Ph.D. DISSERTATION

Deep Content-based Image Retrieval Under Various Learning Conditions

다양한 딥 러닝 학습 환경 하의 컨텐츠 기반 이미지 검색

BY

JANG YOUNG KYUN

FEBRUARY 2022

DEPARTMENT OF ELECTRICAL AND
COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Ph.D. DISSERTATION

Deep Content-based Image Retrieval Under Various Learning Conditions

다양한 딥 러닝 학습 환경 하의 컨텐츠 기반 이미지 검색

BY

JANG YOUNG KYUN

FEBRUARY 2022

DEPARTMENT OF ELECTRICAL AND
COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Deep Content-based Image Retrieval Under Various Learning Conditions

다양한 딥 러닝 학습 환경 하의 컨텐츠 기반 이미지 검색

지도교수 조 남 익

이 논문을 공학박사 학위논문으로 제출함

2021 년 12 월

서울대학교 대학원

전기·정보공학부

장 영 균

장영균의 공학박사 학위논문을 인준함

2021 년 12 월

위 원 장 _____

부위원장 _____

위 원 _____

위 원 _____

위 원 _____

Abstract

Content-based image retrieval, which finds relevant images to a query from a huge database, is one of the fundamental tasks in the field of computer vision. Especially for conducting fast and accurate retrieval, Approximate Nearest Neighbor (ANN) search approaches represented by Hashing and Product Quantization (PQ) have been proposed to image retrieval community. Ever since neural network based deep learning has shown excellent performance in many computer vision tasks, both Hashing and product quantization-based image retrieval systems are also adopting deep learning for improvement. In this dissertation, image retrieval methods under various deep learning conditions are investigated to suggest the appropriate retrieval systems. Specifically, by considering the purpose of image retrieval, the supervised learning methods are proposed to develop the deep Hashing systems that retrieve semantically similar images, and the semi-supervised, unsupervised learning methods are proposed to establish the deep product quantization systems that retrieve both semantically and visually similar images. Moreover, by considering the characteristics of image retrieval database, the face image sets with numerous class categories, and the general image sets of one or more labeled images are separated to be explored when building a retrieval system.

First, supervised learning with the semantic labels given to images is introduced to build a Hashing-based retrieval system. To address the difficulties of distinguishing face images, such as the inter-class similarities (similar appearance between different persons) and the intra-class variations (same person with different pose, facial expressions, illuminations), the identity label of each image is employed to derive the discriminative binary codes. To further develop the face image retrieval quality, Similarity Guided Hashing (SGH) scheme is proposed, where the self-similarity learning with multiple data augmentation results are employed during training. In terms of Hashing-based general image retrieval systems, Deep Hash Distillation (DHD) scheme

is proposed, where the trainable hash proxy that presents class-wise representative is introduced to take advantage of supervised signals. Moreover, self-distillation scheme adapted for Hashing is utilized to improve general image retrieval performance by exploiting the potential of augmented data appropriately.

Second, semi-supervised learning that utilizes both labeled and unlabeled image data is investigated to build a PQ-based retrieval system. Even if the supervised deep methods show excellent performance, they do not meet the expectations unless expensive label information is sufficient. Besides, there is a limitation that a tons of unlabeled image data is excluded from training. To resolve this issue, the vector quantization-based semi-supervised image retrieval scheme: Generalized Product Quantization (GPQ) network is proposed. A novel metric learning strategy that preserves semantic similarity between labeled data, and a entropy regularization term that fully exploits inherent potentials of unlabeled data are employed to improve the retrieval system. This solution increases the generalization capacity of the quantization network, which allows to overcome previous limitations.

Lastly, to enable the network to perform a visually similar image retrieval on its own without any human supervision, unsupervised learning algorithm is explored. Although, deep supervised Hashing and PQ methods achieve the outstanding retrieval performances compared to the conventional methods by fully exploiting the label annotations, however, it is painstaking to assign labels precisely for a vast amount of training data, and also, the annotation process is error-prone. To tackle these issues, the deep unsupervised image retrieval method dubbed Self-supervised Product Quantization (SPQ) network, which is label-free and trained in a self-supervised manner is proposed. A newly designed Cross Quantized Contrastive learning strategy is applied to jointly learn the PQ codewords and the deep visual representations by comparing individually transformed images (views). This allows to understand the image content and extract descriptive features so that the visually accurate retrieval can be performed.

By conducting extensive image retrieval experiments on the benchmark datasets, the proposed methods are confirmed to yield the outstanding results under various evaluation protocols. For supervised face image retrieval, SGH achieves the best retrieval performance for both low and high resolution face image, and DHD also demonstrates its efficiency in general image retrieval experiments with the state-of-the-art retrieval performance. For semi-supervised general image retrieval, GPQ shows the best search results for protocols that use both labeled and unlabeled image data. Finally, for unsupervised general image retrieval, the best retrieval scores are achieved with SPQ even without supervised pre-training, and it can be observed that visually similar images are successfully retrieved as search results.

keywords: Image retrieval, Face image retrieval, Convolutional Neural Network, Deep learning, Approximate nearest neighbor search, Hashing, Product quantization, Metric learning, Semi-supervised learning, Unsupervised learning

student number: 2016-20967

Contents

Abstract **i**

Contents **iv**

List of Tables **vii**

List of Figures **viii**

1 Introduction **1**

1.1 Contribution 3

1.2 Contents 4

2 Supervised Learning for Deep Hashing: Similarity Guided Hashing for Face Image Retrieval / Deep Hash Distillation for General Image Retrieval **5**

2.1 Motivation and Overview for Face Image Retrieval 5

2.1.1 Related Works 9

2.2 Similarity Guided Hashing 10

2.3 Experiments 16

2.3.1 Datasets and Setup 16

2.3.2 Results on Small Face Images 18

2.3.3 Results on Large Face Images 19

2.4 Motivation and Overview for General Image Retrieval 20

2.5 Related Works 22

2.6 Deep Hash Distillation 24
2.6.1 Self-distilled Hashing 24
2.6.2 Teacher loss 27
2.6.3 Training 29
2.6.4 Hamming Distance Analysis 29
2.7 Experiments 32
2.7.1 Setup 32
2.7.2 Implementation Details 32
2.7.3 Results 34
2.7.4 Analysis 37

3 Semi-supervised Learning for Product Quantization: Generalized Product Quantization Network for Semi-supervised Image Retrieval 42
3.1 Motivation and Overview 42
3.1.1 Related Work 45
3.2 Generalized Product Quantization 47
3.2.1 Semi-Supervised Learning 48
3.2.2 Retrieval 52
3.3 Experiments 53
3.3.1 Setup 53
3.3.2 Results and Analysis 55

4 Unsupervised Learning for Product Quantization: Self-supervised Product Quantization for Deep Unsupervised Image Retrieval 58
4.1 Motivation and Overview 58
4.1.1 Related Works 61
4.2 Self-supervised Product Quantization 62
4.2.1 Overall Framework 62
4.2.2 Self-supervised Training 64

4.3 Experiments 67
4.3.1 Datasets 67
4.3.2 Experimental Settings 68
4.3.3 Results 71
4.3.4 Empirical Analysis 71

5 Conclusion **75**

Abstract (In Korean) **88**

List of Tables

2.1 Dataset configuration used to perform face image retrieval experiments. Ids and Resol. are short for identities and resolution, respectively. . . . 16

2.2 mAP scores of different Hashing approaches on small face image datasets. 18

2.3 mAP scores of different Hashing approaches on large face image dataset. 19

2.4 Description of the image retrieval datasets. 32

2.5 mAP scores for different bits on three benchmark image datasets. . . . 33

2.6 Ablation study on my work with ImageNet. ✓indicates that the corresponding element is used. $\mathcal{T}$ denotes types of augmentation, and $Eud.$ is the abbreviation of Euclidean distance. 35

2.7 mAP scores with or without Self-distilled Hashing (SdH). 37

2.8 mAP scores on unseen deformations. 39

3.1 Detailed composition of two benchmark datasets. 54

3.2 mAP scores of different Hashing algorithms on experimental protocol 1. 55

3.3 mAP scores of different Hashing algorithms on experimental protocol 2. 55

4.1 Detailed composition of three benchmark datasets. 67

4.2 mAP scores of different retrieval methods on three benchmark datasets. 68

4.3 mAP scores of SPQ and its variants on three benchmark datasets @ 32-bits. 72

List of Figures

1.1 Comparison of distance calculation. 2

2.1 Example of inter-identity resemblance and intra-identity difficulties. . 6

2.2 A simple visualization of proposed method. $\tilde{x}_1, \tilde{x}_2, \tilde{x}_3$ are the transformed output of x_1, x_2, x_3, respectively, and I only illustrate the case of x_1 for simplicity. x_1 forms a positive pair with $\tilde{x}_1$, and $\tilde{x}_3$ derived from the same identity, and forms a negative pair with $\tilde{x}_2$ derived from the different identity. With this similarity information learned in the latent space, the final output hash codes $\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3$ which are originated from x_1, x_2, x_3, respectively, can preserve pairwise similarity near the Hamming space (cubic) to achieve high retrieval performance. 7

2.3 The overall training process of SGH. 10

2.4 Examples of face images with various augmentation techniques applied. 12

2.5 Block Hashing layer. 14

2.6 An illustration of different quantization loss functions. 15

2.7 Visualization of possible problems with deep hashing due to transformations. The continuous hash code generated by a deep hashing model $\mathcal{H}(\cdot)$ is changed when the input is transformed. In consequence, the binary code quantized with sign operation can also be shifted. However, the degree of transformation cannot be properly reflected in the quantized representation. 21

2.8 The overall training process of Deep Hash Distillation (DHD) framework. (Left) From two different augmentation groups: Teacher $\mathcal{T}_T$ and Student $\mathcal{T}_S$, randomly sampled transformations ($t_T \sim \mathcal{T}_T$, $t_S \sim \mathcal{T}_S$) are individually applied on the input image x to produce $\tilde{x}_T$ and $\tilde{x}_S$. The deep hashing model $\mathcal{H}(\cdot)$ constructed with the deep encoder $E_\theta(\cdot)$ and the hash function $H_\theta(\cdot)$ of one Fully-Connected (FC) layer and Layer Normalization (LN) yields two hash codes h_T and h_S which are learned with $\mathcal{L}_{SdH}$. I apply stop gradient operation on h_T for stable training. (Right) Additionally, I employ trainable hash proxies $P_\theta(\cdot)$ which are used to calculate the class-wise prediction p_T with h_T to optimize with $\mathcal{L}_{HP}$, and pre-defined Gaussian estimator $G(\cdot)$ to regularize h_T with $\mathcal{L}_{bce\text{-}Q}$. 25

2.9 Precision-Recall curves on benchmarks with binary codes @ 64-bits. . 35

2.10 Precision@top-1000 curves on benchmarks with binary codes @ 64-bits. 35

2.11 Average Hamming distance difference between the original and transformed images of ImageNet query set. By varying the s_T, I measure the sensitivity to transformation of ResNet backbone methods where the numbers in legend indicate mAP. Solid lines present DHD variants, and dotted lines present others. $+\mathcal{T}_S$ denotes strong student augmentation is applied during training. A low slope indicates insensitivity to various transformations. 36

