to generate the output feature tensor $\boldsymbol{v}_5$ of the lower branch as

$$\boldsymbol{v}_5 = \boldsymbol{v}_1 + \boldsymbol{v}_4 \tag{4.16}$$

The feature tensors $\boldsymbol{v}_3$ and $\boldsymbol{v}_5$ obtained from the two morphological branches are concatenatively fused to obtain the final output $\boldsymbol{v}_6$ of the morphological feature extraction module as

$$\boldsymbol{v}_6 = CONC(\boldsymbol{v}_3, \boldsymbol{v}_5) \tag{4.17}$$

where *CONC* is the concatenation operation. The feature tensor $\boldsymbol{v}_6$ is enriched by the strong textural and structural information provided by the morphological operations.

In the feature fusion module, the feature tensor $\boldsymbol{u}_3$ obtained from the edge feature extraction module and that obtained from the morphological feature extraction module, namely, $\boldsymbol{v}_6$, are concatenated, resulting in the feature tensor $\boldsymbol{s}$ given by

$$\boldsymbol{s} = CONC(\boldsymbol{u}_3, \boldsymbol{v}_6) \tag{4.18}$$

The feature tensor $\boldsymbol{s}$ then undergoes a pointwise convolution operation, followed by ReLU, to produce the residual feature tensor $\boldsymbol{r}$ as

$$\boldsymbol{r} = ReLU\big(\mathrm{PConv}(\boldsymbol{s})\big) \tag{4.19}$$

where PConv represents the pointwise convolution operation. Finally, the residual feature tensor $\boldsymbol{r}$ is added to the input feature tensor $\mathbf{X}$ to produce the output of the residual block, $\boldsymbol{Y}$, as

$$\boldsymbol{Y} = \boldsymbol{r} + \boldsymbol{X} \tag{4.20}$$

The features produced by the residual block are greatly enhanced by the feature extraction capability of the edge feature extraction module employing the proposed pooling operation as well as by the capability of the morphological feature extraction module in extracting textural and structural information guided by the morphological operations. The richness of the feature tensor $\boldsymbol{Y}$ enables the network to learn highly representational and discriminative feature maps that are critical for high-performance image retrieval.

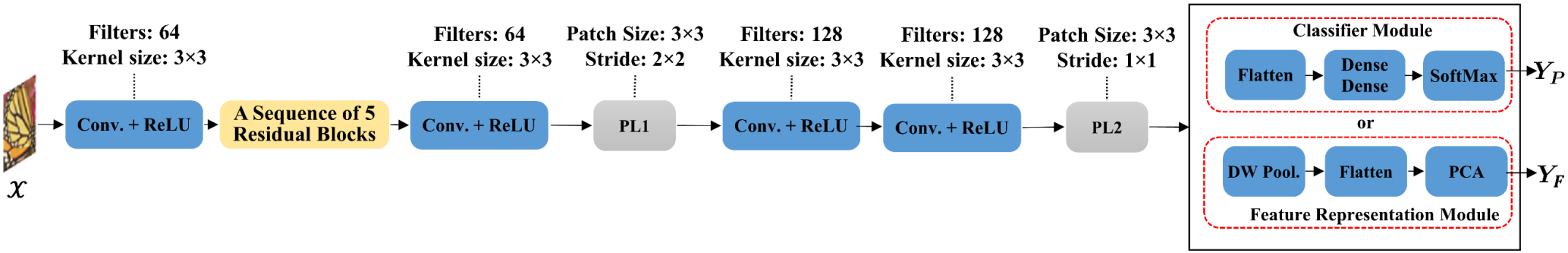

Figure 4.3: Architecture of the proposed image retrieval network. Conv., PL1, PL2, and DW Pool., respectively, denote the feature extraction module, convolution operation, first pooling layer, second pooling layer, and depth-wise pooling layer.

Figure 4.3 shows the overall architecture of the proposed deep image retrieval network referred to as MoFNet. The input to the network is images of size 224×224 denoted as $\boldsymbol{\mathcal{X}}$. Each image $\boldsymbol{\mathcal{X}}$ is passed through the first convolutional layer of the architecture, which consists of 64 filters, each of kernel size 3×3, followed by a ReLU activation function. Then, the resulting feature maps go through a sequence of five residual blocks, proposed in Section 4.2. The feature maps resulting from the sequence of residual blocks are passed through a convolution layer consisting of 64 filters, each of size 3×3, followed by a ReLU activation function. Next, the feature maps obtained from this convolutional layer are fed to the first *Max-m-Min* pooling layer denoted by PL1. The feature maps resulting from PL1 go through a sequence of two convolution operations, each followed by a ReLU activation function, and then the resulting feature maps are passed through another *Max-m-Min* pooling layer denoted by PL2.

The network architecture described so far is used for training as well as for testing. For the purpose of training, the network described so far is supplemented by a classifier module, as shown in Figure 4.3. The output of the second pooling layer, PL2, is fed to the classifier module, which consists of a flattening layer, two dense layers, and a SoftMax layer. The output of the SoftMax layer is a probability vector denoted by Y_P for each class of the image dataset, where each element of the output vector represents the probability of the input belonging to that class.

Once the network has been trained, that classifier module is replaced by the module marked as the feature representation module shown in Figure 4.3. During the testing phase, the output of the pooling layer, the feature representation module, uses the output of the second pooling layer, PL2, followed by a depth-wise pooling layer to form a feature vector. The feature vector is then enhanced using the principal component analysis to obtain the final feature vector Y_F of size 512 representing the features of the input image $\mathcal{X}$.

We refer to the proposed residual block as **Mo**rphological **F**eature-generating residual block (MoF) and the image retrieval network architecture employing the proposed MoF as MoFNet.

For training and testing the proposed retrieval network, six datasets Cifar10 [73], Cinic10 [74], Animals [75], MS-COCO [76], ImageNet [77], and YFCC100M [78] are used. The first three datasets are divided into training and testing sets with a ratio of 4:1. The images in the training set undergo augmentation by random rotation and distortion of the aspect ratio. From each category in

the testing set, five images are randomly selected and used as a set of query images, and the rest of the test images form a set, called search set, from which output images are selected corresponding to a given query image. The COCO dataset is split into a training set consisting of 118K images, a validation set with 5K images, and a test set with 41K images. For this dataset, following the protocol similar to that described in [79], we use the validation set as the query set and the images in the test set as the search set. For the ImageNet, following [79], [80], we randomly select 100 classes, use 100 images from each of these classes are used as the search set, use all the images in the validation set as the query set, and the rest of the images in these classes is used as the training set. For the YFCC dataset, we randomly select 100 classes. Since YFCC is very unbalanced, the selected classes are checked to ensure that the number of images is evenly distributed in each of the selected classes. From each class, we select 500 images as the query set and split the remaining images into search and training sets with a ratio of 4:1. Since YFCC contains a massive number of images, the proposed MoFNet is first trained on the ImageNet dataset and is then fine-tuned on the YFCC dataset. Table 4.1 presents the detailed allocation of images for various datasets, systematically divided into training, query, and search sets.

The proposed network is optimized by training it using the stochastic gradient descent method with a momentum of 0.9. The learning process starts with a learning rate of 0.01, and after each epoch, the learning rate is decayed by a factor of 5×10^{-6}. We use a batch size of 64. The network is trained on a machine with an Nvidia GeForce 3060 12 GB GPU, an Intel Core i7-6700 CPU @3.4 GHz, and 16 GB RAM for up to 200 epochs. The proposed method is implemented using the TensorFlow framework [62] and Keras library [63].

Table 4.1: Distribution of images for different datasets.

Dataset	Number of images in training set	Number of images in query set	Number of images in search set
Cifar10	48k	50	11k
Animals	1.8k	70	370
Cinic10	216k	50	53k
COCO	118k	5k	41k
ImageNet	100k	10k	50k
YFCC	16000k	50k	3950k

4.3 Experimental Results

As discussed in Section 4.2, our proposed residual block consists of two feature extraction modules, namely, the edge feature extraction module and the morphological feature extraction module. To investigate the impact of each of these modules, three variants of the proposed residual block are formed. *Variant 1* is formed by removing both modules. *Variant 2* is formed by removing the edge feature extraction module from the residual blocks, whereas *Variant 3* is formed by removing the edge feature extraction module from the residual blocks.

Table 4.2 gives the results in terms of the mAP of the proposed residual block and its three variants. It is seen from the table that removing the morphological feature extraction module results in noticeable performance degradation (*Variant 3*). Removing the edge feature extraction module (*Variant 2*) impacts the results, although not as strongly as excluding the morphological module. The reason is that the morphologically guided features still contain some amount of edge information, and therefore, the richness of the features is not heavily impacted. However, it is clear that the highest performance is obtained by employing both modules.

Table 4.2: The network performance in terms of mAP of the proposed network and its variants.

Dataset	mAP			
	Variant 1	Variant 2	Variant 3	Proposed
Cifar10	0.8201	0.8681	0.8443	**0.9170**
Animals	0.7273	0.7684	0.7518	**0.8544**
Cinic10	0.7298	0.7803	0.7657	**0.8231**
ImageNet	0.5549	0.6470	0.5994	**0.6955**
COCO	0.6282	0.7310	0.6934	**0.7833**

The best results are shown in **BOLD** font.

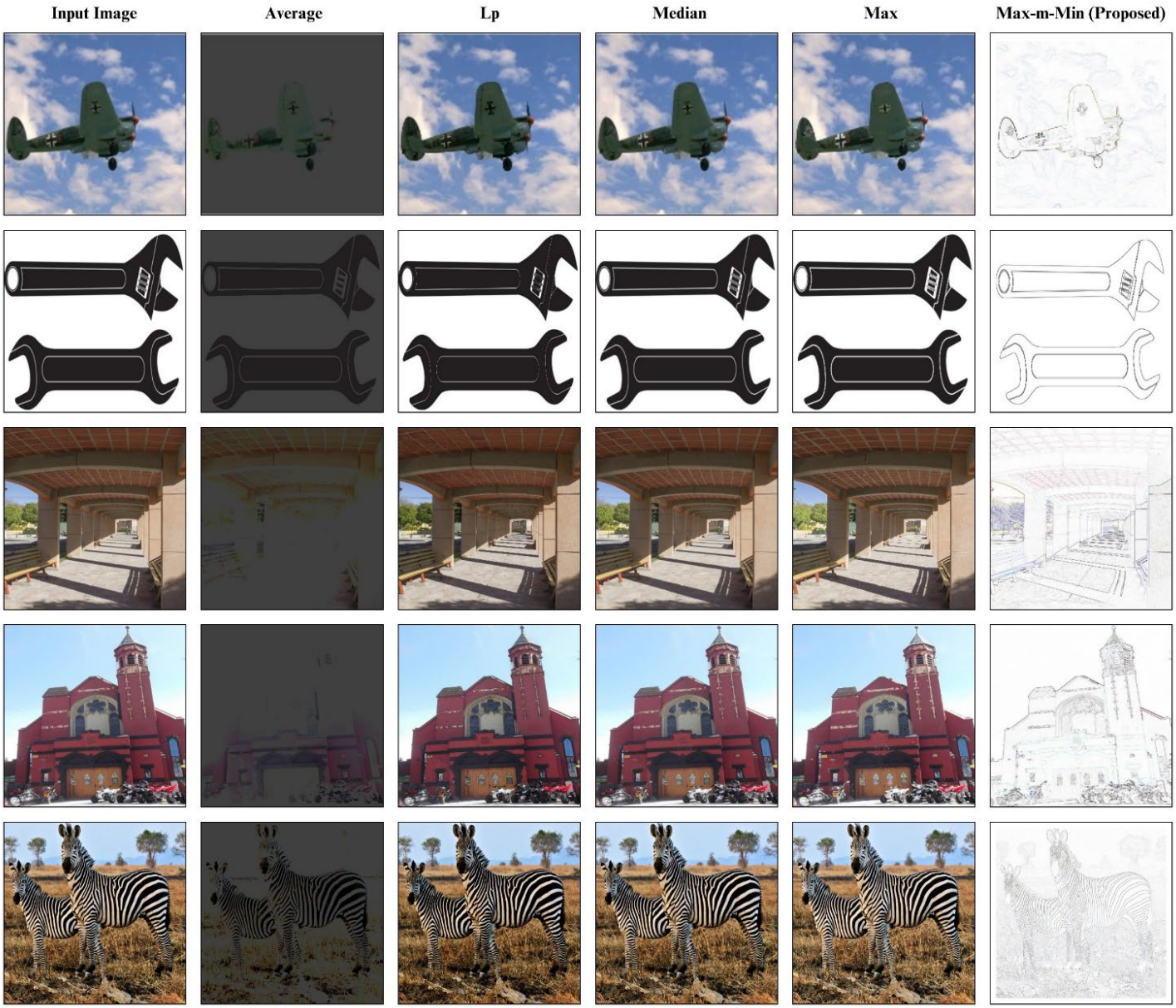

Figure 4.4: Comparison of different pooling methods on various inputs. The pooling size is 2×2 with stride=1. *Max-m-Min* detects fine edges that enable the network to learn texture features effectively.

Before we study the impact of the proposed *Max-m-Min* pooling operation, we examine the result of this operation by applying different pooling methods on some selected images from the Cifar10, Cinic10, Animals, and COCO datasets and compare the results with those obtained by applying average, max, Lp, and median pooling operations. Figure 4.4 shows the output images resulting from using the various pooling operations. It is very clear from this figure that the proposed *Max-m-Min* operation is the most successful in extracting the edge information of the

images. Hence, we expect that the use of the proposed *Max-m-Min* operation should have a more positive impact on the proposed network performance in comparison to the use of the other pooling operations.

We now consider the impact of using the proposed *Max-m-Min* pooling operation on the retrieval performance and compare it with that obtained by employing the other four pooling operations. The results are given in Table 4.3. It is seen from this table that the proposed *Max-m-Min* pooling operation provides the best network performance over all the datasets.