2.12 3D visualized histograms to verify the impact of $\mathcal{L}_{bce\text{-}Q}$. Gallery sets of (i) ImageNet, (ii) NUS-WIDE, and (iii) MS COCO are utilized. x-axis presents value of hash element h, y-axis presents bit position, and z-axis presents frequency counts. 38

2.13 Visualized augmentation results of different groups: weakly-transformed teacher $\mathcal{T}_T$ and strongly-transformed student $\mathcal{T}_S$. 40

2.14 Above: Examples of various transformations applied to the original image. The green box indicates the same search result as the original, and the red box indicates otherwise. Below: Retrieved images on ImageNet dataset. The image output form the strongly transformed $\mathcal{T}_T$ and the image transformed to gray scale show different retrieval results. Nevertheless, it can be seen that visually similar images of the same content are retrieved well. 41

3.1 Above: an illustration of the overall framework of GPQ and its three components: feature extractor F, PQ table Z, and classifier C, where C contributes to build Z. 43

3.2 A two class ($+$: blue, $-$: orange) visualized examples of my training objectives. 1) The left part shows the learning process of N-pair Product Quantization loss $\mathcal{L}_{N\text{-}PQ}$. When I define an anchor as $\hat{\mathbf{x}}_1^+$, the semantically similar points $(\hat{\mathbf{q}}_1^+, \hat{\mathbf{x}}_4^+, \hat{\mathbf{q}}_4^+)$ are pulled together while the semantically dissimilar points $(\hat{\mathbf{x}}_2^-, \hat{\mathbf{q}}_2^-, \hat{\mathbf{x}}_3^-, \hat{\mathbf{q}}_3^-)$ are pushing the anchor. 2) The right part shows the learning process of classification loss $\mathcal{L}_{cls}$ and subspace entropy mini-max loss $\mathcal{L}_{SEM}$. For the data points constrained on the unit hypersphere, the cross entropy of labeled data points is minimized to find prototypes (white stars). Then, the entropy between the prototypes and the unlabeled data points is maximized to move prototypes toward unlabeled data points and find new prototypes (yellow stars). Finally, the entropy of the unlabeled data points is minimized to cluster them near the new prototypes. 48

3.3 The comparison results of GPQ and its variants. 56

3.4 The sensitivity investigation of two balancing parameters: λ_1 and λ_2. . 57

3.5 The t-SNE visualization of deep representations learned by GPQ-T, GPQ-H and GPQ on CIFAR-10 dataset respectively. 57

4.1 A simple framework comparison between contrastive learning (a) and cross quantized contrastive learning (b). The separately sampled two transformations ($t, t' \sim \mathcal{T}$) are applied on an image x to generate two different views $\tilde{x}_i$ and $\tilde{x}_j$, and corresponding deep descriptor $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{x}}_j$ are obtained with the feature extractor $\mathcal{F}(\cdot)$, respectively. The feature representations in contrastive learning are achieved by comparing the similarity between the projection head outputs $\hat{\mathbf{z}}_i$ and $\hat{\mathbf{z}}_j$. Instead of projection, I introduce the quantization head, which collects codebooks of product quantization. By maximizing cross-similarity between the deep descriptor of one view and the product quantized descriptor of the other, both codewords and deep descriptors are jointly trained to contain discriminative image content representations. 59

4.2 An illustration of feature extraction and quantization procedure in SPQ. Randomly sampled data augmentation techniques ($t_n \sim T$) are applied on x_1 and x_2 to produce transformed images (different views). There are two trainable components; 1) CNN-based feature extractor $\mathcal{F}$, and 2) quantization head $\mathcal{Q}$, which collects multiple codebooks to conduct product quantization. For example, I set up two codebooks C_1 and C_2, and illustrate 2D conceptual Voronoi diagram in $\mathcal{Q}$. The original feature space of deep descriptor (feature vector $\hat{\mathbf{x}}_n \in R^D$) is divided into two subspaces and generates subvectors; $\mathbf{x}_{nk}$ where $k = \{1, 2\}$ and $\mathbf{x}_{nk} \in R^{D/2}$. By employing soft quantizer $q_k(\cdot)$ on each $\mathbf{x}_{nk}$, the sub-quantized descriptor $\mathbf{z}_{nk} = q_k(\mathbf{x}_{nk})$ is approximated with the combination of the codewords. Notably, subvectors representing similar features are allocated to the same codeword. The output product quantized descriptor $\hat{\mathbf{z}}_n \in R^D$ is obtained by $[\cdot]_D$ operation which concatenates the sub-quantized descriptors along the D-dimension. 63

4.3 A visualized example of proposed cross quantized contrastive learning strategy. For simplicity, I take the examples $x_1 \mapsto \{\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2\}$, and $x_2 \mapsto \{\hat{\mathbf{x}}_3, \hat{\mathbf{x}}_4, \hat{\mathbf{z}}_3, \hat{\mathbf{z}}_4\}$ from Figure 2.2 LaTeX Error: Can be used only in preambleSee the LaTeX manual or LaTeX Companion for explanation.Your command was ignored.Type I ¡command¿ ¡return¿ to replace it with another command,or ¡return¿ to continue without it.2.2, and paint the feature representations related to x_1 in blue, and x_2 in red. Taking into account the cross-similarity between $\hat{\mathbf{x}}$ and $\hat{\mathbf{z}}$ as: $\hat{\mathbf{x}}_1 \leftrightarrow \{\hat{\mathbf{z}}_2, \hat{\mathbf{z}}_4\}$, $\hat{\mathbf{x}}_2 \leftrightarrow \{\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_3\}$, $\hat{\mathbf{x}}_3 \leftrightarrow \{\hat{\mathbf{z}}_2, \hat{\mathbf{z}}_4\}$,and $\hat{\mathbf{x}}_4 \leftrightarrow \{\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_3\}$, the network is trained to understand the discriminative image contents, while simultaneously collecting frequently occurring local patterns into the codewords. 64

4.4 Precision-Recall curves on benchmarks with binary codes @ 64-bits. . 70

4.5 Precision@top-1000 curves on benchmarks with binary codes @ 64-bits. 70

4.6 t-SNE visualization of deep representations learned by BinGAN, TBH, and SPQ on CIFAR-10 query set respectively. 72

4.7 Sensitivity investigation of two temperature hyper-parameters: τ_q and τ_{cqc} on CIFAR-10 dataset. 73

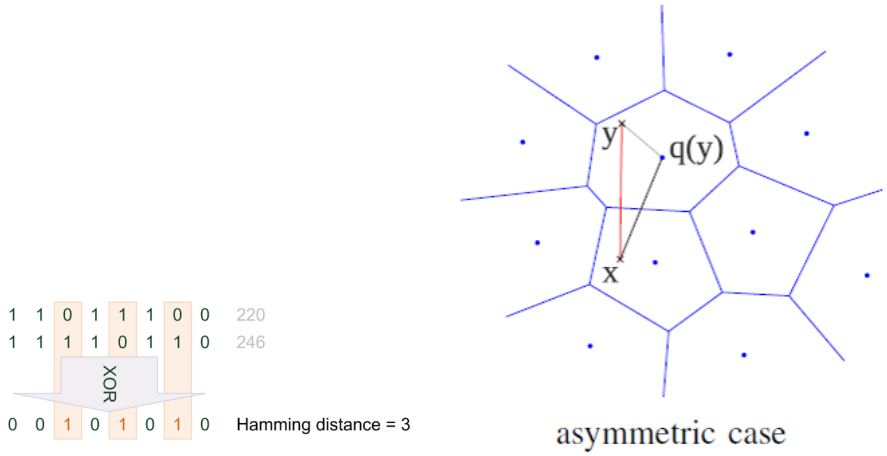
4.8 SPQ retrieval results on CIFAR-10 @ 32-bits. 74

Chapter 1

Introduction

The amount of multimedia data, including images and videos, increases exponentially on a daily basis. Hence, retrieving relevant content from a large-scale database has become a more complicated problem. There have been many kinds of fast and accurate search algorithms, and the Approximate Nearest Neighbor (ANN) search is known to have high retrieval accuracy and computational efficiency. Recent ANN methods mainly focused on deep Hashing and product quantization based schemes [27], because of their low storage cost and fast retrieval speed. To be specific, an image is represented by a binary-valued compact hash code (*binary code*) with only a few tens of bits, and it is utilized to build a database and compute distances for ranking.

The fast image retrieval methods using bit-wise binary code representation can be categorized as *Hashing* and *Product Quantization* (PQ) [45]. Hashing-based methods employ a hash function that maps a high-dimensional vector space to a Hamming space, where the distance between two codes can be measured extremely fast via bit-wise XOR operation. However, Hashing has a limitation in describing the distance between data points because it can produce only a limited number of distinct values. PQ, which is a kind of vector quantization approach that enables to compute distance between the binary representation in the real valued space, has been introduced to alleviate this problem in information retrieval.



(a) Hashing (b) Vector Quantization

Figure 1.1: Comparison of distance calculation.

PQ is performed by decomposing the input feature space into a Cartesian product of several disjoint subspaces (*codebooks*) and find the centroid (*codeword*) of each subspace. Then, from the sub-vectors of the input feature vector, sub-binary code is obtained by replacing each sub-vector with the index of the nearest codeword in the codebook. Since codeword consists of real numbers, PQ allows asymmetric distance calculation in real space using the binary codes, resulting in richer distance representations than Hashing as shown in Figure 1.1.

In comparison, Hashing has the advantage of using discrete and compact binary code to perform fast retrieval, and PQ has the advantage of utilizing complex distance representations to perform accurate retrieval with comparable speed. Briefly, Hashing shows relatively fast search speed and low accuracy, whereas PQ shows slow search speed and high accuracy. According to this analysis, in this dissertation, Hashing is employed for semantically similar image retrieval for face and general image sets, which requires distinctiveness between hash codes of a large number of categories. Besides, PQ is considered for building image retrieval system that searches visually similar general images in which various contents exist with multi label annotations.

Specifically, the proposed methods aim to properly integrate hashing and PQ into the image retrieval system which is configured with the deep learning model. Under the various deep model training conditions, supervised learning that utilizes a training set where the category label information is annotated to each image is introduced to face and general image retrieval systems. Further, to utilize the unlabeled data for the deep model training, semi-supervised and unsupervised learning methods are proposed to construct a general image retrieval system.

1.1 Contribution

In this dissertation, several solutions that can be taken when constructing content-based image retrieval system using deep learning are proposed. First, novel deep hashing methods with supervised learning are proposed for a system that aims to search for semantically similar images at the top. In Similarity Guided Hashing (SGH), a new non-trainable Block Hashing and a metric learning strategy that uses differently augmented images are proposed to obtain discriminative binary hash codes for face image retrieval. In Deep Hash Distillation (DHD), a method that transforms self-distillation to fit deep hashing learning is proposed so that strong data augmentation can be used for hash code learning while hash code acts as binary-like and discriminative.

Second, in order to utilize both labeled and unlabeled data for deep image retrieval model training, semi-supervised learning method is proposed. Although the vector quantization-based ANN search method is a little slower than the Hashing-based one, it has the advantage that the distance can be expressed diversely, so PQ is used for both visually and semantically similar retrieval. In Generalized Product Quantization (GPQ), a novel metric learning scheme that learns PQ codewords and the deep image representation simultaneously with the labeled data, and an entropy mini-max loss function that regularizes labeled data points with unlabeled ones are introduced to fully exploit the inherent potentials of input data.