Table 4.3: The network performance comparison in terms of mAP when PL1, PL2 corresponding to Average, Max, Lp, Median, and *Max-m-Min* pooling operations.

Pooling operation	Cifar10	Animals	Cinic10	COCO	ImageNet	YFCC*
Average	0.8708	0.8514	0.8143	***0.7416***	***0.6853***	***0.3215***
Max	***0.8860***	0.8514	***0.8184***	0.7412	0.6847	0.3210
Lp	0.8810	0.8505	0.8146	0.7292	0.6528	0.2918
Median	0.8797	***0.8535***	0.8170	0.7259	0.6694	0.3002
Max-m-Min	**0.9170**	**0.8544**	**0.8231**	**0.7833**	**0.6955**	**0.3243**

The best results are shown in **BOLD** font, and the second-best results are presented in ***ITALIC*** font.
* The network is pretrained on ImageNet and finetuned on YFCC.

Table 4.4: Training and testing times (in seconds) of the proposed network with different pooling operations.

Pooling operation	Training time* (one epoch)	Testing time*
Average	68.13	0.1473
Max	**67.74**	0.1471
Lp	81.54	0.1485
Median	70.93	***0.1470***
Max-m-Min	***67.96***	**0.1468**

The best results are shown in **BOLD** font, and the second-best results are presented in ***ITALIC*** font.
* The network includes the proposed residual block.

Table 4.4 provides the training and testing times of the proposed network employing the proposed *Max-m-Min* pooling operations and other pooling operations for PL1 and PL2, using the Cifar10 dataset. This table shows that the proposed network using the proposed *Max-m-Min* pooling operation takes training time that is about the same as that taking when the average or max pooling operations is employed. It is seen from the table that the proposed *Max-m-Min* pooling operation takes less testing time than when other pooling operations are used.

Preserving discriminative information while performing pooling operations is crucial to achieving desirable performance in a deep network. The discrimination ability of the pooled features is dominated by diverse and salient information [50]. In max pooling, the most salient information contributing to local features is preserved by enhancing the high-frequency components. As a result, max pooling may be well suited for simple images in which the foreground objects contain high-frequency components, such as the images in the Cifar10 dataset. However, in complex images, such as images in the COCO dataset, foreground items with less texture may disappear after several max pooling operations; thus, the features are more likely to lack the foreground object's features. Additionally, average pooling retains a wide range of data by merging features from all parts of the image. However, average pooling may perform poorly because salient information is constantly linked with nonsalient details. Therefore, a pooling operation that provides a trade-off between salient and diverse information may improve the discriminating capacity of the generated features. In light of the above discussion, Shannon entropy can be used to evaluate this trade-off in the pooling method [50]. Mathematically, Shannon entropy calculates the uncertainty associated with a certain value appearing in a pooling window. Considering pooled activation $o_c[m,n]$ over a pooling window defined by $\mathcal{N}_{a\times b}^{(m,n)}$ as a random variable $\boldsymbol{E}$ with $Z = a.b$ possible values $\{e_1, e_2, \dots, e_Z\}$ and $\{p_1, p_2, \dots, p_Z\}$ as the corresponding probabilities, the entropy $H(e)$ of a pooled activation $\boldsymbol{E}$ is defined as

$$H(e) = -\sum_{z=1}^{Z} p_z \, log_2 \, p_z \tag{4.21}$$

Assuming a 2×2 pooling window ($Z = 4$), the entropy of max pooling is

$$H(e) = -(1\, log_2\, 1) = 0 \tag{4.22}$$

This means there is no uncertainty, as max pooling selects the strongest value; max pooling assigns a probability of 1 to the maximum value and 0 to the others. In contrast, average pooling is calculated by assigning equal probability to every component in the pooling window. Hence, for average pooling, we have

$$H(e) = -\sum_{z=1}^{Z}(\frac{1}{4} log_2 \frac{1}{4}) = 2 \tag{4.23}$$

The proposed *Max-m-Min* pooling operation always considers two extreme values in the pooling window, namely, the maximum and minimum values. Therefore, the entropy for *Max-m-Min* pooling is given by

$$H(e) = -(\frac{1}{2} log_2 \frac{1}{2} + \frac{1}{2} log_2 \frac{1}{2}) = 1 \tag{4.24}$$

This value is between those of the max pooling and average pooling, meaning that the *Max-m-Min* pooling provides an excellent trade-off between these two pooling operations. In other words, the proposed *Max-m-Min* pooling operation can effectively preserve salient features and diverse information. The entropy values of different pooling operations are presented in Table 4.5.

Table 4.5: Shannon entropy $H(e)$ of different pooling methods.

Pooling operation	$H(e)$
Average	2
Max	0
Lp	$[0, 2]$
Median	$0, 1$
Max-m-Min	1

Table 4.6: Impact of the size of morphological operators on the network performance in terms of mAP.

Operator Size	Animals	COCO
2×2	**0.8544**	**0.7833**
3×3	***0.8196***	0.7228
4×4	0.8107	0.7229
5×5	0.8054	***0.7298***
6×6	0.8098	0.7101
7×7	0.7875	0.7043
8×8	0.7964	0.7032
9×9	0.7741	0.6856

The best results are shown in **BOLD** font, and the second-best results are presented in ***ITALIC*** font.

Table 4.7: Impact of using different fusion strategies on the performance of the network in terms of mAP.

Fusion method	Cifar10	Animals	Cinic10	COCO	ImageNet	YFCC
Summation	0.8754	0.8185	0.8376	0.7205	0.5911	0.2954
Concatenation	**0.9170**	**0.8544**	**0.8231**	**0.7833**	**0.6955**	**0.3243**

BOLD font indicates the best performance.

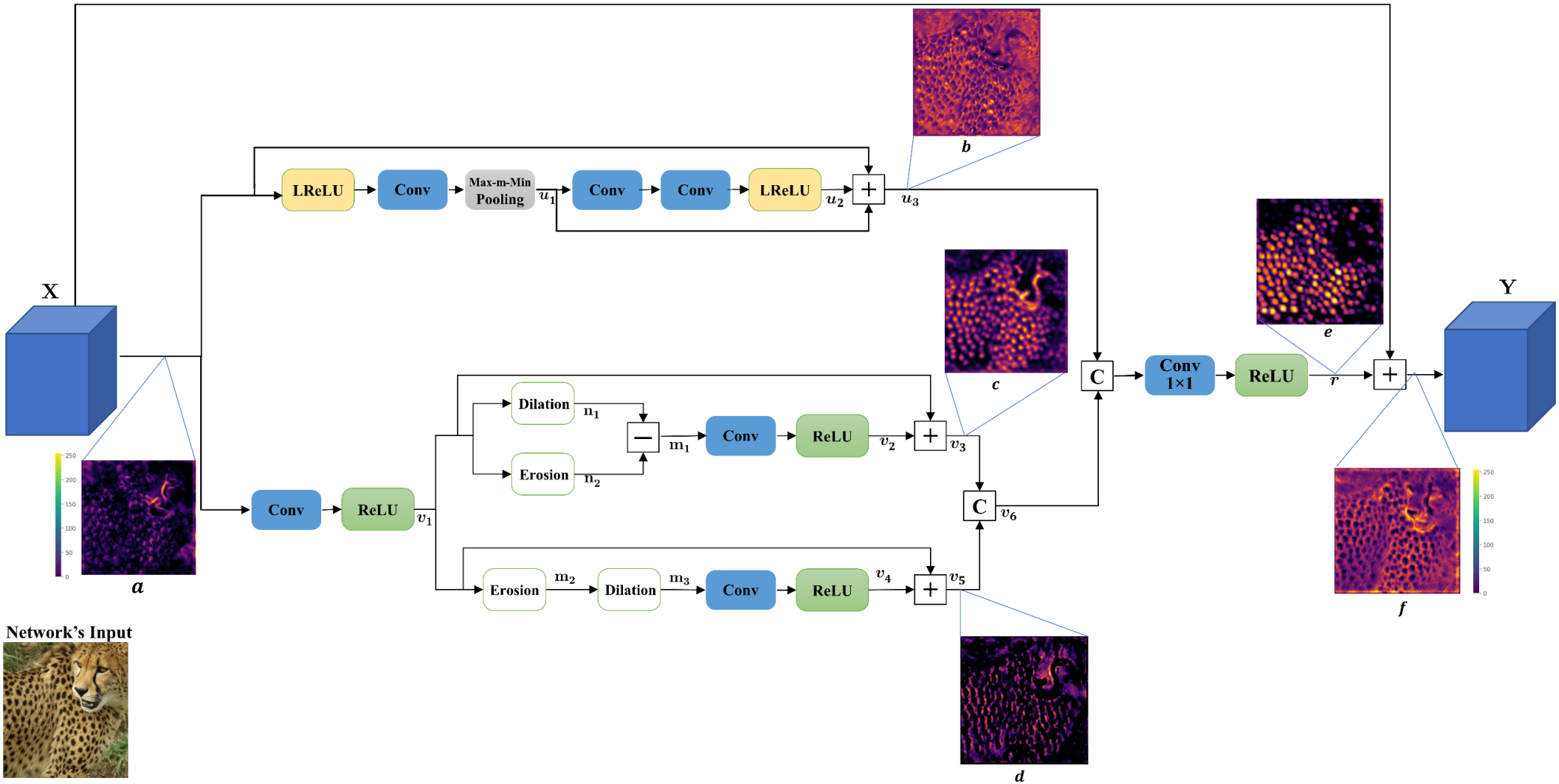

Figure 4.5: Visualization of feature map evolution through the residual block. The residual block outputs a feature vector enriched with textural and structural information.

As mentioned in Section 4.2, using the proposed residual block enables the deep network to learn rich sets of feature maps. In order to visually see as to how the objective of extracting a rich set of feature maps by the proposed residual block we have isolated the 5th residual of an operational proposed network and presented it as Figure 4.5. A comparison of the typical feature maps $b, c,$ and d with the map a shows that the edge extraction module and the two branches of the morphological module are indeed achieving their objectives in that map b has more information on the edges of the input image, whereas maps c and d have more textural and structural information about the image. On the other hand, feature map f has all the information contained in b,c, and d combined together.

As previously mentioned, all the morphological operators employed to generate morphological features use a kernel of size 2×2. We now investigate the effect of using other kernel sizes on the network performance. The results using the Animals and COCO datasets are given in Table 4.6. It is seen from this table that the kernel size of 2×2 provides the best performance and the kernel size of 3×3 yields the second-best performance.

Two strategies can be used for feature fusion in deep networks, namely, summation and concatenation. In the proposed residual block, we utilize concatenation to fuse multiple feature tensors. In Table 4.7, we provide the results of using each of the two fusion schemes. It is seen from this table that concatenation is a better fusion strategy in comparison to the summation strategy, regardless of the dataset used.

Table 4.8: Comparison between the performance (in terms of mAP) of the convolutional neural networks for image retrieval.

Method	Net	F-Tun	Dim	Params	mAP					
					Cifar10	Animals	Cinic10	ImageNet	COCO	YFCC*
Neural Code (2014) [8]	V	Yes	4096	138M	0.6723	0.6387	0.5485	0.6458	0.7512	0.2608
DHN (2016) [81]	A	Yes	64	62M	0.6210	–	–	0.5730	0.7340	–
MSIR (2020) [82]	V	Yes	4096	138M	0.7955	0.4732	0.6163	0.5642	0.6734	0.1645
HashNet (2017) [79]	V	Yes	64	138M	0.7870	**0.6663**	0.7132	**0.6955**	0.7360	**0.2987**
DPAH (2020) [83]	V	Yes	64	138M	–	–	–	**0.7138**	**0.7815**	–
UDPH (2022) [70]	R	Yes	256	44M	0.5520	0.4843	0.4934	0.5357	0.7201	0.2814
Neural Code (2014) [8]	V	No	4096	138M	0.6545	0.6243	0.5154	0.5198	0.7362	0.2343
MSIR (2020) [82]	V	No	4096	138M	0.6643	0.4156	0.5856	0.5337	0.6748	0.1247
DHA (2019) [84]	A	No	64	62M	0.6990	–	–	–	**0.7532**	–
UDPH (2022) [70]	R	No	256	44M	0.4965	0.3998	0.4112	0.5057	0.7073	0.2156
XBL (2021) [85]	X	No	4096	44M	0.7254	0.6629	0.6845	0.6823	0.7182	0.1858
MRDL (2021) [68]	V	No	4096	138M	0.6657	0.6256	0.6534	0.5963	–	–
DELG (2020) [67]	R	No	2048	44M	**0.9046**	**0.6787**	0.7176	0.6914	0.7525	**0.3034**
C-LSH (2018) [86]	E	n/a	2048	103M	0.6365	0.3640	0.3843	0.5963	0.6883	0.2443
TBH (2020) [87]	E	n/a	64	71M	**0.8739**	–	–	–	0.7211	–
MorIRNet (2022) [55]	E	n/a	3136	85M	0.7773	0.6654	0.7168	0.6145	0.6954	0.2778
MSFRNet (Ch.3)	E	n/a	169	62M	0.8431	0.6346	**0.7487**	0.5318	0.5617	0.2224
HMSRNet (Ch.3)	E	n/a	169	62M	0.8643	0.6405	**0.7390**	0.5423	0.5901	0.2361
MoFNet (proposed)	E	n/a	512	64M	**0.9170**	**0.6934**	**0.7317**	**0.6990**	**0.7833**	**0.3243**

BOLD, ***ITALIC***, and **underlined** fonts indicate the best, second-best, and third-best performance, respectively.

"Net" is the backbone network that is used by the corresponding method: VGG16 (V), ResNet101 (R), AlexNet (A), Xception (X), or end-to-end (E). "F-Tun" is "Yes" when the fine-tuned network is used, "No" when the off-the-shelf network is used, and "n/a" when the network is trained from scratch. "Dim" is the final feature vector dimension. "Params" is the number of parameters (in millions). † Euclidean metric is used. Euclidean distance is used as the similarity measurement for all the datasets.