Lastly, an unsupervised learning-based image retrieval method that can exclude labeled data from training. Since obtaining labeled data is expensive and can cause serious performance degradation if mislabeled, self-supervised fashioned unsupervised training is proposed in Self-supervised Product Quantization (SPQ). This has the advantage of deriving a deep model to generate a image representation that discriminates the content of an image by contrasting data augmentation results, allowing semantically and visually similar image retrieval without other human annotations.

1.2 Contents

The rest of the dissertation is organized as follows. Chapter 2 introduces supervised learning to build two separate Hashing based image retrieval systems, one is SGH approach for face image retrieval, and the other is DHD method for general image retrieval. Chapter 3 presents semi-supervised learning to configure PQ based image retrieval system, GPQ strategy for general image retrieval. Chapter 4 demonstrates self-supervised learning to construct PQ based image retrieval system, SPQ scheme for general image retrieval. Finally, chapter 5 concludes this dissertation.

Chapter 2

Supervised Learning for Deep Hashing: Similarity Guided Hashing for Face Image Retrieval / Deep Hash Distillation for General Image Retrieval

2.1 Motivation and Overview for Face Image Retrieval

Learning to hash for image retrieval has made significant progress since the introduction of deep learning. In particular, many researchers attempted to produce compact binary hash codes through supervised learning with image class labels [19, 34, 17, 2, 10], which are shown to provide superior performance to existing methods. Face image retrieval is also being advanced using deep learning [25, 18, 26, 44, 31] to learn facial representations of each identity.

Face image retrieval can be considered a sub-problem of general image retrieval, finding images of the same identity (class label) to the query. Still, face retrieval has subtle differences from the general task due to some particular properties of facial image datasets. To be specific, 1) the variance of the data distribution within a class is often high due to makeup, facial expression, glasses, view-points, etc. (intra-identity difficulties), 2) the similarity between the two different classes is comparatively high because there are many similar faces (inter-identity resemblance), and 3) there are

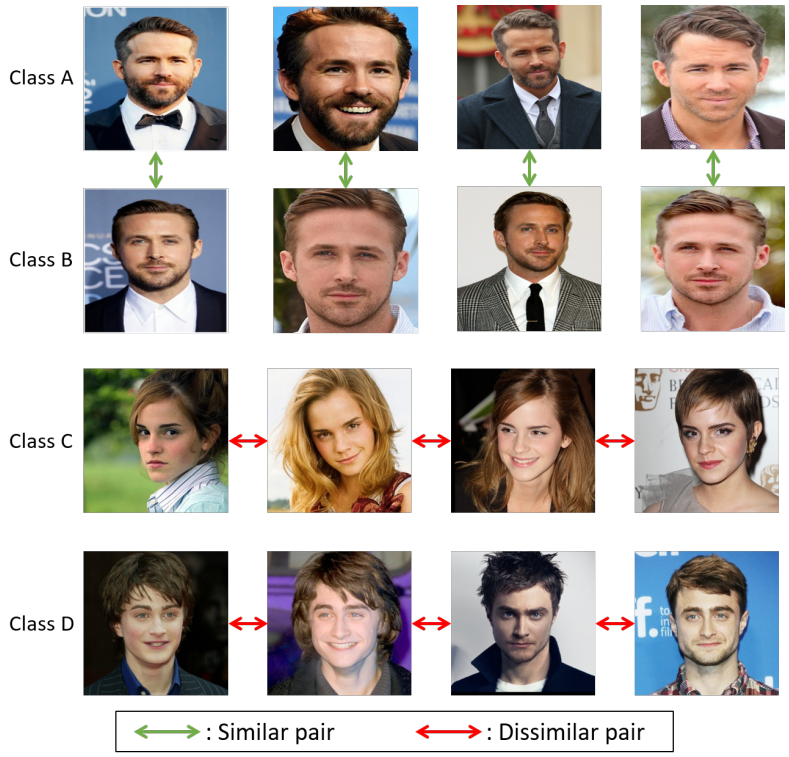


Figure 2.1: Example of inter-identity resemblance and intra-identity difficulties.

relatively many classes to distinguish. Therefore, deep network models for face image retrieval attempt to increase retrieval performance by increasing discriminativity among binary hash codes assigned to each identity.

For this purpose, existing deep face image retrieval methods usually employ common classification losses such as cross-entropy, and utilize the intermediate feature vector as a hash code. Although this approach provides moderate retrieval performance, there exists a limitation in representing the semantic similarity between images because the hash codes are learned only with discrete labels (0 and 1 if one-hot encoded). Therefore, if the similarity between face images is elaborately considered in hash code generation, the retrieval performance can be improved.

To consider similarity more precisely, I refer self-supervised learning [11, 33, 4, 12, 15], which has been widely explored for diverse image feature representa-

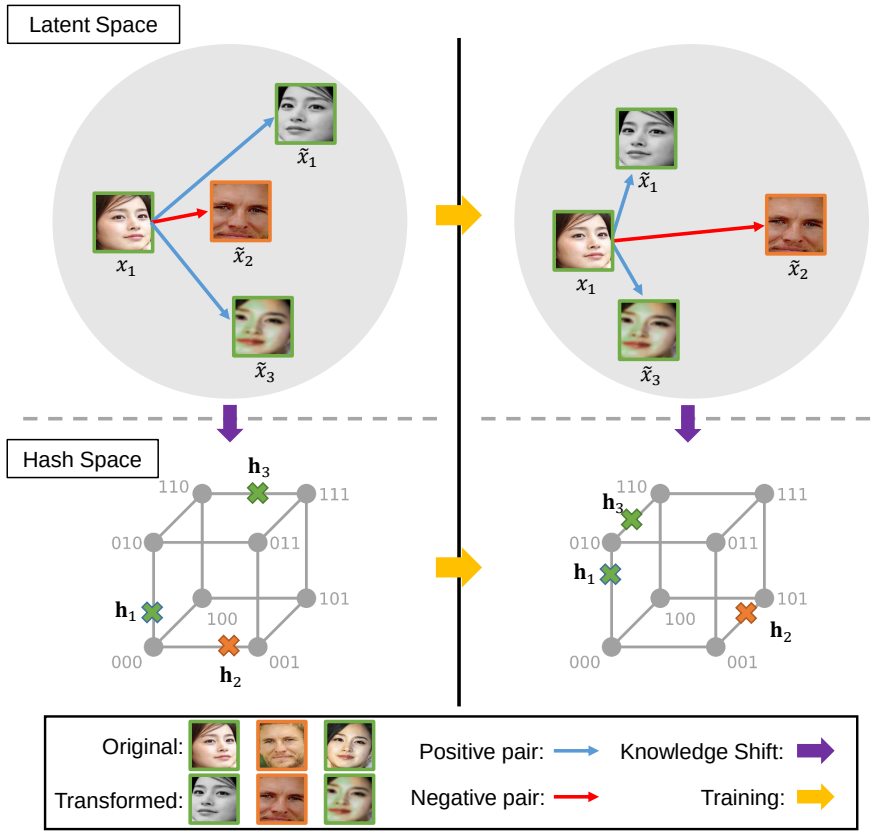


Figure 2.2: A simple visualization of proposed method. $\tilde{x}_1, \tilde{x}_2, \tilde{x}_3$ are the transformed output of x_1, x_2, x_3, respectively, and I only illustrate the case of x_1 for simplicity. x_1 forms a positive pair with $\tilde{x}_1$, and $\tilde{x}_3$ derived from the same identity, and forms a negative pair with $\tilde{x}_2$ derived from the different identity. With this similarity information learned in the latent space, the final output hash codes $\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3$ which are originated from x_1, x_2, x_3, respectively, can preserve pairwise similarity near the Hamming space (cubic) to achieve high retrieval performance.

tion understanding. my method is inspired by the concept of contrastive learning [4, 12], which aims to find similarity between images by contrasting them. Specifically, transformed images originating from the same image are considered similar (self-similarity), and the other transformed images from different images are considered dissimilar (pairwise-similarity). In this case, I exploit the allocated label information when determining the pairwise-similarity, and the transformed images that originate from different images of the same identity are regarded as positive pairs.

For better understanding, I illustrate my conceptual scheme in Figure 2.2. The combination of random data augmentations such as crop and resize, flip, color distortion, and Gaussian blurring is applied to face images with the consideration of facial properties. The self-similarity and the pairwise similarities are considered simultaneously in the latent space during training. Then, the learned similarity knowledge is shifted into the hash space to obtain the discriminative hash codes.

In this dissertation, I propose a new self-similarity assisted convolutional neural network-based Hashing framework; named *Similarity Guided Hashing* (SGH), which generates binary-like hash codes for fast and accurate retrieval. Specifically, I introduce two novel training objectives. First, a novel Self-similarity Pairing loss is introduced on transformed face images with human supervision (identity label annotations). This approach allows a more sophisticated understanding of the correlations between images. Second, to alleviate the gap between discrete binary code and the hash code, I suggest a squared Quantization loss to minimize it fast and moderately. Additionally, a $l2$-regularization is utilized to avoid severe deviations between the original and the transformed image feature representations, and a standard Cross-Entropy function for classification is employed to enhance the general discriminability between hash codes. The entire loss functions are applied to SGH in an end-to-end manner, without any complex training batch configuration strategy.

To demonstrate that my method excels previous methods in various conditions, I construct a new experimental protocol using a face detection algorithm (DSFD) [16] on VGGFace2 [1] test set, a large scale image dataset for face recognition. The dataset I configure holds three times higher resolution images than the existing protocols, thus containing richer facial feature representations. Furthermore, I conduct identity-disjoint experiments proposed in [31] with the subset of VGGFace2 training set. The results show that my SGH encodes face images into the Hamming space properly, despite the absence of the identity information for the network training.

2.1.1 Related Works

This section presents a brief introduction to the deep hashing methods for general images and face images. Refer to the survey [27] to see the early works in non-deep learning binary hashing (ITQ [5], SH [28], KSH [20], SDH [22]) which can be utilize as hashing system for face image retrieval.

General image hashing methods Convolutional Neural Network (CNN)-based hashing approaches with fully supervised learning [30, 19, 87] are leading the mainstream with promising outcomes. For example, CNN Hashing (CNNH) [30] utilizes a CNN to generate compact hash codes by training the network with the given pairwise label information. Deep Supervised Hashing (DSH) [19] learns hash codes by approximating discrete values with relaxation and training them with the supervised signals. Most recently, the method that exploits the global similarity metric [87] shows outstanding performance in the supervised scheme. Meanwhile, for hashing with few labels or without labels, deep semi-supervised learning [76] and deep unsupervised learning [82] also attracted attention with high scores.

Face image hashing methods In terms of face images, there have been several CNN-based hashing approaches [25, 18, 26, 44, 31] that take into account face characteristics. In [25], they proposed an end-to-end framework that simultaneously learns face features and the binary hash codes by minimizing classification and quantization loss at once, named Deep Hashing based on Classification and Quantization errors (DHCQ). Discriminative Deep Hashing (DDH) method [18] integrated divide-and-encode modules to reduce the redundancies among hash codes and the network parameters to improve DHCQ. Discriminative Deep Quantization Hashing (DDQH) [26] upgraded DDH by expanding the number of channels in layers, replacing the divide-and-encode module with a fully-connected layer, and applied a batch normalization quantization module. Discrete Attention Guided Hashing (DAGH) [31] employed dis-

crete identity loss and a multi-attention cascade network to capture face features.