* The results are obtained using the network that is pretrained on ImageNet and fine-tuned on YFCC.

We now compare our proposed MoFNet with several conventional and state-of-the-art image retrieval methods, including Neural Code [8], UDPH [70], C-LSH [86], DHA [84], TBH [87], MorIRNet [55], DHN [81], DPAH [83], MSIR [82], XBL [85], MRDL [68], DELG [67], and HashNet [79]. The results are reported in Table 4.8. It is clear from the table that the proposed MoFNet gives the best performance on all the datasets except for the ImageNet dataset, where its performance is the second-best. In particular, for the Cifar10 dataset, the proposed MoFNet gives a performance that is significantly superior to that of all the other networks. In the case of the YFCC dataset, which is a very challenging large-scale dataset, our proposed MoFNet exhibits the best results, with a mAP of 0.3243. This retrieval performance is significantly higher than the second-best result of mAP=0.3034 obtained by DELG [67]. The relatively poor performance on this dataset by all the methods is due to the dataset size and the fact that its label distribution is not well-balanced. It is to be noted that the higher performance of the proposed MoFNet is achieved despite the fact that it uses a smaller number of parameters than most of the other networks. It should also be noted that the performance of each of those networks that uses the number of parameters smaller than the proposed MoFNet is very much lower.

4.4 Summary

In this chapter, we have developed a deep convolutional neural network guided to extract the textural, structural, and edge information contained in images. Recognizing that morphological operations process, the texture and structure of objects based on their geometrical properties and that edges represent fundamental features of an image, we have used these ideas in our network. We have utilized morphological operations to guide the network to extract textural and structural information. Also, a novel pooling operation has been designed to extract the edge information in an image. Extensive experimental results have confirmed the effectiveness of our proposed network, significantly enhancing the learning quality of the deep network for image retrieval tasks.

Chapter 5

Hashing-based Re-ranking Technique for Image Retrieval

Re-ranking is a task of refining an initially ranked list of images obtained from an image retrieval technique for a given query image, with the goal of enhancing retrieval performance in an efficient manner. However, existing re-ranking methods suffer from high computational complexity, leading to slow and resource-intensive operations that render them impractical for real-life applications. This necessitates the development of a computationally efficient re-ranking approach that effectively improves retrieval performance. Image hashing is one of the techniques that has shown promising performance along with computational efficiency for the task of image retrieval [32], [88]. Motivated by these advantages offered by image hashing, this chapter develops a novel and computationally efficient re-ranking method for image retrieval, utilizing the speedy and proficient nature of image hashing techniques for image representation [89].

5.1 RefinerHash: A New Hashing-based Re-ranking Technique for Image Retrieval

5.1.1 Initial Retrieval List

In order to prepare the initial retrieval list, we employ two distinct methods. The first method, detailed later in this chapter, is based on our approach presented in [89], which utilizes A novel Block-wise technique and a pre-trained deep network to generate the initial retrieval list. We refer to this method as Block-wide Hopfield-based Image Retrieval (BHIR). The second method is MoFNet, proposed in [72] and extensively detailed in Chapter 4. MoFNet differs as it requires a deep network to be trained. By strategically using these methods in tandem, we aim to explore the impact of both improving representational capacity through MoFNet and employing effective re-ranking strategies, with the goal of maximizing the overall performance in our image retrieval tasks.

5.1.1.1 Block-wise Hopfield-based Image Retrieval

The general framework of the initial list creation process of [89] is depicted in Figure 5.1. The process consists of two main steps: training and testing. During the training step, we partition each image available in the training set into b blocks. We extract features from these blocks using deep features using ResNet50 deep network. Using these features, we calculate weight matrices, $\boldsymbol{W}_b^j$, for each block, which are then summed to obtain b weight matrix sets, $\boldsymbol{W_b}$. During the testing step, these matrices are used for calculating scores for initial list creation.

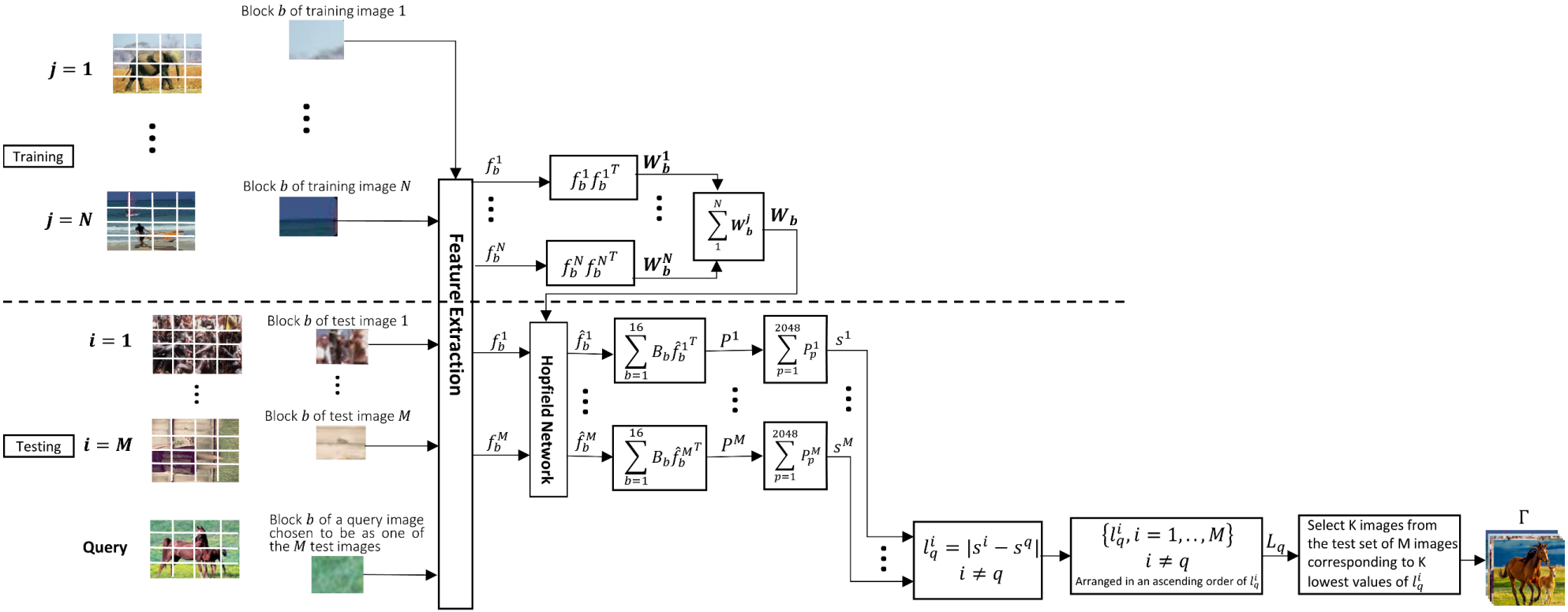

Figure 5.1: Framework of the proposed method for the initial retrieval list creation. During the training step, the weight matrix that is used in the testing step, $\boldsymbol{W_b}$, is calculated to create the initial list. The "Feature Extraction" module can be based on either low-level features or deep features. The sizes of feature vectors, f_b^N, f_b^M and $\boldsymbol{W_b}$ are 2048×1, 2048×1 and 2048×2048, respectively. The output is K images, which are stored in a set called Γ. "T" which is the super-script to the feature vectors, is the transpose operator.

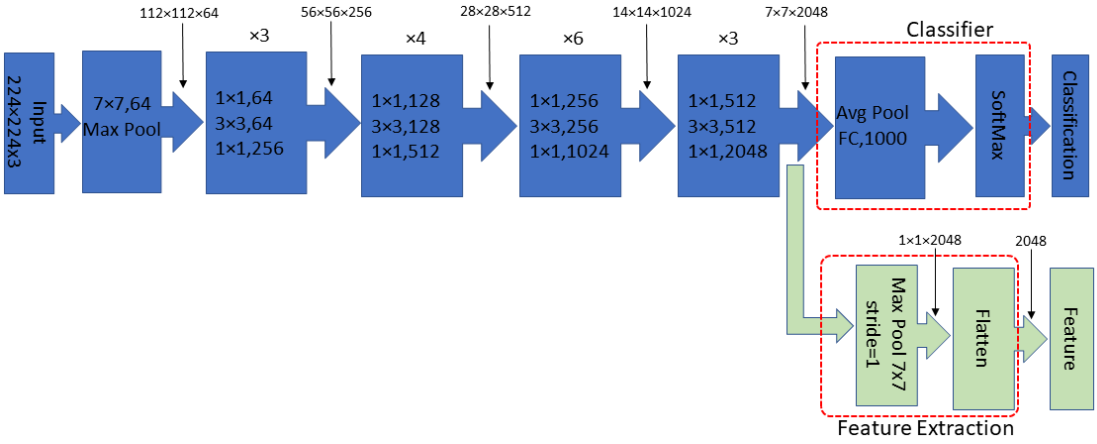

Figure 5.2: The ResNet50 architecture and the modifications made for its use as a feature extractor. The depicted architecture shows the size of filters and the dimensions of the outputs of convolutional modules. The notation "$n \times n$, k" in the convolutional modules denotes a filter size of $n \times n$ with k filters. The values at the top of each module denotes the repetition of the corresponding module. The values above the arrows represent the size of the output of the corresponding module. The Classification head is removed from the pre-trained ResNet50 and replaced with the "Feature Extraction" module. For simplicity, residual shortcuts and activation functions are not shown.

We utilize the ResNet50 [90] convolutional neural network, as depicted in, for feature extraction. We modify the standard structure of ResNet50 to fit our purpose. As illustrated in Figure 5.2, the classification head of ResNet50 is replaced with a feature extraction module. Specifically, our feature extraction module takes the output of the last convolutional layer, size 7×7×2048, and passes it through a global max pooling layer with a window size of 7×7 and a stride of 1. The output from this pooling layer then undergoes a flattening process, resulting in a 2048-dimensional vector. This vector serves as our desired image representation. The choice of ResNet50 was motivated by its demonstrated efficiency in extracting intricate image features and its robustness in handling a variety of visual tasks. Additionally, using a pre-trained model allows us to capitalize on the knowledge gained from large-scale datasets, thereby providing a strong foundation for our feature extraction.

An example of the blocking scheme is shown in Figure 5.3. As the figure shows, the weights are assigned based on the location of the blocks within an image, and the blocks in the central region of the image, which contain the most informative visual information, are assigned greater weights compared to those located along the borders. These weight values are pre-defined and

determined experimentally. For our experiments, the following values are used in the block weight vector:

$$B = \begin{bmatrix} B_1 & B_2 & B_3 & B_4 \\ B_5 & B_6 & B_7 & B_8 \\ B_9 & B_{10} & B_{11} & B_{12} \\ B_{13} & B_{14} & B_{15} & B_{16} \end{bmatrix} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 2.0 & 2.0 & 0.5 \\ 0.5 & 2.0 & 2.0 & 0.5 \\ 0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix}$$
$$\vec{B}_b = \{B_1, \dots, B_{16}\} \tag{5.1}$$

Each image in the dataset is divided into b blocks. For each specific block position, we collate the same block across all images to form a block set. For instance, block set 1 contains the first block from all images in the dataset. For each of these block sets, we compute a distinct weight matrix. This process is repeated across all block positions, yielding a collection of weight matrices that uniquely correspond to each block position within the images. The first step in calculating the desired weight matrices is to compute the weight matrices for each image in the training set:

$$\boldsymbol{W}_b^j = f_b^j f_b^{j^T} \tag{5.2}$$

where $\boldsymbol{W}_b^j$ is the weight matrix for the b-th block of the j-th image and f_b^j is the feature vector for the b-th block of the j-th image. The superscript T represents the transpose operation, and j is the number of images in the training set, with $j \in \{1 \dots N\}$. Therefore, for each block set, there are N weight matrices. Adding these weight matrices (for a block set) yields the desired weight matrix for block set b as follows:

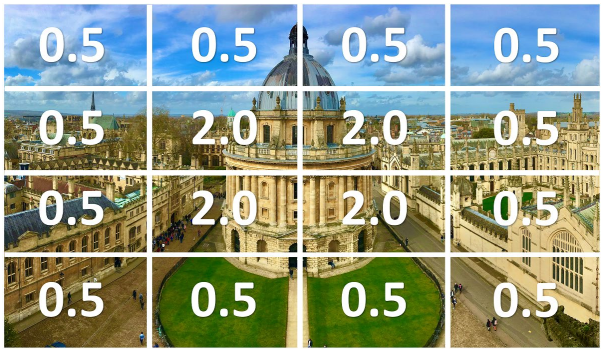

Figure 5.3: The block weight vector. The center region includes four blocks and has a four times greater influence on the score calculation than the blocks at the borders. These values are found experimentally and provide the best results.

$$\boldsymbol{W}_b = \sum_{j=1}^{N} \boldsymbol{W}_b^j \tag{5.3}$$

Considering that there are 16 block sets, we have a total of 16 different weight matrices, each with a size of 2048×2048. These matrices encapsulate the associative memory capabilities of the Hopfield network to be used in the initial retrieval list creation.

5.1.1.2 MoFNet

In Chapter 4 of this work, a detailed explanation of MoFNet is provided. MoFNet is specifically designed to enhance the representational capacity of features for image retrieval. It achieves this by integrating two innovative approaches: a new pooling mechanism known as Maximum minus Minimum (*Max-m-Min*) pooling and a novel morphological feature-generating residual block (MoF).

The *Max-m-Min* pooling operation is a novel pooling operation that focuses on extracting edge features. It calculates the difference between the maximum and minimum element values within a pooling window of a feature map to highlight the edge features. When the pooling operation is applied on an input feature map, the resultant feature map consists of robust edge features.