SGH is the first work to split the latent space and the hash space, and introduce a self and pairwise-similarity learning. By assisting deep hashing model training with complicated similarity knowledge, SGH is able to surpass existing algorithms with state-of-the-art experimental outcomes. Furthermore, especially to precisely assess the advantage of my proposal in face image retrieval protocols, I configure a new dataset with larger images than the previous ones, and SGH also shows the best results.

2.2 Similarity Guided Hashing

The goal of deep Hashing for face image retrieval is to map an input image x to an identity-wise discriminative K-bits binary code $\mathbf{b} \in \{-1, 1\}^K$. However, the network training process does not include binarization, since the sign function used for binary encoding is non-differentiable. Instead, my deep model is trained in the real-valued space with the self-similarity and pairwise-similarity to learn continuous facial representations. At the same time, I minimize the quantization error that occurs during the binary conversion to reduce the gap between discrete binary code and real-valued hash code. Also, the knowledge learned in the latent space is shifted into the hash space while minimizing the classification error of hash codes to increase retrieval accuracy.

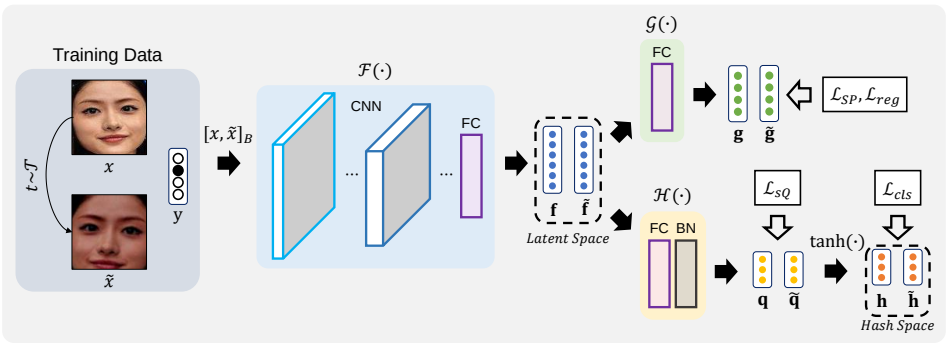


Figure 2.3: The overall training process of SGH.

As illustrated in Figure 2.3, my Self-supervision Guided Hashing (SGH) consists of three components with trainable parameters. From a face image input x, feature extractor $\mathcal{F}(\cdot)$ outputs the face feature vector $\mathbf{f}$. Any CNN architecture commonly used for image retrieval such as AlexNet [14], VGG-F [38] or ResNet [6] can be employed as a feature extractor. I select ResNet as my baseline with a simple modification in the number of convolutional filter channels.

Following the observations in [4, 12], I introduce projection head $\mathcal{G}(\cdot)$ to learn the self and pairwise-similarity with the dimension-reduced vector $\mathbf{g}$, rather than directly utilizing $\mathbf{f}$ for training. Note that, I only apply a single fully-connected layer (FC) without non-linearity. This has the advantage of being able to transfer the knowledge learned from $\mathbf{g}$ to $\mathbf{f}$ without redundancy.

Hashing head $\mathcal{H}(\cdot)$ is similar to the BNQ module proposed in [26]. The FC layer embeds the latent space into the hash space of the desired bit length, and the batch normalization layer [7] (BN) speeds up the convergence and improves the stability of learning. The output of the Hashing head, $\mathbf{q}$, is mapped to $\mathbf{h}$ consisting of real values between -1 and 1 through the $\tanh(\cdot)$ function. When I conduct retrieval, the input image x is converted to the hash code $\mathbf{h}$ by passing through the SGH framework, and finally becomes a discrete binary code with the sign function.

Following the works presented in [4, 12], I apply various data augmentation techniques on the face image for training. As illustrated in Figure 2.4, I choose five methods to transform images. Most of the hyperparameters for augmentation are directly taken from [4] except for color jitter strength as 1/5, to take into account the skin color characteristics between different races. As listed in Figure 2.4, all transformations are applied with random probabilities in a sequential manner, starting with the resized crop and ending with the Gaussian blur, to build augmentation family $\mathcal{T}$.

The advantages that can be considered from each augmentation technique are: 1) local, global and adjacent views with the resized crop, 2) mirrored inputs with the horizontal flip, 3) color distortions with color jitter, 4) color independent facial repre-

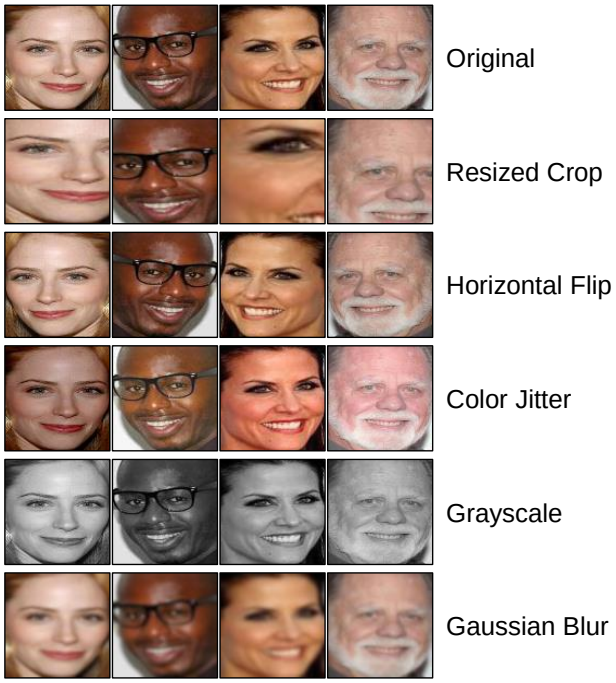


Figure 2.4: Examples of face images with various augmentation techniques applied.

sentations with grayscale, and 5) noise in image with Gaussian blur. As a result, the contents included in the face image are robustly compared between the original and the transformed one in a self-supervised way to find the discriminative facial image representations.

Suppose, we are given a dataset with N training samples; $\mathcal{X} = \{(x_1, \mathbf{y}_1), ..., (x_N, \mathbf{y}_N)\}$ where each image x_i is assigned a one-hot encoded label $\mathbf{y}_i \in \{0, 1\}^c$ of c identities. Transformed images are obtained as $\tilde{x}_i = t(x_i)$ for all samples $\{\tilde{x}_i\}_{i=1}^N$, where t is from $t \sim \mathcal{T}$. I propose three training objectives that learn complex facial representations in a high-dimensional latent space and make the hash code discriminative near the Hamming space. To cope with difficult-to-distinguish features of the face images such as similar appearance among different people, and variations in one person (facial expression, makeup, pose, illumination), I introduce Self-similarity Pairing loss that trains $\mathcal{F}(\cdot)$ and $\mathcal{G}(\cdot)$ as:

$$\mathcal{L}_{SP}(\mathbf{g}, \mathbf{y}) = \frac{1}{N_B} \sum_{i=1}^{N_B} (\mathcal{L}_{CE}(S_i, Y_i)) \tag{2.1}$$

where N_B denotes the number of images in a training batch. Similar to the concept proposed in [24] to conduct metric learning, I employ a standard cross-entropy loss $\mathcal{L}_{CE}$, where S_i denotes cosine similarity between the i-th $\mathbf{g}$ and every $\tilde{\mathbf{g}}$ in a batch; $S_i = \left[\mathbf{g}_i^T \tilde{\mathbf{g}}_1, ..., \mathbf{g}_i^T \tilde{\mathbf{g}}_{N_B}\right]$, and Y_i denotes a similarity between the i-th label and every label in a batch; $Y_i = \left[\mathbf{y}_i^T \mathbf{y}_1, ..., \mathbf{y}_i^T \mathbf{y}_{N_B}\right]$, and I apply normalization as $Y_i / \|Y_i\|_1$ to balance the contribution. The self-similarity part $\mathbf{g}_i^T \tilde{\mathbf{g}}_i$ can solve intra-identity difficulties, and the remaining parts contribute to inter-identity discriminability.

Additional $l2$-regularization of embedding vectors is employed as:

$$\mathcal{L}_{reg}(\mathbf{g}) = \frac{1}{N_B \cdot D_{\mathbf{g}}} \sum_{i=1}^{N_B} \sum_{j=1}^{D_{\mathbf{g}}} \left(g_{ij}^2\right) \tag{2.2}$$

where $D_{\mathbf{g}}$ is the dimensionality of $\mathbf{g}$, and g_{ij} are the j-th element of $\mathbf{g}_i$. This regularization term forces the norm of $\mathbf{g}$ and $\tilde{\mathbf{g}}$ to be small, avoiding severe deviation.

The last element of my architecture is the "Block Hashing," which converts a large real-valued feature vector into a binary hash code. The structure of the block Hashing is detailed in Figure 2.5, which receives the descriptor as the input and produces the hash code as the output. The final hash code is set to have K bits where K can be set bigger for the larger database. For the pre-defined hash size K, I set the size of the fully connected layer to $K \times M$ where M is called the block size. As shown in the figure, I divide the $K \times M$ descriptor ($\hat{d} = [d_1, \ldots, d_{K \times M}]$) into M blocks, and I extract the maximum of the Softmax of the elements in the i-th block as

$$z_i = \max(\text{Softmax(elements of descriptor in the } i\text{-th block)}). \tag{2.3}$$

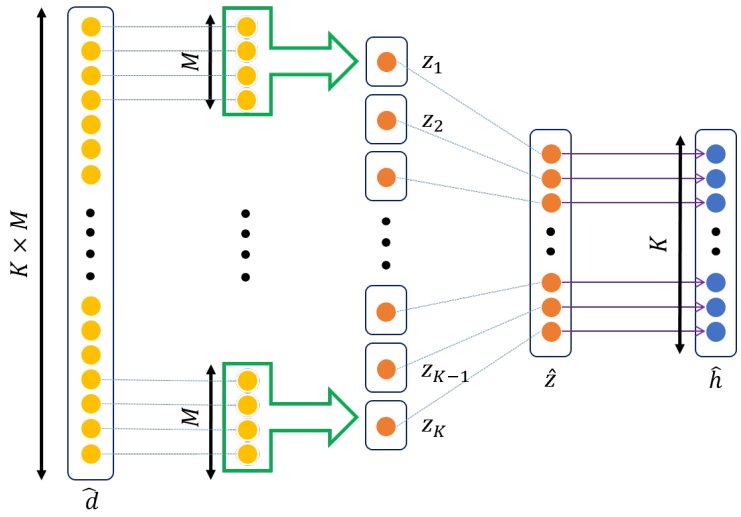


Figure 2.5: Block Hashing layer.

Then, these are concatenated to be a vector $\hat{z} = [z_1, z_2, \cdots, z_K]$, and the final hash code $\hat{h}$ is obtained as

$$\hat{h} = [h_i \mid h_i = \tanh(z_i - \beta),\ i = 1, \ldots, K] \tag{2.4}$$

where β is a balancing parameter.