The morphological feature-generating residual block (MoF) consists of three modules: an edge feature extraction module, a morphological feature extraction module, and a feature fusion module. The edge feature extraction module utilizes the *Max-m-Min* pooling method alongside conventional convolution operations to extract high-frequency components and learn discriminative features. The morphological feature extraction module employs nonlinear morphological operations to extract textural and structural information from the image, contributing to a rich set of features. Lastly, the feature fusion module combines the outputs from the previous two modules, providing the network with high representational features for the task of image retrieval.

The MoFNet architecture includes multiple layers of conventional convolution layers, a sequence of five MoF residual blocks, and some *Max-m-Min* pooling layers. During training, the network employs a classifier module, while for testing, the classifier module is replaced with a feature representation module. The trained network is then used as a feature representation module.

5.1.1.3 Initial Retrieval List Creation using BHIR

Before the creation of the initial retrieval list for a given query, all the M images in the test set should go through the feature extraction step to obtain their corresponding feature vectors, namely $f_b^i, i \in \{1 \dots \mathrm{M}\}$, following the same procedure used in the training step. These obtained feature vectors, f_b^i, are then applied to the trained Hopfield network. In our image retrieval method, the network is used to transform the feature vectors of the images during the testing phase. By employing the network, with its weight matrix calculated in the training part (as discussed in Section 3.1.4), our method takes advantage of the associative memory capability of the network to align the feature vectors of the images more closely with the vectors of similar images. Utilizing the Hopfield network, each feature vector f_b^i is transformed into a new vector, $\hat{f}_b^i$.

Once all the new feature vectors, $\hat{f}_b^i$, are calculated, they are used to generate a similarity score for every image in the test set with respect to the query image. This query image is one of the M images in the test set. The process of creating the initial list begins with the calculation of the score for image i of the test set as follows:

$$s^i = \sum_{p=1}^{2048} P_p^i \tag{5.4}$$

Here, P^i is calculated using:

$$P^i = \sum_{b=1}^{16} B_b \hat{f}_b^{i^T} \tag{5.5}$$

where B_b is weight for block b. Next, we compute the distance measure, which is done by subtracting the score of the query image (s^q) from the score calculated for every other image in the test set. This is represented as:

$$l_q^i = \left| s^i - s^q \right|, i \in \{1 \dots M\}, i \neq q \tag{5.6}$$

where, s^i and s^q are the scores calculated for images in the test set and the query image, respectively, and $|.|$ is the absolute value operation. Afterward, we arrange the computed l_q^i values in ascending order:

$$L_q = sort\left(\left\{l_q^i, i = 1, .., M\right\}\right), i \neq q \tag{5.7}$$

From this sorted list, we select the images corresponding to the K lowest values of l_q^i that are more likely to include similar images to the query image. These selected images form the initial retrieval list (denoted as Γ), which serves as the input for further processing in the re-ranking step.

5.1.1.4 Initial Retrieval List Creation using MoFNet

In the image retrieval process using MoFNet, the first step is to compute the feature maps for a query image and all images in the database. To find the initial retrieval list, the query feature vector is compared with all the feature vectors from the database using a suitable similarity measurement. This comparison measures the similarity between the query and each image in the database based on their respective feature maps. The images are then ranked according to their similarity scores. The top K images, which have the highest similarity scores, are selected to form the initial retrieval list Γ.

5.1.1.5 Hash Code Generation

In this section, we develop RefinerHash, which is a computationally efficient re-ranking scheme for image retrieval. We first present the generation of hash codes using Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT), yielding concise yet comprehensive representations of the images. These hash codes form the backbone of the re-ranking process. Utilizing the hash codes, we present our image search technique that employs an efficient tree data structure to boost search speed and retrieval accuracy.

General hash code generation technique: Our approach involves generating hash codes for a given image by combining the codes calculated for the different rotations of the image. We initiate the process by partitioning an image into blocks. Following this, a transformation is applied to these blocks, and the median value of the transformed blocks, denoted as m, is calculated. Subsequently, a single bit of the hash code is computed by comparing this median value with the average coefficient value of the transformation within each block. This bit-generation procedure is summarized as follows:

$$f(A_b) = \begin{cases} 0, & A_b < m \\ 1, & A_b \geq m \end{cases} \tag{5.8}$$

where A_b represents the average coefficient value calculated for the b-th block. Once each bit has been calculated for all blocks, the next step is to create the final hash code, which is created by merging the bit outputted by applying the hash code generation technique on all blocks.

Hash code generation based on DCT: Figure 5.4 shows our proposed image hashing technique using DCT transformation. As the figure shows, the process involves deriving four distinct hash codes, denoted as h_m, with $m \in \{1, 2, 3, 4\}$, from concatenating the hash codes obtained by rotating the image to four orientations: $\theta = \{0, \pi/2, \pi, 3\pi/2\}$. Subsequently, these images undergo a DCT transformation. However, directly applying DCT to the entire image may include irrelevant background information, limiting the representational capacity of the hash codes [32]. To overcome this limitation, our approach selectively applies DCT to the central region of the image.

The central region is defined by a 128×128 patch which is partitioned into α equal blocks, where $\alpha = 16$ and each block of size 32×32 pixels. The hash code calculation starts by applying DCT to each of these blocks:

$$\Phi^i_{\theta,\alpha} = DCT\left(I^i_{\theta,\alpha}\right) \tag{5.9}$$

where $I^i_{\theta,\alpha}$ is the α-th block of the i-th image rotated θ degrees. The average coefficient value, $\mu^i_{\theta,\alpha}$, is computed from these DCT coefficients for each block. These average values are further averaged for all α blocks in the central region of the image, resulting in values denoted as μ^i_θ. Each bit of the hash code, $h^i_{\theta,\alpha}$, is determined based on the comparison of $\mu^i_{\theta,\alpha}$ with the median value, m^i_θ, as follows:

$$h^i_{\theta,a} = \begin{cases} 0, & \mu^i_{\theta,\alpha} < m^i_\theta \\ 1, & \mu^i_{\theta,\alpha} \geq m^i_\theta \end{cases} \tag{5.10}$$

where m^i_θ is the median of all the average values for the i-th image when rotated by θ degrees. By processing all the blocks and concatenating the obtained bits, we generate the final hash code for the i-th image as follows:

$$h^i = h^i_{3\pi/2,16} \ \ldots \ h^i_{0,1}, i = 1 \ldots \mathrm{K} \tag{5.11}$$

The size of h^i is equivalent to the product of the number of rotations and windows. As such, the size of a DCT-based hash code totals 4×16, or 64 bits. Given that our initial list comprised K images, we consequently generated K hash codes.

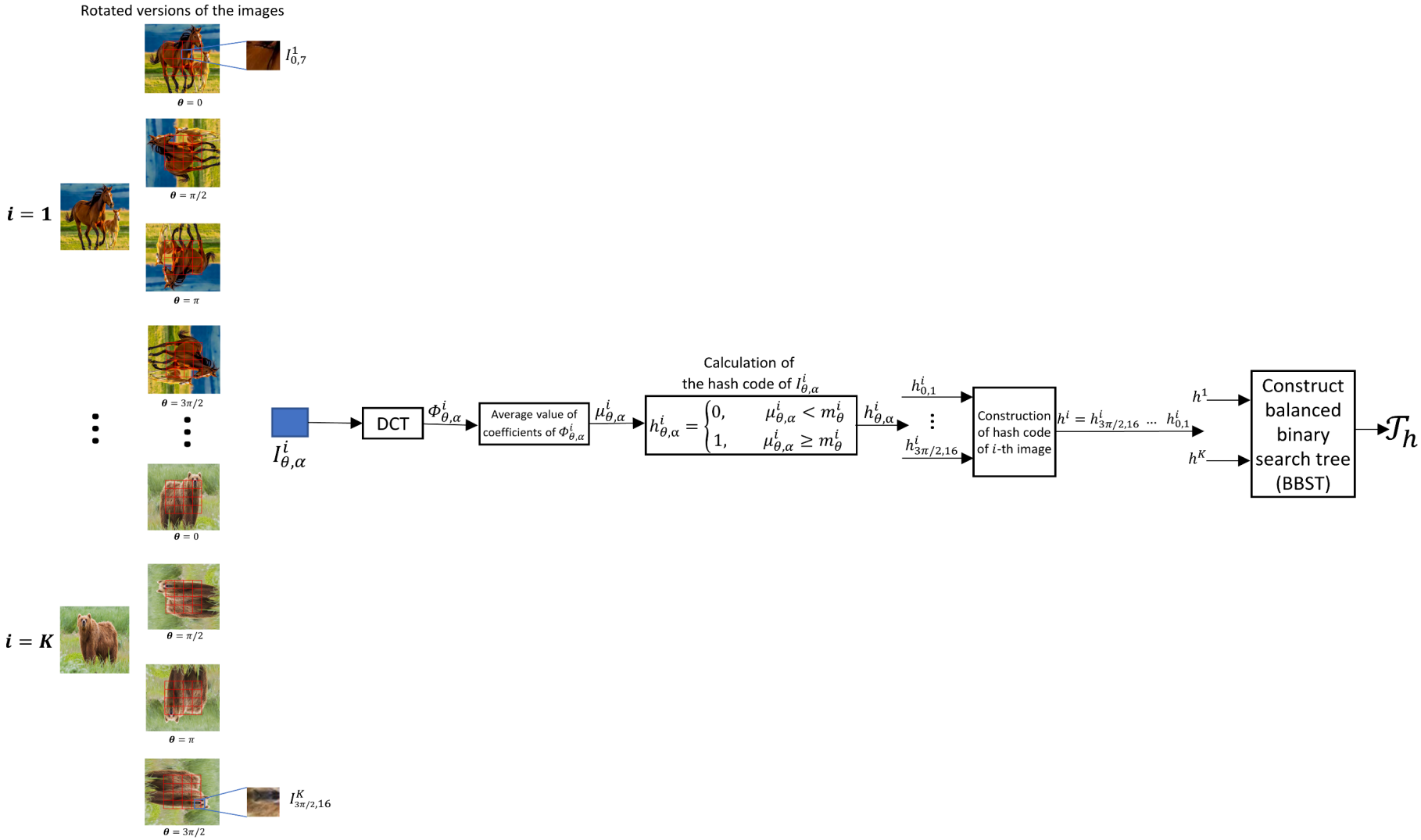

Figure 5.4: An overview of how a hash code is generated using DCT. Given rotated images, the hash codes based on DCT are calculated on partitioned central regions. The output is a hash code generated by concatenating all the hash codes calculated for an image. All the final hash codes are used to build a tree ($\mathcal{T}_h$). K is the number of images available in the initial retrieval list, θ is the image orientation, and α is the block number.

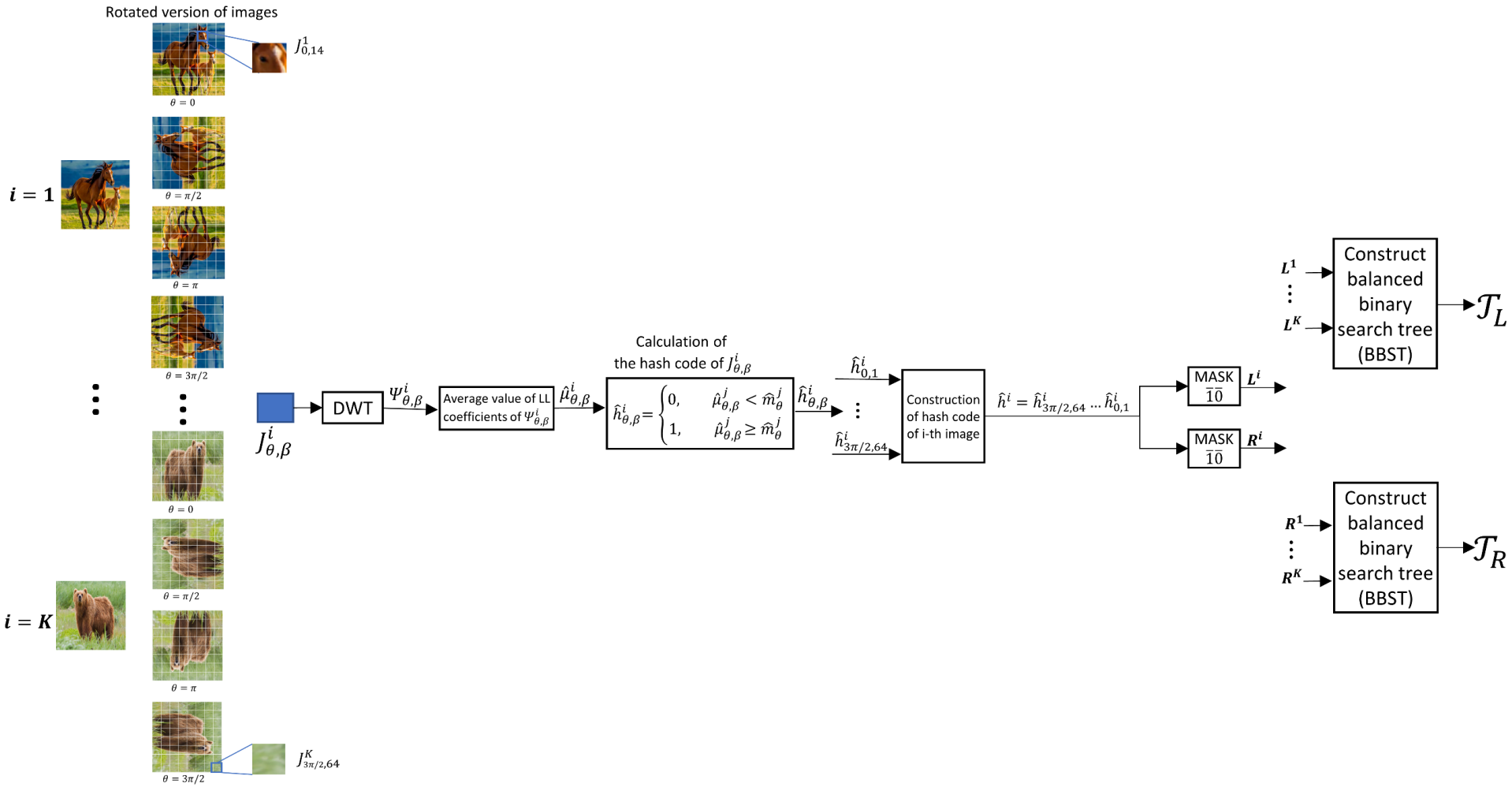

Figure 5.5: An overview of how hash codes are generated using DWT. Given rotated images, the DWT-based hash codes are calculated on the partitioned whole image. The hash codes are then divided into equal-length hash codes, namely L^i and R^i. These two hash codes are used to build two different trees ($\mathcal{T}_L$ and $\mathcal{T}_R$). Note that $\Psi^i_{\theta,\beta}$ is the coefficient of the LL band after the first decomposition, θ is the image orientation, β is the block number, and K is the number of images in the initial retrieval list.

Hash code generation based on DWT: Our proposed DWT-based hashing technique is shown in Figure 5.5. As the figure shows, the algorithm for calculating hash codes using DWT is similar to that for generating DCT-based hash codes. However, the DWT-based method generates two hash codes per image, taking advantage of the multi-resolution representation that DWT offers. The DWT transformation decomposes an image into four bands: low-low (LL), low-high (LH), high-low (HL), and high-high (HH). We specifically select the LL band, which contains the most critical information about the original image [91]. Therefore, the hash codes derived from this LL band encapsulate essential and expansive details about the image, effectively representing its content.

Similar to the DCT-based hashing technique, we introduce a step of rotating images prior to the application of DWT, aiming to improve the representational diversity of the generated hash codes. This rotation presents a different perspective of the image content, thus potentially providing additional features. By generating hash codes from the LL band of the rotated images, our method captures a broader range of details, resulting in a more comprehensive representation of the image. This rotation-aided approach thereby enhances the robustness and representational capacity of the generated hash codes.

The process begins by dividing a given image I into β blocks with a size of 32×32. After the process of dividing the image and applying the DWT transformation to each block, we then derive a coefficient, $\Psi^i_{\theta,\beta}$. From these coefficients, we select the low-low (LL) bands after the first decomposition of the DWT for further processing. We proceed by averaging the coefficient for each block to yield $\hat{\mu}^i_{\theta,\beta}$. The subsequent step is the calculation of the median for all the blocks of each orientation of a given image, denoted as, $\hat{m}^i_\theta$. Given $\hat{\mu}^i_{\theta,\beta}$ and $\hat{m}^i_\theta$, we can compute one bit of the hash code as follows:

$$\hat{h}^i_{\theta,\beta} = \begin{cases} 0, & \hat{\mu}^i_{\theta,\beta} < \hat{m}^i_\theta \\ 1, & \hat{\mu}^i_{\theta,\beta} \geq \hat{m}^i_\theta \end{cases} \tag{5.12}$$

Once the bits for all blocks have been computed, the final step in the DWT-based method is to construct the hash code by concatenating the calculated bits for all the blocks of images rotated θ degrees, leading us to a hash code as follows:

$$\hat{h}^i = \hat{h}^i_{3\pi/2,64} \ \dots \ \hat{h}^i_{0,1}, i = 1 \dots K \tag{5.13}$$

The length of the hash code for an image, represented by $\hat{h}^i$, is $4 \times 64 = 256$ bits due to the image rotation across four different orientations. We partition this hash code into two equal sections, L^i and R^i, representing the left and right parts of $\hat{h}^i$, respectively. This division is performed as follows:

$$L^i = \hat{h}^i \odot [\bar{1}, \bar{0}]$$
$$R^i = \hat{h}^i \odot [\bar{0}, \bar{1}] \tag{5.14}$$

where $\odot$ denotes the canonical (element-wise) multiplication operation, while $\bar{1} \in R^{128}$ and $\bar{0} \in R^{128}$ are vectors of ones and zeros, respectively. These operations essentially mask one-half of the hash code, retaining the other half. The sections filled with zeros (masked) are subsequently omitted, leaving us with L^i and R^i, each of length 128 bits. These are then used to construct the trees $\mathcal{T}_{\mathcal{L}}$ and $\mathcal{T}_{\mathcal{R}}$, respectively. These trees, along with the tree constructed based on DCT, is the foundation of our re-ranking method. They facilitate efficient image search by serving as structured repositories of the hash codes. Using these hash code repositories, similar images can be quickly identified and retrieved.

5.1.1.6 Image Search

As discussed before, BBSTs are efficient and effective tree data structures that would be highly beneficial for heavy search operations like image retrieval. In light of this advantage of the BBSTs, in our proposed method, we utilize BBSTs as the core mechanism for the image search process. Here, the hash codes function as the keys, and the image filenames serve as the corresponding values. Given the three distinct hash code sets generated for the images in the test set, we create three trees. Each tree is constructed using one of the three hash code sets: $\mathcal{T}_{\mathrm{h}}$ uses the DCT-based hash codes while $\mathcal{T}_{\mathrm{L}}$ and $\mathcal{T}_{\mathcal{R}}$ employ the DWT-based hash codes. Figure 5.6 shows the architecture of our proposed image search method. As shown in the figure, the search starts with the generation of hash codes for the query image using the same algorithms previously used for the test set. Specifically, for each query, three distinct hash codes—L^{q},R^{q}, and h^{q}—are computed and used to search the corresponding tree. These hash codes are used to search the trees. The goal is to identify a subtree whose root's hamming distance to the query hash code is smaller than a predetermined threshold (e.g., less than 20% of the maximum possible distance between two hash codes). When

this criterion is met, the search halts, and the node at which the search stopped, along with all nodes within its subtree, form a candidate list. Figure 5.7 shows an example of this process.

Following the querying of all three trees and identification of the corresponding subtrees, we obtain three candidate sets: δ_1,δ_2, and δ_3. Using these sets, we then construct the following four subsets:

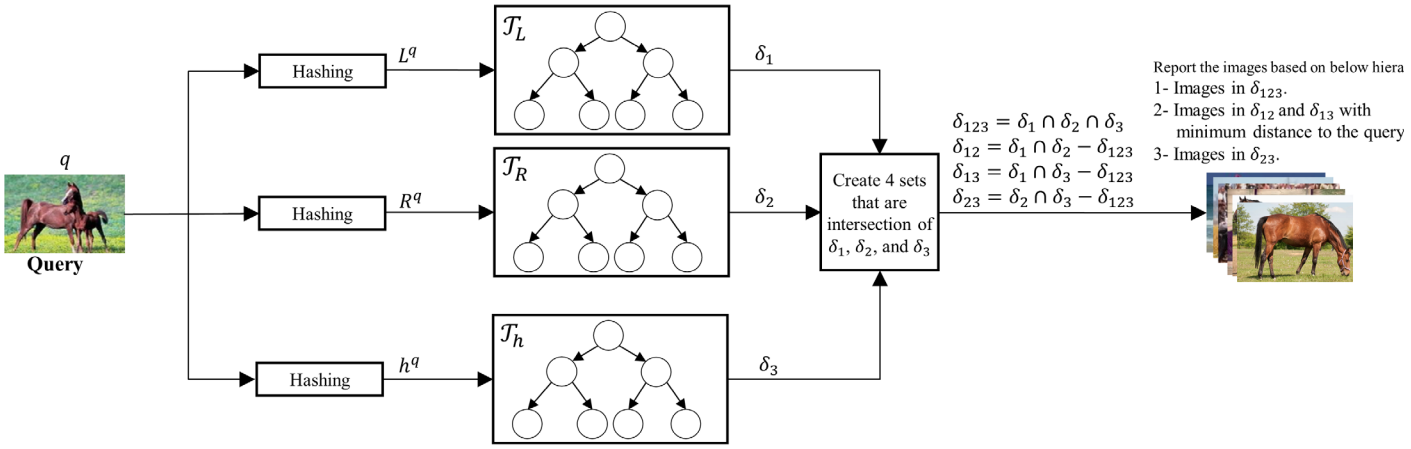

Figure 5.6: Overview of the proposed image search. Three trees are built based on hash codes generated using DWT and DCT transformations. The final image report is done by first reporting all the images in δ_{123}, and if the number of reported images is not sufficient, images in the set of $\delta_{12}, \delta_{13}, \delta_{23}$.

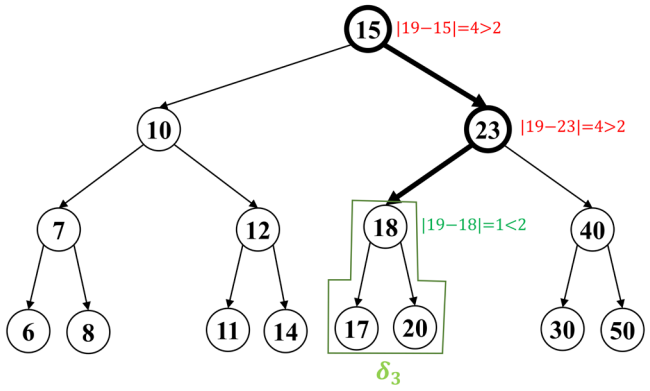

Figure 5.7: Illustration of the process of searching a subtree to generate a candidate list, denoted as δ_3, using a threshold of 2 and a hash code of $h^q = 19$. The bold path represents the search route taken to reach the desired subtree, which is enclosed in a green box. Note that this is a simplified example, and the actual hash codes are longer and stored in binary format, with larger trees.

$$\begin{aligned}
\delta_{123} &= \delta_1 \cap \delta_2 \cap \delta_3 \\
\delta_{12} &= \delta_1 \cap \delta_2 - \delta_{123} \\
\delta_{13} &= \delta_1 \cap \delta_3 - \delta_{123} \\
\delta_{23} &= \delta_2 \cap \delta_3 - \delta_{123}
\end{aligned} \tag{5.15}$$

where $\cap$ and $-$ represent set intersection and set subtraction operations, respectively. An example of the interrelationships between these sets is shown in the form of a Venn diagram in Figure 5.8. As shown, δ_{123} contains images present in all three candidate sets δ_1, δ_2, and δ_3. Meanwhile, δ_{12} comprises images that occur in δ_1 and δ_2, excluding those in δ_{123} to prevent duplication. The sets δ_{13} and δ_{23} are defined in a similar manner. Based on these four subsets, we report the final image list according to the following hierarchy:

1. Images in δ_{123}.
2. Images in δ_{12} and δ_{13} with a shorter Hamming distance to the query.
3. Images in δ_{23}.

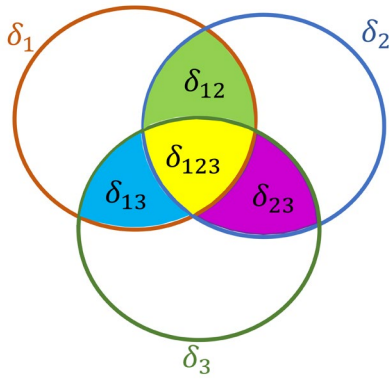

Figure 5.8: A Venn diagram illustrating the interrelationships among sets derived from tree searches. Majority voting determines the similar images for a given query image, with the most similar ones expected to reside in δ_{123}.

Table 5.1: Performance and time complexity comparison of *RefinerHash* with other re-ranking techniques.

Method		mAP					Time (Seconds)				
Baseline	Re-ranking	Cifar10	NUS-WIDE	ImageNet	COCO	YFCC	Cifar10	NUS-WIDE	ImageNet	COCO	YFCC
BHIR		0.7761	0.6323	0.5371	0.6154	0.2224	1120	5040	22400	2380	183300
	+AQE [92]	0.8108	0.6647	0.5898	0.6756	0.2378	*8.9*	*9.2*	*11.4*	*11.2*	*12.1*
	+RCN [93]	*0.8294*	0.6686	0.6023	0.6904	0.1949	41.4	49.5	73.4	70.9	96.5
	+CLEBFSR [25]	0.7934	0.6595	0.55.48	0.6372	0.1701	9.4	10.7	15.6	15.6	14.2
	+TXRBFSR [25]	0.7916	0.6524	0.55.05	0.6219	0.1987	9.3	11.5	15.3	15.9	15.1
	+TDBG [94]	0.8114	0.6734	*0.62.76*	0.6645	0.2047	12.6	16.3	24.3	29.3	40.6
	+SCMR-R [95]	0.7843	*0.7011*	0.65.94	*0.6972*	*0.2457*	33.5	40.1	45.6	46.9	67.8
	+RefinerHash (proposed)	**0.8627**	**0.8090**	**0.69.82**	**0.7401**	**0.2651**	**4.3**	**4.8**	**5.9**	**4.9**	**7.6**

Results in **bold**, *italic* and underlined fonts indicate, respectively, the best, the second-best and the third-best performance.

Table 5.2: Performance comparison in terms of mAP.

Method	Dimension	mAP				
		Cifar10	NUS-WIDE	ImageNet	COCO	YFCC
MSIR (2020) [82]	4096	0.7955	0.6887	0.5642	0.6734	0.1645
PIHE (2021) [96]	128	0.6210	0.5198	0.4630	0.6156	0.1473
C-LSH (2018) [86]	2048	0.6365	0.5634	0.5963	0.6883	0.2443
XBL (2021) [85]	4096	0.7254	0.7182	0.6823	0.7182	0.1858
MRDL (2021) [68]	4096	0.6657	0.6993	0.5963	0.6365	0.2187
CWAH (2022) [97]	512	0.7357	0.7154	0.6481	0.7034	0.2454
DETR (2022) [98]	1000	0.5641	0.7056	0.6128	0.6582	0.2105
UAIR (2022) [99]	128	0.8521	0.7994	0.6473	0.7254	0.2686
GeM (2019) [100]	512	0.8143	0.7656	0.6565	0.6801	0.2256
GreedyHash (2018) [101]	128	0.7585	0.6717	0.6506	0.6554	0.2355
SBA (2019) [102]	512	0.8581	0.7965	0.6856	0.7010	0.2569
AutoRet (2022) [103]	1024	0.6550	0.6175	0.5843	0.6265	0.2403
MSFRNet (Ch.3)	169	0.8431	0.7067	0.5318	0.5617	0.2224
HMSRNet (Ch.3)	169	0.8643	0.7091	0.5423	0.5901	0.2361
MoFNet (Ch.4)	512	*0.9170*	*0.7965*	*0.6990*	*0.7833*	*0.3243*
MSFRNet (Ch.3)+RefinerHash	169	0.8501	0.7124	0.5734	0.5874	0.2374
HMSRNet (Ch.3)+RefinerHash	169	0.8667	0.7153	0.5799	0.6148	0.2482
MoFNet (Ch.4)+RefinerHash	512	**0.9214**	**0.8187**	**0.7021**	**0.7849**	**0.3272**

BOLD, *ITALIC* and underlined fonts indicate, respectively, the best, the second-best and the third-best performance.