In this approach, each block is generated from a face image descriptor of a separate slice. Hence, blocks can effectively contain some characteristics of facial representations which are less relevant to each other. Unlike divide-and-encode module in [**?**], my block Hashing layer does not have any trainable parameters, which can greatly reduce the redundancy. In addition, since I extract the maximum of the block-wise output during the back propagation, the gradient from the next layer is passed back to only that neuron which achieved the max. As a result, discriminative information of the large real-valued descriptor is well maintained while each element of the block is effectively trained to be a representative value of the block according to the classification result of the descriptor.

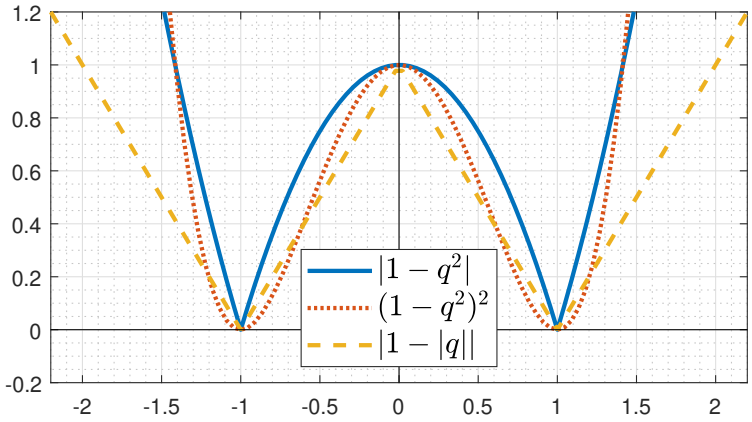


Figure 2.6: An illustration of different quantization loss functions.

Moving on to hash code learning, I present two loss functions to train $\mathcal{F}(\cdot)$ and $\mathcal{H}(\cdot)$. First, to cut down the gap between the discrete binary code and the real-valued hash code, I compute and minimize the squared Quantization loss as:

$$\mathcal{L}_{sQ}(\mathbf{q}) = \frac{1}{N_B \cdot K} \sum_{i=1}^{N_B} \sum_{j=1}^{K} \left(|1 - q_{ij}^2|\right) \tag{2.5}$$

where q_{ij} is the j-th element of $\mathbf{q}_i$, and $|\cdot|$ is the operation to return the absolute value. This term aims to minimize the quantization error and contributes to approximate the discrete binary code $\mathbf{b}$ with $\mathbf{q}$. Following [26], I compute the loss before non-linear activation to increase the convergence speed and accuracy.

To compare with previously proposed quantization loss functions [18, 44], I plot each one, including mine in Figure 2.6. The double squared one ($(1-q^2)^2$, orange dotted line) has a convex shape near -1 and 1, which makes it difficult to be strictly approximated, and the double absolute one ($|1-|q||$, yellow dashed line) generates relatively small error values, making a weak contribution. Therefore, I utilize the loss function (blue line) that can complement each other's shortcomings by combining the two.

Second, to make the binary code discriminative with the identity labels, I estimate the class probability on $\mathbf{h}$, which is the relaxed version of $\mathbf{b}$ as:

$$\mathcal{L}_{cls}(\mathbf{h}, \mathbf{y}) = \frac{1}{N_B} \sum_{i=1}^{N_B} \mathcal{L}_{CE}(\text{FC}_{cls}(\mathbf{h}_i), \mathbf{y}_i) \tag{2.6}$$

where FC_{cls} is a fully connected layer that outputs the probability prediction for all identities. In summary, high retrieval accuracy is achieved by taking into account the self and pairwise-similarity in latent space, as well as the general classification performance of the hash codes generated from the original and transformed images.

2.3 Experiments

2.3.1 Datasets and Setup

Table 2.1: Dataset configuration used to perform face image retrieval experiments. Ids and Resol. are short for identities and resolution, respectively.

Dataset	# Retrieval	# Test	#Ids	Resol.
closed-set Protocol				
YouTube Faces	63,800	7,975	1,595	32×32
FaceScrub	67,177	2,650	530	32×32
VGGFace2-Test	75,296	2,500	500	96×96
Open-set Protocol				
VGGFace2-Train	17,940	500	100	96×96

Datasets . Following the closed-set protocol proposed in [25, 18, 26, 44] where the retrieval database includes training samples, I conduct experiments on two well-organized face image retrieval datasets and a high-resolution face image dataset that I constructed. The **YouTube Faces** dataset (Y.T.F) [29] is composed of video frames containing faces, which is designed to study unconstrained face recognition problems. There are 1,595 different identities to classify, where 40 images for each person are

randomly selected to be the training data, and five images per person are utilized for the test data. A total of 63,800 images are used to build a training set and a retrieval database, and 7,975 images are used as a query dataset for the experiment. The **FaceScrub** dataset (F.S) [21] consists of 106,863 face images which are collected from the Internet. There are 530 celebrity classes to distinguish with about 200 images per person. I choose five images for each person to utilize a total of 2,650 images in the experiment, and the remaining images are used for training and constructing a retrieval database. For both datasets above, the size of the image is fixed at 32×32.

In addition, I configure a new higher resolution face image retrieval dataset with **VGGFace2** (V.F2) [1], which contains over 3.3 million images of more than 9,000 identities in the training set, and about 0.15 million images of 500 identities in the test set. I apply a fast and accurate deep learning-based face detection algorithm DSFD [16] to the images in the *test set* of V.F2 to extract strict face images. I select the images whose width and height do not exceed 1,024 for detection and utilize the output face images that have confidence scores over 0.99 to build a dataset. Also, all extracted images have been resized to 96×96, and images that do not exceed 96 in either width and height are discarded during this process. As a result, I could collect a total of 75,296 face images of 500 different identities. Similar to existing dataset protocols, I split the dataset into two as 72,796 images for the training set, and the rest 2,500 images for the test (5 images per identity).

Following the identity-disjoint (open-set) protocol proposed in [31], I randomly sample 100 identities from the *training set* of V.F2 with a total of 17,940 images, which do not participate in the training process. The same face detection and resizing method are applied to conduct retrieval with the model trained from V.F2 closed-set protocol. Five images per identity are randomly selected for testing, and the remaining images are utilized to build a retrieval database. The exact numbers are reported in Table 2.1.

Table 2.2: mAP scores of different Hashing approaches on small face image datasets.

Method	YouTube Faces				FaceScrub			
	12-bits	24-bits	36-bits	48-bits	12-bits	24-bits	36-bits	48-bits
ITQ [5] + CNN	0.0248	0.1900	0.3420	0.4394	0.0186	0.0352	0.0504	0.0667
SH [28] + CNN	0.0154	0.0851	0.1603	0.2421	0.0036	0.0081	0.0114	0.0145
KSH [20] + CNN	0.0481	0.2663	0.4167	0.5047	0.0230	0.0348	0.0767	0.1026
SDH [22] + CNN	0.5474	0.7676	0.8100	0.8331	0.1281	0.2388	0.2934	0.3291
DSH [19]	0.5034	0.6011	0.7132	0.7354	0.5341	0.5955	0.6112	0.6234
DHCQ [25]	0.8108	0.8892	0.9122	0.9258	0.4986	0.5834	0.6215	0.6387
DDH [18]	0.8808	0.9212	0.9340	0.9412	0.5985	0.6121	0.6445	0.6765
DDH-Deep	0.9322	0.9455	0.9580	0.9718	0.6215	0.6479	0.6883	0.6983
DDQH [26]	0.9002	0.9553	0.9684	0.9820	0.6452	0.6824	0.7120	0.7355
DDQH-Deep	0.9580	0.9782	0.9790	0.9834	0.6618	0.7122	0.7654	0.7798
DCBH [44]	0.9753	0.9899	0.9914	0.9922	0.7182	0.7317	0.7696	0.7862
DAGH [31]	0.9744	0.9926	0.9938	0.9946	0.7284	0.7919	0.8172	0.8204
SGH	**0.9902**	**0.9933**	**0.9955**	**0.9966**	**0.8970**	**0.9219**	**0.9319**	**0.9345**

Setup . To evaluate face retrieval quality, I employ four metrics: 1) mean average precision (**mAP**), 2) precision within Hamming distance 2 (**P@H$\leq$2**) for different bit-lengths, 3) precision-recall curves (**PR curves**), and 4) precision with respect to top-M returned image (**P@Top-**M). In terms of calculating mAP scores, I select the top 50 images from the retrieval ranked-list results. The retrieval accuracy is estimated based on whether the returned images and the query image have the same identity label or not. I set the length of binary codes as 12, 24, 36, and 48 to examine the performance according to the number of bits.

2.3.2 Results on Small Face Images

As shown in Table 2.1, I calculate the mAP scores on small face-image datasets; Y.T.F and F.S for both non-deep learning-based and deep learning-based Hashing approach over four types of bit-lengths. Deep Hashing methods generally outperform non-deep Hashing ones because elaborately labeled identity information is fully utilized during deep model training and makes deep hash codes discriminative. In the case of SDH,

Table 2.3: mAP scores of different Hashing approaches on large face image dataset.

Method	VGGFace2 12-bits	24-bits	36-bits	48-bits
closed-set Protocol				
DDH-Deep	0.3385	0.3815	0.4120	0.4424
DDQH-Deep	0.3507	0.3934	0.4324	0.4658
DCBH	0.6612	0.6882	0.7084	0.7254
DAGH	0.6576	0.6998	0.7273	0.7556
SGH	**0.8521**	**0.8886**	**0.9046**	**0.9174**
Open-set Protocol				
DDH-Deep	0.2154	0.2689	0.3413	0.3995
DDQH-Deep	0.2193	0.2872	0.3736	0.4285
DCBH	0.2456	0.3013	0.3885	0.4313
DAGH	0.2334	0.3126	0.3906	0.4425
SGH	**0.3342**	**0.4380**	**0.4920**	**0.5594**

which employs supervised signals, it also shows promising outcomes even though deep learning is not exploited.

To verify that my self-similarity training scheme and squared Quantization loss are effective, I conduct experiments with DDH-Deep and DDQH-Deep, which share the same network architecture with SGH, noting that network architecture is the only factor that contributes to improvement. In addition, since DCBH and DAGH have claimed their contributions to the network architecture, I do not add any modification to them. Following the experimental results shown in Table 2.2 and Figure 2.3, 2.4, my SGH can achieve the best performance over all bit-lengths, and accurately retrieves relevant images more in the top ranks.

2.3.3 Results on Large Face Images

The images included in Y.T.F and F.S are too small to represent detail facial characteristics. Therefore, I construct a new dataset with VGGFace2 where the resolution of all images is three times higher to compare the retrieval performance of SGH with the existing deep face image retrieval algorithms. As observed in Table 2.3, SGH surpasses

all compared methods by large margins in both the closed-set and open-set protocols, even if the network architecture is the same (DDH-Deep, DDQH-Deep). In a nutshell, the proposed loss functions and training strategy of SGH can make the network generate descriptive face hash codes from large face images to conduct accurate retrieval.

2.4 Motivation and Overview for General Image Retrieval

Learning to hash has been one of the most important tasks in the image retrieval community. Especially for retrieval from large-scale databases, hashing is essential due to its high search speed and low storage cost. By converting high-dimensional data points into compact binary codes with a hash function, the retrieval system can utilize a simple bit-wise XOR operation to define a distance between the images. A wide variety of works have been studied for this purpose [5, 28, 20, 22], and are still being actively pursued.