5.2 Experimental Results

We first analyze the computational complexity of the proposed RefinerHash algorithm asymptotically. The computational complexity of our proposed method is primarily determined by two parts: the creation of the initial retrieval list and the re-ranking of this list. Given an image dataset divided into N images for the training set (to calculate the weight matrix) and M images for the test set, the complexity of our method for the initial list creation process can be analyzed in two phases: the weight calculation step and the testing step. In the weight calculation step, each image in the training set is divided into b blocks, and the calculation of the weight matrix has a complexity of $\mathcal{O}(\mathrm{bd}^2\mathrm{N})$, where d is the dimension of the feature vector and the squared term, i.e., d^2, arises from the matrix multiplication process[1]. Similarly, in the testing step, the cost of calculating the initial list is $\mathcal{O}(\mathrm{bd}^2\mathrm{M})$ for M images in the test set. Hence, the total time complexity for the creation of the initial retrieval list is $\mathcal{O}(\mathrm{bd}^2(\mathrm{M}+\mathrm{N}))$. It should be pointed out that our method can greatly benefit from the parallel computation capabilities of a GPU, as the computations required for each block are independent of the other blocks, thereby allowing for efficient parallel processing. Therefore, utilizing a GPU can substantially reduce the complexity of creating the initial list to $\mathcal{O}(\mathrm{d}^2(\mathrm{M}+\mathrm{N}))$.

As detailed in Section 5.1.1, the proposed re-ranking method starts with the generation of hash codes for K images in the initial retrieval list. The hash code generation includes three main steps: image rotation, transformation, and hash code calculation. For an image of size $\mathrm{n} \times \mathrm{n}$, the complexity of the rotation is $\mathcal{O}(\mathrm{Kn}^2)$. The transformation step, involving the application of the DCT and DWT to b blocks, yields complexities of $\mathcal{O}(\mathrm{bKn}^2)$ and $\mathcal{O}(\mathrm{bK}(\log \mathrm{n})^2)$, respectively [104]. Lastly, calculating the hash code and finding the median for all blocks results in an additional complexity of $\mathcal{O}(\mathrm{bKn}\log \mathrm{n})$. By adding these together, we find the accumulated complexity for the hash code generation to be $\mathcal{O}(\mathrm{Kn}^2) + \mathcal{O}(\mathrm{bKn}^2) + \mathcal{O}(\mathrm{bK}(\log \mathrm{n})^2) + \mathcal{O}(\mathrm{bKn}\log \mathrm{n})$. However, among these terms, the complexity of the transformation step, $\mathcal{O}(\mathrm{bKn}^2)$, dominates as n

[1] The best-known algorithm for matrix multiplication, as of the time of writing, achieves a time complexity of $\mathcal{O}(\mathrm{n}^{2.3728596})$ [106]. Here, we approximate this as $\mathcal{O}(\mathrm{n}^2)$ for simplicity. This approximation does not affect the conclusions drawn from our analysis.

increases. Consequently, the overall time complexity of the hash code generation process simplifies to $\mathcal{O}(\mathrm{bKn}^2)$. The three groups of calculated hash codes for the K images are then used to create three trees with a complexity of $\mathcal{O}(\log \mathrm{K})$. However, in real-world scenarios, the creation of the trees can be performed in advance and offline. Therefore, given that the trees are pre-constructed and loaded, the only requirement is to calculate the initial list for a specified query image and perform a tree search, which has a computation cost of $\mathcal{O}(\mathrm{M})$ and $\mathcal{O}(\log \mathrm{K})$, respectively. Therefore, the overall computational cost for obtaining the final retrieval list is $\mathcal{O}(\mathrm{M}) + \mathcal{O}(\log \mathrm{K})$, which simplifies to $\mathcal{O}(\mathrm{M})$ (For a detailed proof, please refer to Appendix A). The above discussion shows that the contribution of the proposed re-ranking method to the overall retrieval complexity is significantly lower than that of the initial list creation. The above discussion shows that the contribution of the proposed re-ranking method in the overall retrieval complexity is significantly lower than initial list creation.

We now provide the performance of the proposed RefinerHash using the benchmark datasets, Cifar10[73], NUS-WIDE [105], MS-COCO [76], ImageNet [77], and YFCC100M [78]. The performance is compared with that of AQE [92], RCN [93], CLEBFSR [25], TXRBFSR [25], TDBG [94], and SCMR-R [95] by applying them to a common BHIR baseline retrieval technique on the five benchmark datasets. The results are reported in Table 5.1. It is seen from the table that the proposed RefinerHash gives a performance superior to that of all the other methods regardless of the dataset used and requires a computational cost that is significantly lower than required by the others. In particular, for the NUS-WIDE dataset, the proposed method gives a performance that is 10 percent higher than that of SCMR-R, the second-best performing scheme, at a computational cost that is more than seven times lower. Similarly, for the COCO dataset, RefinerHash outperforms SCMR-R, the second-best performing method, by 7.5 percent at a computational cost that is ten times lower.

We now compare our re-ranking method with all the image retrieval techniques, including our methods presented in this thesis, namely, MSFRNet, HMSRNet and MoFNet with a number of state-of-the-art image retrieval methods, including MSIR [82], PIHE [96], C-LSH [86], GreedyHash [101], XBL [85], MRDL [68], CWAH [97], DETR [98], UAIR [99], GeM [100] and AutoRet [103]. The results are reported in Table 5.2. As seen in the table, the combination of RefinerHash with other methods shows performance that is either comparable to or superior over

all other methods, regardless of whether they use re-ranking or not. It is seen from this table that when we combine the proposed hashing-based re-ranking method with our methods proposed in Chapters 3 and 4, the proposed re-ranking technique RefinerHash combined with MoFNet (Chapter 4) provides the best results than all the methods for comparison in the table. The robustness and superior performance of our proposed image retrieval technique are further confirmed through the various precision-recall curves illustrated in Figure 5.9. The curves in the figure are created by plotting 100 pairs of precision-recall points, derived from incremental 1% increases in recall along the recall-axis and their corresponding precision values. The curves are then interpolated for a cohesive visualization of performance.

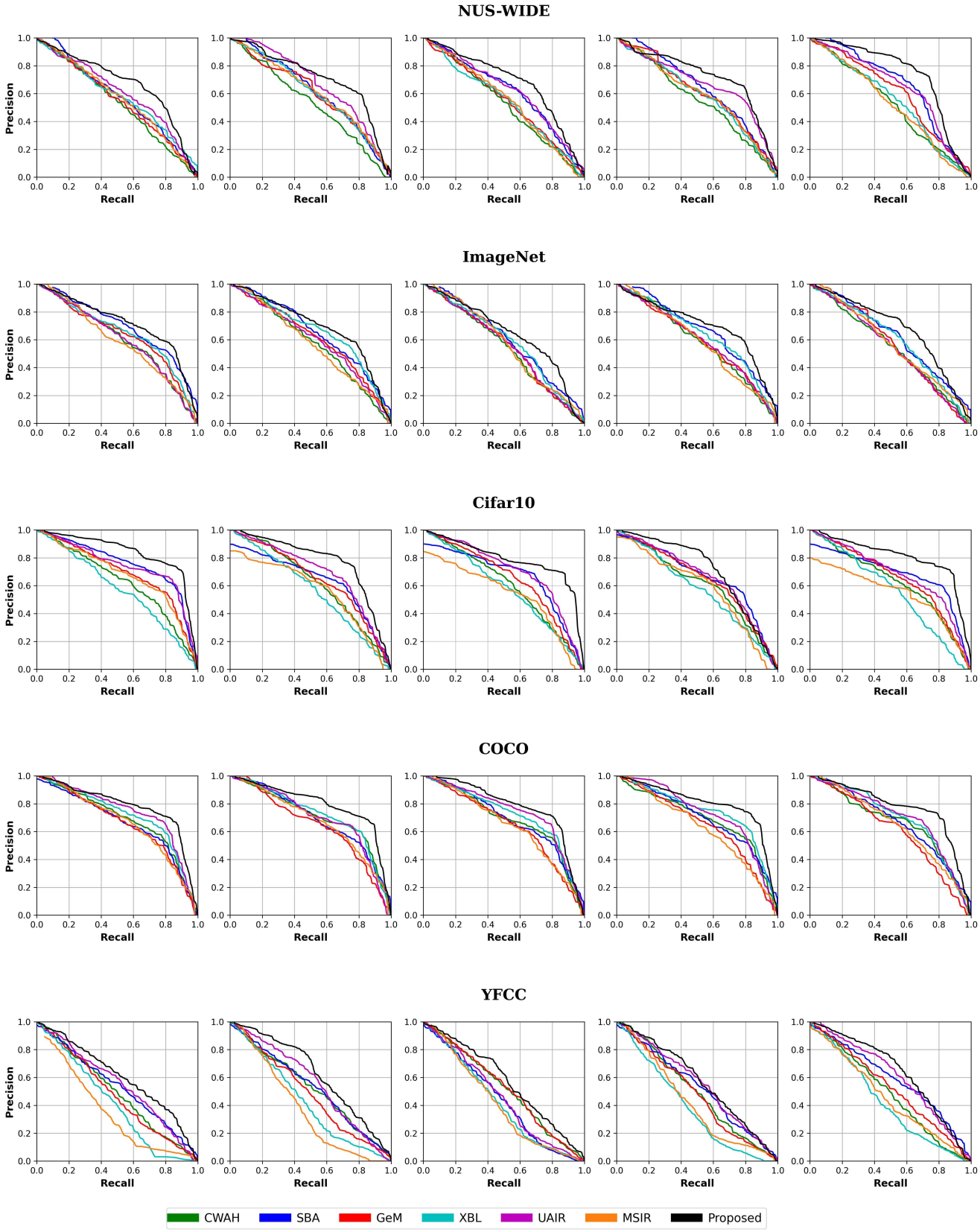

Figure 5.9: Comparative visualization of precision-recall curves for some query images.

5.3 Summary

In this chapter, we have developed a novel and computationally efficient hashing-based re-ranking technique for the task of image retrieval. Since the re-ranking of the results is an overhead for any image retrieval task, it is very important that the computational complexity of the re-ranking step be as small as possible. The main advantage of the proposed hash-based re-ranking technique lies in its ability to significantly enhance the retrieval performance of an image retrieval method at a very low computational cost. The performance of the proposed hash-based re-ranking technique has been compared with other re-ranking techniques by applying them to a common baseline retrieval technique. It has been shown that the image retrieval performance using the proposed hash-based re-ranking technique is superior to that obtained by using the other re-ranking methods at a computational cost that is several times smaller than that required by the other schemes. The retrieval performance using the proposed hash-based re-ranking technique has also been compared with a number of image retrieval techniques regardless of whether or not they employ a re-ranking technique. It has been demonstrated that employing a solution that enhances the representational capacity of a deep network and also includes the proposed hash-based re-ranking technique results in superior performance, significantly outperforming other image retrieval methods.

Chapter 6

Conclusion and Future Work

6.1 Concluding Remarks

Image retrieval is a critical function in numerous real-world scenarios. With the exponential growth in the size of image databases and the increasing complexity of image content, the pursuit of efficient and high-performing retrieval techniques remains an active research focus. With the advent of large storage devices and affordable image acquisition equipment, image retrieval has seen a surge in popularity. Yet, the growing size and diversity of image databases have heightened the complexity of the image retrieval task. Recent methods have leveraged deep learning for its superior feature extraction capabilities, often at the expense of complexity and practicability, particularly in storage and power-constrained applications. In this thesis, several low-complexity, high-performance deep convolutional neural networks for the task of image retrieval have been proposed by employing three unique strategies for generating rich, discriminating feature sets.

In Chapter 3, we have developed two methods to improve the representational capacity of a deep image retrieval network using spatial information. Recognizing the importance of spatial information in improving retrieval performance, we have proposed two novel residual blocks that primarily focus on generating feature maps enriched with spatial information. These residual blocks have been specifically designed to provide solid spatial information to the deep network through residual learning by using different scales and levels of abstraction of a deep network.

In Chapter 4, we have developed a deep image retrieval network by guiding the network with textual, structural, and edge information in order to improve the representational capacity of the networks. We have proposed a new residual block designed to guide the deep network by incorporating textural and structural information into the feature maps, enabling it to produce rich sets of features for image retrieval. Particularly, we use morphological operations to guide the network in extracting textural and structural information. In addition, we have developed a novel pooling operation for extracting the edge information in an image.

In Chapter 5 of this thesis, we have developed a novel hashing-based re-ranking technique for the task of image retrieval. Recognizing the need to minimize computational complexity due to the re-ranking being an overhead for any image retrieval task, we have proposed a low-complexity re-ranking method using image hashing to enhance the retrieval performance of an image retrieval method. The proposed hash-based re-ranking technique provides a unique approach to enhance retrieval performance by generating and utilizing multiple hash codes at a very low computational cost. The novelty of the proposed re-ranking method lies in its ability to balance the dual objectives of improving retrieval performance while maintaining computational efficiency.

The effectiveness and efficiency of the proposed image retrieval methods have been validated through a series of extensive experimental evaluations.

6.2 Future Work

In this thesis, several image retrieval methods have been developed. In light of the findings and methodologies presented in this thesis, there are several directions for future research. One such direction is the exploration and application of deep-based image hashing methods in the re-ranking techniques. The development of a real-time responsive image retrieval system based on the

computationally inexpensive methods proposed in this thesis could provide substantial contributions to this field. This direction of research would be beneficial, particularly in the context of emerging fields such as autonomous driving. Another direction for future research is the adaptation of the proposed deep image retrieval networks for use on mobile devices which usually have limited computational resources. Moreover, all the deep image retrieval networks presented are based on supervised learning, which requires labeled data for training. A potential direction for future research is to explore solutions that can adapt these deep networks to unsupervised learning settings to offer greater flexibility in various domains and applications.

References

[1] H. Ayadi, M. Torjmen-Khemakhem, and J. X. Huang, "Term dependency extraction using rule-based Bayesian Network for medical image retrieval," *Artificial Intelligence in Medicine*, vol. 140, Jun. 2023, doi: 10.1016/J.ARTMED.2023.102551.

[2] M. Hendriksen, M. Bleeker, S. Vakulenko, N. van Noord, E. Kuiper, and M. de Rijke, "Extending CLIP for Category-to-Image Retrieval in E-Commerce," in *Proc. European Conference in Information Retrieval*, 2022, pp. 289–303. doi: 10.1007/978-3-030-99736-6_20/.

[3] M. Flickner *et al.*, "Query by image and video content: The QBIC system," *Computer*, vol. 28, no. 9, pp. 23–32, 1995, doi: 10.1109/2.410146.

[4] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. IEEE International Conference on Computer Vision*, 1999, pp. 1150–1157.

[5] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *Proc. European Conference on Computer Vision*, 2006, pp. 404–417. doi: 10.1007/11744023_32.

[6] M. Yasmin, S. Mohsin, and M. Sharif, "Intelligent image retrieval techniques: a survey," *Journal of Applied Research Technology*, vol. 12, no. 1, pp. 87–103, 2014, doi: 10.1016/S1665-6423(14)71609-8.

[7] S. Wang, K. Han, and J. Jin, "Review of image low-level feature extraction methods for content-based image retrieval," *Sensor Review*, vol. 39, no. 6, pp. 783–809, Nov. 2019, doi: 10.1108/SR-04-2019-0092.

[8] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky, "Neural codes for image

retrieval," in *Proc. European Conference on Computer Vision*, 2014, pp. 584–599.

[9] A. Babenko and V. Lempitsky, "The inverted multi-index," in *Proc. IEEE International Conference on Computer Vision*, 2012, pp. 3069–3076. doi: 10.1109/CVPR.2012.6248038.

[10] A. B. Yandex and V. Lempitsky, "Aggregating local deep features for image retrieval," in *Proc. IEEE International Conference on Computer Vision*, 2015, pp. 1269–1277. doi: 10.1109/ICCV.2015.150.

[11] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 512–519. doi: 10.1109/CVPRW.2014.131.

[12] J. Wan *et al.*, "Deep learning for content-based image retrieval: a comprehensive study," in *Proc. ACM International Conference on Multimedia*, 2014, pp. 157–166. doi: 10.1145/2647868.2654948.

[13] Y. Gong, L. Wang, R. Guo, and S. Lazebnik, "Multi-scale orderless pooling of deep convolutional activation features," in *Proc. European Conference on Computer Vision*, 2014, pp. 392–407. doi: 10.1007/978-3-319-10584-0_26.

[14] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable Image Recognition," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8697–8710.

[15] Y. Li, X. Kong, L. Zheng, and Q. Tian, "Exploiting hierarchical activations of neural network for image retrieval," in *Proc. ACM International Coference on Multimedia*, 2016, pp. 132–136. doi: 10.1145/2964284.2967197.

[16] W. Zhou, H. Li, J. Sun, and Q. Tian, "Collaborative Index Embedding for Image Retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 5, pp. 1154–1166, May 2018, doi: 10.1109/TPAMI.2017.2676779.

[17] K. Ozaki and S. Yokoo, "Large-scale landmark retrieval/recognition under a noisy and diverse dataset," *arXiv preprint:190604087*, 2019.

[18] 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, 2018.

[19] O. Chum, A. Mikulík, M. Perdoch, and J. Matas, "Total recall II: query expansion revisited," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 889–896. doi: 10.1109/CVPR.2011.5995601.

[20] D. Qin, S. Gammeter, L. Bossard, T. Quack, and L. Van Gool, "Hello neighbor: accurate object retrieval with k-reciprocal nearest neighbors," in *Proc. IEEE Conference on*

Computer Vision and Pattern Recognition, 2011, pp. 777–784. doi: 10.1109/CVPR.2011.5995373.

[21] Z. Zhong, L. Zheng, D. Cao, and S. Li, "Re-Ranking person re-identification with k-reciprocal encoding," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3652–3661. doi: 10.1109/CVPR.2017.389.

[22] H. Jegou, H. Harzallah, and C. Schmid, "A contextual dissimilarity measure for accurate and efficient image search," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8. doi: 10.1109/CVPR.2007.382970.

[23] F. Wu, S. Yan, J. S. Smith, and B. Zhang, "Vehicle re-identification in still images: Application of semi-supervised learning and re-ranking," *Signal Processing: Image Communication*, vol. 76, pp. 261–271, Aug. 2019, doi: 10.1016/J.IMAGE.2019.04.021.

[24] R. Arandjelovic and A. Zisserman, "Three things everyone should know to improve object retrieval," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 2911–2918. doi: 10.1109/CVPR.2012.6248018.

[25] A. Ahmed and S. J. Malebary, "Query expansion based on top-ranked images for content-based medical image retrieval," *IEEE Access*, vol. 8, pp. 194541–194550, 2020, doi: 10.1109/ACCESS.2020.3033504.

[26] J. Ouyang, W. Zhou, M. Wang, Q. Tian, and H. Li, "Collaborative image relevance learning for visual re-ranking," *IEEE Transactions on Multimedia*, vol. 23, pp. 3646–3656, 2021, doi: 10.1109/TMM.2020.3029886.

[27] M. Rashad, I. Afifi, and M. Abdelfatah, "RbQE: An Efficient Method for Content-Based Medical Image Retrieval Based on Query Expansion," *Journal of Digital Imaging*, vol. 36, no. 3, pp. 1248–1261, Jun. 2023, doi: 10.1007/S10278-022-00769-7.

[28] D. Krotov and J. J. Hopfield, "Dense associative memory for pattern recognition," in *Proc. International Conference on Neural Information Processing Systems*, 2016, pp. 1180–1188.

[29] D. Krotov and J. Hopfield, "Large associative memory problem in neurobiology and machine learning," in *Proc. International Conference on Learning Representations*, 2020, pp. 1–13.

[30] H. Ramsauer *et al.*, "Hopfield networks is all you need," in *arXiv preprint:2008.02217*, 2020.

[31] F. Sabahi, M. O. Ahmad, and M. N. S. Swamy, "Hopfield network-based image retrieval using re-ranking and voting," in *Proc. IEEE Canadian Conference on Electrical and Computer Engineering*, 2017, pp. 1–4. doi: 10.1109/CCECE.2017.7946798.

[32] F. Sabahi, M. O. Ahmad, and M. N. S. Swamy, "Content-based image retrieval using

perceptual image hashing and hopfield neural network," in *Proc. IEEE International Midwest Symposium on Circuits and Systems*, 2018, pp. 352–355. doi: 10.1109/MWSCAS.2018.8623902.

[33] C. Wu, J. Zhu, D. Cai, C. Chen, and J. Bu, "Semi-Supervised nonlinear hashing using bootstrap sequential projection learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 6, pp. 1380–1393, Jun. 2013, doi: 10.1109/TKDE.2012.76.

[34] W. Pronobis *et al.*, "Sharing hash codes for multiple purposes," in *arXiv preprint:1609.03219*, 2017.

[35] X. Zhang, M. Wang, and J. Cui, "Efficient indexing of binary LSH for high dimensional nearest neighbor," *Neurocomputing*, vol. 213, pp. 24–33, Nov. 2016, doi: 10.1016/J.NEUCOM.2016.05.095.

[36] D. Xu, J. Wu, D. Li, Y. Tian, X. Zhu, and X. Wu, "SALE: self-adaptive LSH encoding for multi-instance learning," *Pattern Recognition*, vol. 71, pp. 460–482, 2017, doi: 10.1016/J.PATCOG.2017.04.029.

[37] L. Li, C. C. Yan, W. Ji, B. W. Chen, S. Jiang, and Q. Huang, "LSH-based semantic dictionary learning for large scale image understanding," *Journal of Visual Communication and Image Representation*, vol. 31, pp. 231–236, Aug. 2015, doi: 10.1016/J.JVCIR.2015.06.008.

[38] R. K. Karsh, A. Saikia, and R. H. Laskar, "Image authentication based on robust image hashing with geometric correction," *Multimedia Tools and Applications*, vol. 77, no. 19, pp. 25409–25429, Oct. 2018, doi: 10.1007/s11042-018-5799-6.

[39] S. Liu and Z. Huang, "Efficient image hashing with geometric invariant vector distance for copy detection," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 15, no. 4, pp. 1–22, Dec. 2019, doi: 10.1145/3355394.

[40] M. Roy, D. M. Thounaojam, and S. Pal, "A perceptual hash based blind-watermarking scheme for image authentication," *Expert Systems with Applications*, vol. 227, p. 120237, Oct. 2023, doi: 10.1016/J.ESWA.2023.120237.

[41] M. Hanif, H. Ling, W. Tian, Y. Shi, and M. Rauf, "Re-ranking person re-identification using distance aggregation of k-nearest neighbors hierarchical tree," *Multimedia Tools and Applications*, vol. 80, no. 5, pp. 8015–8038, Feb. 2021, doi: 10.1007/S11042-020-10123-0.

[42] K. H. Rosen, *Handbook of graph theory*, 2nd ed. Chapman and Hall/CRC, 2013. doi: 10.1201/B16132.

[43] H. Samet and Hanan, "The quadtree and related hierarchical data structures," *ACM Computing Surveys*, vol. 16, no. 2, pp. 187–260, Jun. 1984, doi: 10.1145/356924.356930.

[44] M. Komorowski and T. Trzciński, "Random Binary Search Trees for Approximate Nearest Neighbour Search in Binary Spaces," *Applied Soft Computing*, vol. 79, pp. 87–93, Jun. 2019, doi: 10.1016/J.ASOC.2019.03.031.

[45] P. Gupta, A. Jindal, Jayadeva, and D. Sengupta, "ComBI: Compressed Binary Search Tree for Approximate k-NN Searches in Hamming Space," *Big Data Research*, vol. 25, p. 100223, Jul. 2021, doi: 10.1016/J.BDR.2021.100223.

[46] T. Zhi, L. Y. Duan, Y. Wang, and T. Huang, "Two-stage pooling of deep convolutional features for image retrieval," in *Proc. IEEE International Conference on Image Processing*, 2016, pp. 2465–2469. doi: 10.1109/ICIP.2016.7532802.

[47] M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," in *Proc. International Conference on Learning Representations*, 2013, pp. 1–5.

[48] B. Wang, Y. Liu, W. Xiao, Z. Xiong, and M. Zhang, "Positive and negative max pooling for image classification," in *Proc. IEEE International Conference on Consumer Electronics*, 2013, pp. 278–279. doi: 10.1109/ICCE.2013.6486894.

[49] P. Sermanet, S. Chintala, and Y. LeCun, "Convolutional neural networks applied to house numbers digit classification," in *Proc. IEEE International Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3288–3291.

[50] Z. Shi, Y. Ye, and Y. Wu, "Rank-based pooling for deep convolutional neural networks," *Neural Networks*, vol. 83, pp. 21–31, Nov. 2016.

[51] D. Yu, H. Wang, P. Chen, and Z. Wei, "Mixed pooling for convolutional neural networks," in *Proc. International Conference on Rough Sets and Knowledge Technology*, 2014, pp. 364–375. doi: 10.1007/978-3-319-11740-9_34.

[52] A. Jose, R. D. Lopez, I. Heisterklaus, and M. Wien, "Pyramid pooling of convolutional feature maps for image retrieval," in *Proc. IEEE International Conference on Image Processing*, 2018, pp. 480–484. doi: 10.1109/ICIP.2018.8451361.

[53] O. Rippel, J. Snoek, and R. P. Adams, "Spectral representations for convolutional neural networks," in *Proc. International Conference on Neural Information Processing Systems*, 2015, pp. 2449–2457.

[54] J. Serra and P. Soille, *Mathematical morphology and its applications to image processing*, vol. 1. Dordrecht: Springer, 1994.

[55] F. Sabahi, M. O. Ahmad, and M. N. S. Swamy, "MorIRNet: A deep image retrieval network using morphological feature and residual block," in *Proc. IEEE International Midwest Symposium on Circuits and Systems*, 2022, pp. 1–4. doi:

10.1109/MWSCAS54063.2022.9859341.

[56] G. Franchi, A. Fehri, and A. Yao, “Deep morphological networks,” *Pattern Recognition*, vol. 102, p. 107246, Jun. 2020, doi: 10.1016/j.patcog.2020.107246.

[57] A. Esmaeilzehi, M. O. Ahmad, and M. N. S. Swamy, “SRNMSM: A Deep Light-Weight Image Super Resolution Network Using Multi-Scale Spatial and Morphological Feature Generating Residual Blocks,” *IEEE Transactions on Broadcasting*, vol. 68, no. 1, pp. 58–68, Mar. 2022, doi: 10.1109/TBC.2021.3126275.

[58] F. Sabahi, M. O. Ahmad, and M. N. S. Swamy, “Improving deep features for image retrieval using multi-source spatial information,” in *Proc. IEEE International Symposium on Circuits and Systems*, 2023, pp. 1–5. doi: 10.1109/ISCAS46773.2023.10182225.