In recent years, techniques for hash learning have been significantly advanced by deep learning, which is called *deep hashing*. By integrating the hash function into the deep framework, deep encoder and hash function are simultaneously learned to generate image hash codes. Regarding the training of deep hashing, the leading techniques are pairwise similarity approaches that use sets of similar or dissimilar image pairs [30, 34, 2, 76], and global similarity in company with classification approaches that use class labels assigned to images [44, 10, 87].

Since hash-based retrieval systems compute the distance between images with binary codes, corresponding codes need to be quantized with *sign* operation, from the continuous real space to the discrete Hamming space of $\{-1, 1\}$. In this process, the continuously optimized image representation is altered, and quantization error occurs, which in turn degrades the discriminative capability of the hash code. This becomes even more problematic when an input image is transformed and deviated from the original distribution.

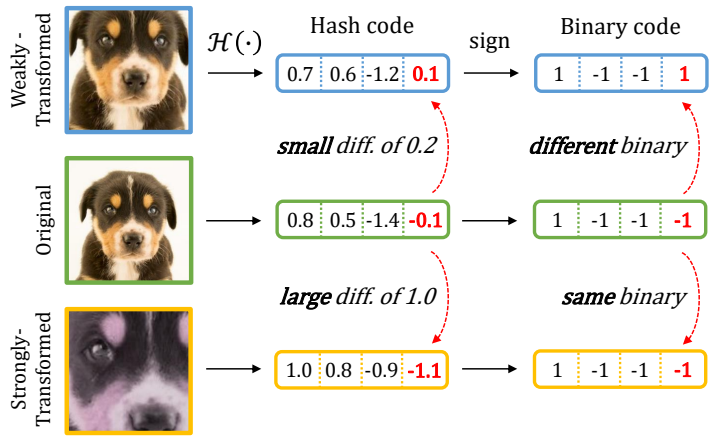


Figure 2.7: Visualization of possible problems with deep hashing due to transformations. The continuous hash code generated by a deep hashing model $\mathcal{H}(\cdot)$ is changed when the input is transformed. In consequence, the binary code quantized with sign operation can also be shifted. However, the degree of transformation cannot be properly reflected in the quantized representation.

To avoid performance degradation due to transformations, the most common solution is to generalize the deep model by training it with augmented data having various transformations. However, it is challenging to apply this augmentation strategy to deep hashing training since discrepancy in the representation may occur. Figure 2.7 shows an example case that may appear: 1) *The sign of the hash code can be shifted with a slight change*. Specifically, the last element of the weakly-transformed image's hash code differs by 0.2 ($-0.1 \rightarrow 0.1$) from the original, but it results in $-1 \rightarrow 1$ shift in the Hamming space. 2) *The sign of the quantized hash code does not shift even with the big change in the hash code*. The last element of a strongly transformed image's hash code differs by 1.0 ($-0.1 \rightarrow -1.1$) from the original, resulting in no shifts in the Hamming space. Namely, the direct use of data augmentation in deep hashing increases the discrepancy between Hamming and real space, which hinders finding the optimal binary code.

To resolve this issue, I introduce a novel concept dubbed *Self-distilled Hashing*, which customizes self-distillation [92, 93] to prevent severe discrepancy in deep hash-

ing training. Specifically, based on the understanding of the relation between cosine distance and Hamming agreement, I minimize the cosine distance between the hash codes of two different views (transformed results) of one image to maximize the Hamming agreement between their binary outcomes. Further for stable learning, I separate the difficulties of transformations as easy and difficult, and transfer the hash knowledge from easy to difficult.

Moreover, I propose two additional training objectives that optimize hash codes to enhance the self-distilled hashing: 1) a hash proxy-based similarity learning, and 2) a binary cross entropy-based quantization loss. The first term allows the deep hashing model to learn global (inter-class) discriminative hash codes with temperature-scaled cosine similarity. The second term contributes to making the hash code naturally move away from the binary threshold in a classification manner with likelihood estimators.

By combining all of my proposals, I construct a **D**eep-**H**ash **D**istillation framework (DHD), which yields discriminative hash codes for fast image retrieval. I conduct extensive experiments on single and multi-labeled benchmark datasets for image hashing evaluation. In addition, I validate the effectiveness of self-distilled hashing using data augmentation on the existing methods [2, 94, 95, 87] and show the performance improvements. Furthermore, I establish that DHD is applicable with a variety of deep backbone architectures including vision transformers [96, 93, 97]. Experimental results confirm that DHD significantly improves the retrieval performance with the state-of-the-art scores.

2.5 Related Works

For a better understanding, I present a brief introduction to the deep hashing methods and the research that inspired my proposal. Refer [27] to see more details of the early works in non-deep hashing approaches (ITQ [5], SH [28], KSH [20], SDH [22]).

Deep hashing methods. Hashing algorithms using deep learning techniques such as Convolutional Neural Network (CNN) are leading the mainstream with striking results. For example, CNNH [30] utilizes a CNN to generate compact hash codes by training a network with given pairwise label information. DHN [34] learns hash codes by approximating discrete values with relaxation and trains them with supervised signals. HashNet [2] adopts the inner product to measure pairwise similarity between hash codes and tackles the data imbalance problem by employing weighted maximum likelihood estimation. DCH [94] employs Cauchy distribution to minimize the Hamming distance of the images with the same class label.

Hash center-based methods. There have been several approaches to find out class-wise hash representatives (centers), which can provide global similarity to hash codes by including the process of predicting image class labels with hash codes during training [44, 10, 76, 87]. CSQ [87] uses pre-defined orthogonal binary hash targets to guarantee distant Hamming distance between classes and makes hash codes to follow the targets. DPN [95] employs randomly assigned target vectors with maximal inter-class similarity and utilizes bit-wise hinge-like loss. Unlike DPN and CSQ, which use a hash target that is not trainable, in DHD, the hash center is set as a trainable proxy which jointly learns the similarity with the hash codes during training.

Self-distillation. Inspired by knowledge distillation, self-distillation emerges as a concept that employs a single network to generalize itself in a self-taught fashion, and plenty of works demonstrate its benefits in improving deep model performance [92, 93]. Many of them utilize a simple Siamese architecture to explore and learn the visual representation with data augmentation, by contrasting two different augmented results of one image. Similarly, I conduct the self-distillation with augmentations in deep hashing to see the hash codes of two different views of one image simultaneously. Additionally, in accordance with the characteristics of hashing, I consider a method of minimizing the cosine distance that behaves similarly to the distance in the Hamming

space to reduce the representation discrepancy during model training.

2.6 Deep Hash Distillation

The goal of a deep hashing model $\mathcal{H}$ of Deep Hash Distillation (DHD) is to map an input image x to a K-dimensional binary code $\mathrm{b} \in \{-1, 1\}^K$ in Hamming space. For this purpose, $\mathcal{H}$ is optimized to find a high quality real-valued hash code h, and then sign operation is utilized to quantize h as b. Instead of including non-differentiable quantization process in model training, I learn $\mathcal{H}$ in the real space to estimate optimal b with continuously relaxed h while fully exploiting the power of data augmentation. I notate trainable component with θ as a subscript.

2.6.1 Self-distilled Hashing

In general, $\mathcal{H}$ is trained in the real space to obtain discriminative h, which should maintain its property in the Hamming space even if quantized to b. Therefore, it is important to align h and b to carry a similar representation during training. However, when data augmentation is applied to the training input and the following change occurs in h, there can be misalignment between h and b. Thus, direct use of augmentations can cause discrepancies in representation between h and b, which degrades performance.

Hamming distance as cosine distance. It is noteworthy that the Hamming distance between the binary codes can be interpreted as cosine distance (1-cosine similarity). That is, for deep hashing, the cosine similarity between hash codes h_i and h_j can be utilized to approximate the Hamming distance between the binary codes b_i and b_j as:

$$\mathcal{D}_{\mathrm{H}}(\mathrm{b}_i, \mathrm{b}_j) \simeq \frac{K}{2}(1 - \mathcal{S}\left(\mathrm{h}_i, \mathrm{h}_j\right)) \tag{2.7}$$

where $\mathrm{b}_i = \mathrm{sign}(\mathrm{h}_i)$, $\mathrm{b}_j = \mathrm{sign}(\mathrm{h}_j)$, $\mathcal{D}_{\mathrm{H}}(\cdot, \cdot)$ denotes Hamming distance, $\mathcal{S}(\cdot, \cdot)$ de-

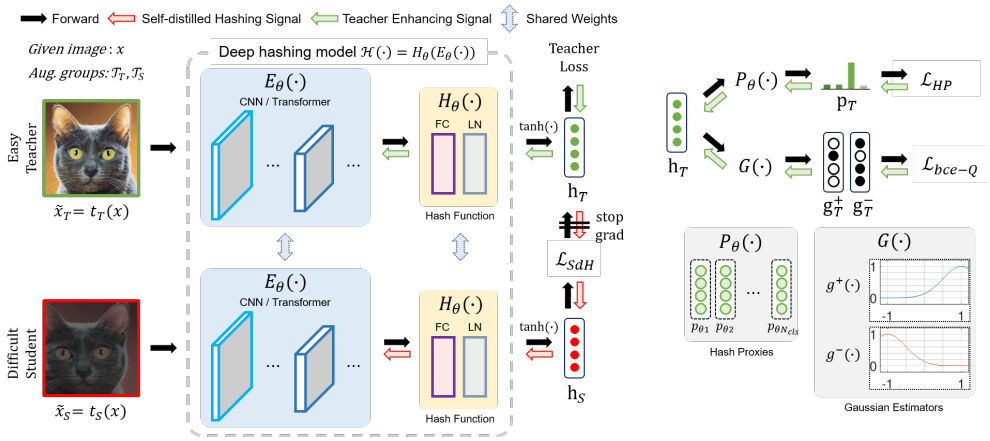


Figure 2.8: The overall training process of Deep Hash Distillation (DHD) framework. (Left) From two different augmentation groups: Teacher $\mathcal{T}_T$ and Student $\mathcal{T}_S$, randomly sampled transformations ($t_T \sim \mathcal{T}_T$, $t_S \sim \mathcal{T}_S$) are individually applied on the input image x to produce $\tilde{x}_T$ and $\tilde{x}_S$. The deep hashing model $\mathcal{H}(\cdot)$ constructed with the deep encoder $E_\theta(\cdot)$ and the hash function $H_\theta(\cdot)$ of one Fully-Connected (FC) layer and Layer Normalization (LN) yields two hash codes h_T and h_S which are learned with $\mathcal{L}_{SdH}$. I apply stop gradient operation on h_T for stable training. (Right) Additionally, I employ trainable hash proxies $P_\theta(\cdot)$ which are used to calculate the class-wise prediction p_T with h_T to optimize with $\mathcal{L}_{HP}$, and pre-defined Gaussian estimator $G(\cdot)$ to regularize h_T with $\mathcal{L}_{bce\text{-}Q}$.

notes cosine similarity. That is, the minimized cosine distance between the hash codes minimizes the Hamming distance between the binary codes.

Easy-teacher and difficult-student. As shown in Figure 2.8, I propose a self-distilled hashing scheme, which supports the training of deep hashing models with augmentations. I employ weight-sharing Siamese structure to contrast hash codes of two different views (augmentation results) of one image at once. According to the observations in self-distillation works [93], keeping the output representation of one side steady has a significant impact on performance gain. Therefore, I configure two separate augmentation groups to provide input views with different difficulties of transformation: one is weakly-transformed easy teacher $\mathcal{T}_T$, and the other is strongly-transformed difficult student $\mathcal{T}_S$. Here, I control the difficulty in a stochastic sampling manner as: employ-

ing the same hyper-parameter s_T to all transformations in the group, and make them occur less (weakly) or more (strongly) by scaling their own probability of occurrence. While this manner makes the teacher representation stable, it has the advantage that few extreme examples that produce unstable results are not completely ruled out and contribute to learning. Besides,I stop the gradient of the teacher view's corresponding hash codes to avoid collapsing into trivial solutions.

Loss computation. For a given image x, self-distillation is conducted with image views as: $\tilde{x}_T = t_T(x)$ and $\tilde{x}_S = t_S(x)$, where t_T, t_S are randomly sampled transformations from $\mathcal{T}_T, \mathcal{T}_S$, respectively. The deep encoder E_θ and the hash function H_θ take $\tilde{x}_T$ and $\tilde{x}_S$ as inputs and produce corresponding hash code h_T and h_S. Then, the proposed Self-distilled Hashing (SdH) loss is computed as:

$$\mathcal{L}_{SdH}(\mathrm{h}_T, \mathrm{h}_S) = 1 - \mathcal{S}(\mathrm{h}_T, \mathrm{h}_S) \tag{2.8}$$

Optimizing $\mathcal{H}$ with $\mathcal{L}_{SdH}$ results in the alignment of h_T and h_S, and thus b_T and b_S as follows Eqn. 2.7, which in turn reduces the discrepancy in representation between two differently transformed output binary codes.

Flexibility. Note that self-distilled hashing is applicable to the other common deep hashing models [2, 94, 95, 87] with regard to exploiting data augmentation during training. Furthermore, various backbones [14, 6, 96, 93, 97] can be utilized as deep encoder, and any hash function H_θ configuration is compatible. For simplicity, I employ a single FC layer with a layer normalization to address overly dominant binary bits and balance the intra-binary representation, and apply *tanh* operation at the end to be bound in $[-1, 1]$.

2.6.2 Teacher loss

Besides self-distilled hashing, additional training signals such as supervised learning loss, and quantization loss are required to obtain the discriminative hash codes. I only employ teacher hash codes to compute the losses, in order to transfer the learned hash knowledge to the student's codes.

Proxy-based similarity learning. Supervised hash similarity learning with pre-defined non-trainable binary hash targets has shown great performance [87, 95]. However, the hash target has limitations in that it requires a complex initialization process, and cannot contain semantic similarity by itself. Therefore, as shown in Figure 2.8, I introduce a collection of trainable hash proxies P_θ that can simply be initialized along with deep parameters of $\mathcal{H}$, and use this to compute class-wise prediction p_T with h_T as:

$$\mathrm{p}_T = [\mathcal{S}(p_{\theta 1}, \mathrm{h}_T), \mathcal{S}(p_{\theta 2}, \mathrm{h}_T), ..., \mathcal{S}(p_{\theta N_{cls}}, \mathrm{h}_T)] \tag{2.9}$$

where p_θ is a hash proxy assigned to each of the i category and N_{cls} denotes the number of categories. Then, I use p_T to learn the similarity with class label y by computing Hash Proxy (HP) loss as:

$$\mathcal{L}_{HP}(\mathrm{y}, \mathrm{p}_T, \tau) = H\left(\mathrm{y}, \mathrm{Softmax}(\mathrm{p}_T/\tau)\right) \tag{2.10}$$

where τ is a temperature scaling hyper-parameter, $H(\mathrm{u}, \mathrm{v}) = -\sum_k u_k \log v_k$ is a cross entropy, and Softmax operation is applied along the dimension of p_T. Although other hashing losses [2, 94, 95, 87] are available, I employ $\mathcal{L}_{HP}$ since it shares the property of cosine similarity with the self-distilled hashing. Note that, similar to Eqn 2.7, $\mathcal{L}_{HP}$ can learn Hamming agreement if scaled with τ.

Reducing quantization error. To make continuous hash code elements act like binary bits, the deep hashing methods [94, 2, 44, 87] aim to reduce the quantization error by minimizing the distance (e.g. Euclidean) between the hash code bit and its closest binary goal ($+1$ or -1) in a regression manner. However, since the purpose of hashing is to classify the sign of each bit, it is a more natural choice to view it as a binary classification: maximum likelihood problem. Hence, I adopt a pre-defined Gaussian distribution estimator $g(h)$ of mean m and standard deviation σ as:

$$g(h) = \exp\left(-\frac{(h-m)^2}{2\sigma^2}\right) \tag{2.11}$$

to evaluate the binary likelihood of hash code element h. By employing two estimators: g^+ of $m = 1$, and g^- of $m = -1$ with the same σ, I compute the likelihoods and a Binary Cross Entropy-based (BCE) quantization loss as:

$$\mathcal{L}_{bce\text{-}Q}(\mathrm{h}_T) = \frac{1}{K}\sum_{k=1}^{K}\left(H_b\left(b_k^+, g_k^+\right) + H_b\left(b_k^-, g_k^-\right)\right) \tag{2.12}$$

where $H_b(u, v) = -u\log v + (1-u)\log(1-v)$ is a binary cross entropy, g_k^+, g_k^- denotes k-th hash code element's estimated likelihood: $g_k^+ = g^+(h_k)$, $g_k^- = g^-(h_k)$, and b_k^+, b_k^- denotes binary likelihood labels which are obtained as:

$$b_k^+ = \frac{1}{2}\left(\mathrm{sign}(h_k) + 1\right), b_k^- = 1 - b_k^+ \tag{2.13}$$

As a result, quantization error is reduced by a binary classification loss with the given estimators, allowing to use the merits of cross entropy.

2.6.3 Training

Total training loss. Suppose we are given a training mini-batch of N_B data points: $\mathcal{X}_B = \{(x_1, \mathrm{y}_1), ..., (x_{N_B}, \mathrm{y}_{N_B})\}$ where each image x_i is assigned a label $\mathrm{y}_i \in \{0, 1\}^{N_{cls}}$. Training views are obtained as $\tilde{x}_{Ti} = t_{Ti}(x_i)$ and $\tilde{x}_{Si} = t_{Si}(x_i)$ for all data points, where $t_{Ti} \sim \mathcal{T}_T$ and $t_{Si} \sim \mathcal{T}_S$. Total loss $\mathcal{L}_T$ for DHD is computed with $\mathcal{X}_B$ as:

$$\mathcal{L}_T(\mathcal{X}_B) = \frac{1}{N_B} \sum_{n=1}^{N_B} (\mathcal{L}_{HP} + \lambda_1 \mathcal{L}_{SdH} + \lambda_2 \mathcal{L}_{bce\text{-}Q}) \tag{2.14}$$

where λ_1 and λ_2 are hyper-parameters that balance the influence of the training objectives. The entire DHD framework is trained in an end-to-end fashion.

Multi-label case. In the case of determining semantic similarity between multi-hot labeled images, the previous works [30, 34, 87] simply checked whether the images share at least one positive label or not. However, learning with the above similarity has limitations in that the label dependency is ignored. Thus, I aim to capture the intelligence that appears in label dependency by utilizing the Softmax cross entropy with the normalized multi-hot label y. Specifically, y is converted as $\mathrm{y} = \mathrm{y}/\|\mathrm{y}\|_1$ to balance the contribution of each label, and the same $\mathcal{L}_{HP}$ is computed to optimize the deep hashing model for multi-label image retrieval.

Algorithm. Detailed training process is provided in Algorithm 1 for reproducibility of DHD framework.

2.6.4 Hamming Distance Analysis

For a given input image x_i and x_j, a deep hashing model $\mathcal{H}$ produces corresponding hash codes h_i and h_j, which are quantized to binary codes b_i and b_j in $\{-1, 1\}^K$ with

Algorithm 1 DHD training for batch size N_B

Initialize θ_E with pretrained model weights.

Initialize θ_H and θ_P with Xavier initialization.

$g^{+}(h) = \exp\left(-\frac{(h-1)^2}{2\sigma^2}\right)$

$g^{-}(h) = \exp\left(-\frac{(h+1)^2}{2\sigma^2}\right)$

Input: Parameters of each component: $\theta_E, \theta_H, \theta_P$

Input: $\mathcal{X}_B = \{(x_1, \mathrm{y}_1), ..., (x_{N_B}, \mathrm{y}_{N_B})\}$

for n in $\{1,\ldots,N_B\}$ **do**

  draw two transformations $t_{2n-1} \sim \mathcal{T}_T, t_{2n} \sim \mathcal{T}_S$

  $\tilde{x}_{2n-1} \leftarrow t_{2n-1}(x_n)$

  $\tilde{x}_{2n} \leftarrow t_{2n}(x_n)$

  $\mathrm{h}_{2n-1} \leftarrow \tanh(H_{\theta_H}(E_{\theta_E}(\tilde{x}_{2n-1})))$

  $\mathrm{h}_{2n} \leftarrow \tanh((H_{\theta_H}(E_{\theta_E}(\tilde{x}_{2n})))$

  $\mathrm{p}_{2n-1} \leftarrow P_{\theta_P}(\mathrm{h}_{2n-1})$

end for

$\ell_{SdH} \leftarrow \mathcal{L}_{SdH}$ with $\{\mathrm{h}_{2n-1}, \mathrm{h}_{2n}\}_{n=1}^{N_B}$

$\ell_{HP} \leftarrow \mathcal{L}_{HP}$ with $\{\mathrm{p}_{2n-1}, y_n\}_{n=1}^{N_B}$

$\ell_{bce\text{-}Q} \leftarrow \mathcal{L}_{bce\text{-}Q}$ with $g^{+}, g^{-}, \{\mathrm{h}_{2n-1}\}_{n=1}^{N_B}$

$\theta_{E,H} \leftarrow \theta_{E,H} - \gamma\left(\frac{\partial \ell_{SdH}}{\partial \theta_{E,H}} + \frac{\partial \ell_{HP}}{\partial \theta_{E,H}} + \frac{\partial \ell_{BCE\text{-}Q}}{\partial \theta_{E,H}}\right)$

$\theta_P \leftarrow \theta_P - \gamma\frac{\partial \ell_{HP}}{\partial \theta_P}$