[59] F. Sabahi, M. O. Ahmad, and M. N. S. Swamy, “Development of a deep image retrieval network using hierarchical and multi-scale spatial features,” in *Proc. IEEE International Symposium on Circuits and Systems*, 2023, pp. 1–5. doi: 10.1109/ISCAS46773.2023.10181673.

[60] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Communications*, vol. 60, no. 6, pp. 84–90, 2017, doi: 10.1145/3065386.

[61] C. Shorten and T. M. Khoshgoftaar, “A survey on image data augmentation for deep learning,” *Journal of Big Data*, vol. 60, no. 1, pp. 1–48, Dec. 2019, doi: 10.1007/978-3-319-46484-8_21.

[62] M. Abadi *et al.*, “TensorFlow: large-scale machine learning on heterogeneous systems,” 2015. https://www.tensorflow.org/

[63] F. Chollet, “GitHub - Keras-team/Keras: Deep learning for humans.”

[64] F. Yu, V. Koltun, and T. Funkhouser, “Dilated residual networks,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 636–644. doi: 10.1109/CVPR.2017.75.

[65] X. Li, J. Yang, and J. Ma, “Large scale category-structured image retrieval for object identification through supervised learning of CNN and SURF-based matching,” *IEEE Access*, vol. 8, pp. 57796–57809, 2020, doi: 10.1109/ACCESS.2020.2982560.

[66] I. Ha, H. H. H. Kim, S. Park, and H. H. H. Kim, “Image retrieval using bim and features from pretrained VGG network for indoor localization,” *International Journal of Building Science and its Applications*, vol. 140, pp. 23–31, Aug. 2018, doi: 10.1016/J.BUILDENV.2018.05.026.

[67] B. Cao, A. Araujo, and J. Sim, “Unifying deep local and global features for image search,”

in *Proc. European Conference on Computer Vision*, Aug. 2020, pp. 726–743. doi: 10.1007/978-3-030-58565-5_43.

[68] K. T. K. T. Ahmed, S. Jaffar, M. G. Hussain, S. Fareed, A. Mehmood, and G. S. Choi, "Maximum response deep learning using markov, retinal & primitive patch binding with googlenet & VGG-19 for large image retrieval," *IEEE Access*, vol. 9, pp. 41934–41957, 2021.

[69] S. Cheng, H. Lai, L. Wang, and J. Qin, "A novel deep hashing method for fast image retrieval," *The Visual Computer*, vol. 35, no. 9, pp. 1255–1266, Sep. 2019, doi: 10.1007/S00371-018-1583-X.

[70] Y. Ma, Q. Li, X. Shi, and Z. Guo, "Unsupervised deep pairwise hashing," *Electronics*, vol. 11, no. 5, pp. 744–755, 2022, doi: 10.3390/electronics11050744.

[71] R. Punithavathi, A. Ramalingam, C. Kurangi, A. S. K. Reddy, and J. Uthayakumar, "Secure content based image retrieval system using deep learning with multi share creation scheme in cloud environment," *Multimedia Tools and Applications*, vol. 80, no. 17, pp. 26889–26910, Jul. 2021, doi: 10.1007/S11042-021-10998-7.

[72] F. Sabahi, M. O. Ahmad, and M. N. S. Swamy, "A deep image retrieval network using Max-m-Min pooling and morphological feature generating residual blocks," *International Journal of Multimedia Information Retrieval*, vol. 12, no. 1, pp. 1–14, Apr. 2023, doi: 10.1007/S13735-023-00274-9.

[73] A. Krizhevsky, V. Nair, and G. Hinton, "CIFAR-10 and CIFAR-100 datasets," 2009. https://www.cs.toronto.edu/~kriz/cifar.html

[74] L. N. Darlow, E. J. Crowley, A. Antoniou, and A. J. Storkey, "CINIC-10 is not ImageNet or CIFAR-10," *arXiv preprint:1810.03505*, 2018.

[75] X. Wang, K. Yu, C. Dong, and C. C. Loy, "Recovering realistic texture in Image super-resolution by deep spatial feature transform," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 606–615. doi: 10.1109/CVPR.2018.00070.

[76] T. Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Proc. European Conference on Computer Vision*, 2014, pp. 740–755. doi: 10.1007/978-3-319-10602-1_48.

[77] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[78] B. Thomee *et al.*, "YFCC100M: the new data in multimedia research," *Communications*, vol. 59, no. 2, pp. 64–73, 2016, doi: 10.1145/2812802.

[79] Z. Cao, M. Long, J. Wang, and P. S. Yu, "HashNet: Deep learning to hash by continuation," *Proc. IEEE International Conference on Computer Vision*. pp. 5609–5618, 2017. doi:

10.1109/ICCV.2017.598.

[80] S. Eghbali and L. Tahvildari, “Deep spherical quantization for image search,” in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11682–11691. doi: 10.1109/CVPR.2019.01196.

[81] H. Zhu, M. Long, J. Wang, and Y. Cao, “Deep hashing network for efficient similarity retrieval,” in *Proc. AAAI Conference on Artificial Intelligence*, 2016, pp. 2415–2421. doi: 10.1609/aaai.v30i1.10235.

[82] H. Zhu, “Massive-scale image retrieval based on deep visual feature representation,” *Journal of Visual Communication and Image Representation*, vol. 70, p. 102743, 2020.

[83] R. Wang, R. Wang, S. Qiao, S. Shan, and X. Chen, “Deep position-aware hashing for semantic continuous image retrieval,” in *Proc. IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 2482–2491. doi: 10.1109/WACV45572.2020.9093468.

[84] J. Xu, C. Guo, Q. Liu, J. Qin, Y. Wang, and L. Liu, “DHA: Supervised deep learning to hash with an adaptive loss function,” in *Proc. IEEE/CVF International Conference on Computer Vision Workshop*, 2019, pp. 3054–3062. doi: 10.1109/ICCVW.2019.00368.

[85] S. Gkelios, A. Sophokleous, S. Plakias, Y. Boutalis, and S. A. Chatzichristofis, “Deep convolutional features for image retrieval,” *Expert Systems with Applications*, vol. 177, p. 114940, 2021, doi: 10.1016/j.eswa.2021.114940.

[86] M. Ghayoumi, M. Gomez, K. E. Baumstein, N. Persaud, and A. J. Perlowin, “Local sensitive hashing (LSH) and convolutional neural networks (CNNs) for object recognition,” in *Proc. IEEE International Conference on Machine Learning and Applications*, 2018, pp. 1197–1199. doi: 10.1109/ICMLA.2018.00193.

[87] Y. Shen *et al.*, “Auto-encoding twin-bottleneck hashing,” in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2815–2824. doi: 10.1109/CVPR42600.2020.00289.

[88] F. Sabahi, M. O. Ahmad, and M. N. S. Swamy, “Perceptual image hashing using random forest for content-based image retrieval,” in *Proc. IEEE International New Circuits and Systems Conference*, 2018, pp. 348–351. doi: 10.1109/NEWCAS.2018.8585506.

[89] F. Sabahi, M. O. Ahmad, and M. N. S. Swamy, “RefinerHash: A new hashing-based re-ranking technique for image retrieval,” *under review*.

[90] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[91] P. Govindaraj and R. Sandeep, “Ring partition and dwt based perceptual image hashing

with application to indexing and retrieval of near-identical images," in *Proc. IEEE International Conference on Advances in Computing and Communication*, 2016, pp. 421–425. doi: 10.1109/ICACC.2015.90.

[92] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman, "Total recall: automatic query expansion with a generative feature model for object retrieval," in *Proc. IEEE International Conference on Computer Vision*, 2007, pp. 1–8. doi: 10.1109/ICCV.2007.4408891.

[93] F. Ye, M. Dong, W. Luo, X. Chen, and W. Min, "A new re-ranking method based on convolutional neural network and two image-to-class distances for remote sensing image retrieval," *IEEE Access*, vol. 7, pp. 141498–141507, 2019, doi: 10.1109/ACCESS.2019.2944253.

[94] G. Lao *et al.*, "Three degree binary graph and shortest edge clustering for re-ranking in multi-feature image retrieval," *Journal of Visual Communication and Image Representation*, vol. 80, p. 103282, Oct. 2021, doi: 10.1016/J.JVCIR.2021.103282.

[95] L. Wang, X. Qian, X. Zhang, and X. Hou, "Sketch-based image retrieval with multi-clustering re-ranking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 12, pp. 4929–4943, Dec. 2020, doi: 10.1109/TCSVT.2019.2959875.

[96] X. Yuan and Y. Zhao, "Perceptual image hashing based on three-dimensional global features and image energy," *IEEE Access*, vol. 9, pp. 49325–49337, 2021, doi: 10.1109/ACCESS.2021.3069045.

[97] F. Lu and G. H. Liu, "Image retrieval using contrastive weight aggregation histograms," *Digital Signal Processing*, vol. 123, p. 103457, Apr. 2022, doi: 10.1016/J.DSP.2022.103457.

[98] C. G. Ban, Y. Hwang, D. Park, R. Lee, R. Y. Jang, and M. S. Choi, "Multi-Subject Image Retrieval by Fusing Object and Scene-Level Feature Embeddings," *Applied Sciences*, vol. 12, no. 24, p. 12705, Dec. 2022, doi: 10.3390/APP122412705.

[99] L. Huang, C. Bai, Y. Lu, S. Zhang, and S. Chen, "Unsupervised adversarial image retrieval," *Multimedia Systems*, vol. 28, no. 2, pp. 673–685, 2022, doi: 10.1007/S00530-021-00866-7.

[100] F. Radenovic, 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, 2019.

[101] S. Su, C. Zhang, K. Han, and Y. Tian, "Greedy hash: towards fast optimization for accurate hash coding in CNN," in *Proc. International Conference on Neural Information Processing Systems*, 2018, pp. 806–815. doi: 10.5555/3326943.3327018.

[102] J. Xu, C. Wang, C. Qi, C. Shi, and B. Xiao, "Unsupervised semantic-based aggregation of

deep convolutional features," *IEEE Transactions on Image Processing*, vol. 28, no. 2, pp. 601–611, Feb. 2019, doi: 10.1109/TIP.2018.2867104.

[103] M. M. Monowar, M. A. Hamid, A. Q. Ohi, M. O. Alassafi, and M. F. Mridha, "AutoRet: a self-supervised spatial recurrent network for content-based image retrieval," *Sensors*, vol. 22, no. 6, p. 2188, Mar. 2022, doi: 10.3390/S22062188.

[104] S. H. Ieng, E. Lehtonen, and R. Benosman, "Complexity analysis of iterative basis transformations applied to event-based signals," *Frontiers in Neuroscience*, vol. 12, p. 373, Jun. 2018.

[105] P. Zhang, W. Zhang, W.-J. Li, and M. Guo, "Supervised hashing with latent factor models," in *Proc. ACM International Conference on Research and Development in Information Retrieval*, 2014, pp. 173–182. doi: 10.1145/2600428.2609600.

[106] J. Alman and V. V. Williams, "A refined laser method and faster matrix multiplication," in *Proc. ACM-SIAM Symposium Discrete Algorithms*, 2021, pp. 522–539. doi: 10.1137/1.9781611976465.32.

Appendix A: Proofs

Proposition 1: Given $M \geq K > 0$, the limit of the ration of φ_2 and φ_1 as M approaches infinity converges to zero, where $\varphi_1 \in \mathcal{O}(M)$ and $\varphi_2 \in \mathcal{O}(\log K)$.

Proof. Let $\varphi_1 = C_1 M$ and $\varphi_2 = C_2 log\ K$, where C_1 and C_2 are positive constants representing the rate of growth of each function. Through analysis of the limit of the ratio of φ_1 and φ_2 as M approaches infinity, we have

$$\lim_{M\to\infty} \frac{\varphi_2}{\varphi_1} = \lim_{M\to\infty} \frac{C_2 \log K}{C_1 M} \tag{A.16}$$

Since $\log K$ grows slower than M, as M approaches infinity, the limit converges to zero:

$$\lim_{M\to\infty} \frac{C_2 \log K}{C_1 M} = 0 \tag{A.17}$$

∴ Asymptotically, the growth of φ_2 is negligible compared to the growth of φ_1. □

Proposition 2: Time complexity of $\mathcal{O}(M) + \mathcal{O}(\log K)$ is $\mathcal{O}(M)$ for $M \geq K > 0$.

Proof. Let function φ be the sum of two other functions, φ_1 and φ_2, such that $\varphi = \varphi_1 + \varphi_2$, where $\varphi_1 \in \mathcal{O}(M)$ and $\varphi_2 \in \mathcal{O}(\log K)$. Then we can express the upper bound of $\varphi_1$and φ_2 as

$$\begin{aligned} &\exists C_1 > 0, \exists N_1 \in \mathbb{N}, \forall n > N_1, \varphi_1 \leq C_1 M \\ &\exists C_2 > 0, \exists N_2 \in \mathbb{N}, \forall n > N_2, \varphi_2 \leq C_2 \log K. \end{aligned} \tag{A.18}$$

We have φ as

$$\varphi = \varphi_1 + \varphi_2, \tag{A.19}$$

Then, by applying the upper bounds for φ_1 and φ_2 from Eq. 22, we can derive an upper bound for φ as

$$\varphi = \varphi_1 + \varphi_2 \leq C_1 M + C_2 \log K \tag{A.20}$$

or,

$$\varphi \leq M \left(C_1 + C_2 \frac{\log K}{M} \right). \tag{A.21}$$

Now, based on Proposition 1, as M grows large, the term C_2 becomes negligible compared to $C_1 M$. Therefore, the dominant term here is $C_1 M$, leading us to express the upper bound of φ as:

$$C_1 + C_2 \frac{\log K}{M} \leq C_1. \tag{A.22}$$

or

$$\varphi \leq C_1 M \tag{A.23}$$

So, the time complexity $\mathcal{O}(M) + \mathcal{O}(log\ K)$ is dominated by $\mathcal{O}$(M), which leads us to conclude that the overall time complexity is $\mathcal{O}$(M). □