Output: Updated $\theta_E, \theta_H, \theta_P$

sign operation ($\mathrm{b}_i = \mathrm{sign}(\mathrm{h}_i)$, $\mathrm{b}_j = \mathrm{sign}(\mathrm{h}_j)$), respectively. For retrieval, Hamming distance $\mathcal{D}_{\mathrm{H}}$ is computed with the binary codes as:

$$\mathcal{D}_{\mathrm{H}}(\mathrm{b}_i, \mathrm{b}_j) = \mathrm{XOR}\left(\mathrm{b}_i, \mathrm{b}_j\right), \tag{2.15}$$

where XOR is a bit-wise count operation that outputs in the range [0, K]. From a mathematical point of view, XOR can be interpreted as:

$$\begin{aligned}\text{XOR}\left(\mathrm{b}_i, \mathrm{b}_j\right) &= \frac{1}{2}\left(K - \mathrm{b}_i^T \cdot \mathrm{b}_j\right) \\ &= \frac{1}{2}\left(K - \left\|\mathrm{b}_i\right\|_2 \left\|\mathrm{b}_j\right\|_2 \mathcal{S}\left(\mathrm{b}_i, \mathrm{b}_j\right)\right) \\ &= \frac{K}{2}\left(1 - \mathcal{S}\left(\mathrm{b}_i, \mathrm{b}_j\right)\right),\end{aligned} \tag{2.16}$$

where $\left\|\mathrm{b}_i\right\|_2 = \left\|\mathrm{b}_j\right\|_2 = \sqrt{K}$, and $\mathcal{S}\left(\cdot,\cdot\right)$ denotes cosine similarity which also can be notated as:

$$\begin{aligned}\mathcal{S}\left(\mathrm{b}_i, \mathrm{b}_j\right) &= \cos\alpha_{ij} \\ &= \frac{\mathrm{b}_i^T \cdot \mathrm{b}_j}{\left\|\mathrm{b}_i\right\|_2 \left\|\mathrm{b}_j\right\|_2},\end{aligned} \tag{2.17}$$

where α_{ij} is the angle between b_i and b_j. It should be noted that $1 - \mathcal{S}\left(\mathrm{b}_i, \mathrm{b}_j\right)$ presents a cosine distance between b_i and b_j and can be approximated with h_i and h_j as:

$$1 - \mathcal{S}\left(\mathrm{b}_i, \mathrm{b}_j\right) \simeq 1 - \mathcal{S}\left(\mathrm{h}_i, \mathrm{h}_j\right) \tag{2.18}$$

where $1 - \mathcal{S}\left(\mathrm{h}_i, \mathrm{h}_j\right)$ is a cosine distance between h_i and h_j. Therefore, minimizing the cosine distance allows to reduce the Hamming distance.

2.7 Experiments

To evaluate my DHD, I conduct image retrieval experiments against several conventional and modern methods. Three hashing based image retrieval benchmark datasets are explored, and I explain the composition of each dataset in Table 2.4.

Dataset	# Database	# Train	# Query	N_c
ImageNet	128,503	13,000	5,000	100
NUS-WIDE	149,736	10,500	2,100	21
MS COCO	117,218	10,000	5,000	80

Table 2.4: Description of the image retrieval datasets.

2.7.1 Setup

Following the protocol utilized in deep hashing methods [2, 94, 87], I adopt three benchmark datasets: single-labeled ImageNet, and multi-labeled NUS-WIDE, and MS COCO with the same compositions.

Evaluation metrics. To evaluate retrieval quality, I employ three metrics: 1) mean average precision (**mAP**), 2) precision-recall curves (**PR curves**), and 3) precision with respect to top-M returned image (**P@Top-**M). Regarding mAP score computation, I select the top-M images from the retrieval ranked-list results. The returned images and the query image are considered relevant whether one or more category labels are the same. I set binary code length: hash code dimensionality K as 16, 32, and 64, to examine the performance according to the code size.

2.7.2 Implementation Details

Data augmentation. Following the works presented in [4], I choose family $\mathcal{T}$ of five image transformations: 1) resized crop, 2) horizontal flip, 3) color jitter, 4) grayscale, and 5) blur, where all of each are sampled uniformly with a given probability and se-

Method	Backbone	ImageNet			NUS-WIDE			MS COCO		
		16-bit	32-bit	64-bit	16-bit	32-bit	64-bit	16-bit	32-bit	64-bit
ITQ [5]	Non-deep	0.266	0.436	0.576	0.435	0.396	0.365	**0.566**	0.562	0.502
SH [28]		0.210	0.329	0.418	0.401	0.421	0.423	0.495	0.507	0.510
KSH [28]		0.160	0.298	0.394	0.394	0.407	0.399	0.521	0.534	0.536
SDH [22]		**0.299**	**0.455**	**0.585**	**0.575**	**0.590**	**0.613**	0.554	**0.564**	**0.580**
CNNH [30]	AlexNet [14]	0.315	0.473	0.596	0.655	0.659	0.647	0.599	0.617	0.620
DHN [34]		0.367	0.522	0.627	0.712	0.739	0.751	0.701	0.710	0.735
HashNet [2]		0.425	0.559	0.649	0.720	0.745	0.758	0.685	0.714	0.742
DCH [94]		0.636	0.645	0.656	0.740	0.752	0.763	0.695	0.721	0.748
DHD(Mine)		**0.657**	**0.701**	**0.721**	**0.780**	**0.805**	**0.820**	**0.749**	**0.781**	**0.792**
DPN [95]	ResNet [6]	0.828	0.863	0.872	0.783	0.816	0.838	0.796	0.838	0.861
CSQ [87]		0.851	0.865	0.873	0.810	0.825	0.839	0.750	0.824	0.852
DHD(Mine)		**0.864**	**0.891**	**0.901**	**0.820**	**0.839**	**0.850**	**0.839**	**0.873**	**0.889**
DHD(Mine)	ViT [96]	0.927	0.938	0.944	0.837	0.862	0.870	0.886	0.919	0.939
	DeiT [93]	0.932	0.943	0.948	0.839	0.861	0.867	0.883	0.913	0.925
	SwinT [97]	**0.944**	**0.955**	**0.956**	**0.848**	**0.867**	**0.875**	**0.894**	**0.930**	**0.945**

Table 2.5: mAP scores for different bits on three benchmark image datasets.

quentially applied to the inputs. I keep the internal parameters of each transformation equal to [4]. For self-distilled hashing, I configure two groups with $\mathcal{T}$, where the difficult student group is $\mathcal{T}_S = \mathcal{T}$, and the easy teacher group $\mathcal{T}_T$ is configured by scaling all transform occurrence with s_T, which is in the range of $(0, 1]$. I set $\mathcal{T}_T$ as the default for the methods trained without SdH.

Experiments. Retrieval experiments are conducted by dividing backbones as: Non-deep, AlexNet [14], ResNet (ResNet50) [6] , and vision transformers [96, 93, 97]. For non-deep hashing approaches: ITQ [5], SH [28], KSH [20] and SDH [22], I report the results directly from the latest works [2, 94, 87] for comparison. I set up the same training environment by leveraging PyTorch framework and the image transformation functions of korina [72] library for augmentation. I employ Adam optimizer [13] and decay the learning rate with cosine scheduling [79] for training deep hashing methods. Especially for DHD hyper-parameters, s_T is set to 0.2 for AlexNet, and 0.5 for other backbones. τ is set by considering N_{cls} as $\{0.2, 0.6, 0.4\}$ for {ImageNet, NUS-WIDE, MS COCO}, respectively. λ_1 and λ_2 are set equal to 0.1 for a balanced contributions each training objective, and σ in $\mathcal{L}_{bce\text{-}Q}$ is set to 0.5 as default.

2.7.3 Results

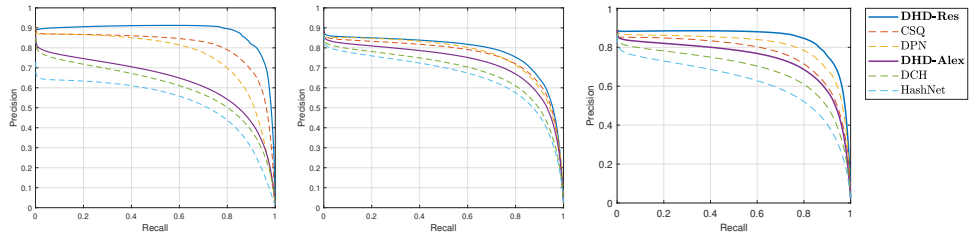
The mAP scores are calculated by varying the top-M for each dataset as: ImageNet@1000, NUS-WIDE@5000 and MS COCO@5000 to make a fair comparison with previous works [2, 94, 87]. The results are listed in Table 2.2, where the highest score for each backbone is shown in bold, and I my DHD method. Among the non-deep hashing methods, SDH shows the best retrieval results by employing supervised label signals in hash function learning. Deep hashing methods generally outperform non-deep hashing ones, since elaborately labeled annotations are fully utilized during training. For ImageNet, NUS-WIDE, and MS COCO, averaging the mAP scores of all bit lengths yields 36.3%p, 33.7%p, and 25.0%p differences between the non-deep and deep methods, respectively.

Notably, my DHD shows the best mAP scores for all datasets in every bit length with every deep backbone architecture. In particular for AlexNet backbone hashing approaches, DHD shows performance improvement of 16.3%p, 7.9%p, and 9.2%p by averaging the mAP scores of all bit lengths in three dataset results orderly, compared to others. In comparison with ResNet backbone methods, DHD also achieves 2.7%p, 1.8%p, and 4.7%p higher retrieval scores on average. In line with the trend of other computer vision tasks, I introduce transformer-based image representation learning architectures: ViT [96], DeiT [93], and SwinT [97] to the hashing community and perform retrieval experiments. As reported, when the transformer is integrated into the DHD framework, it delivers outstanding results for the benchmark image datasets with the increase of 5.8%p, 2.2%p, and 4.8%p, in the same as above, compared to ResNet backbone DHD.

To further demonstrate that DHD genuinely provides quality search outcomes, I plot graph of the PR curve and the precision for the top 1,000 retrieved images at 64 bits in Figures 2.9 and 2.10. As shown in the experimental results, I can confirm that the proposed DHD establishes the state-of-the-art retrieval performance.

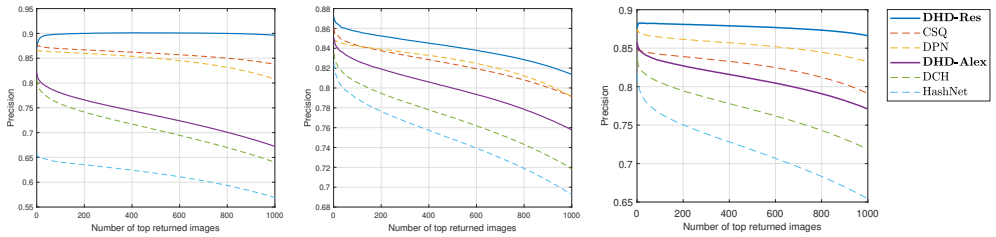
	$\mathcal{L}_{HP}$	$\mathcal{L}_{SdH}$	$\mathcal{L}_{bce\text{-}Q}$	$\mathcal{T}$	mAP	
					16-bit	64-bit
(1)	✓			$\mathcal{T}_T$	0.640	0.702
(2)	✓		✓	$\mathcal{T}_T$	0.645	0.709
(3)	✓			$\mathcal{T}_S$	0.569	0.639
(4)	✓		✓	$\mathcal{T}_S$	0.585	0.647
(5)	✓	✓		$\mathcal{T}_T + \mathcal{T}_S$	0.650	0.712
(6)	✓	$Eud.$	✓	$\mathcal{T}_T + \mathcal{T}_S$	0.640	0.698
(7)	✓	✓	✓	$\mathcal{T}_T + \mathcal{T}_S$	**0.657**	**0.721**

Table 2.6: Ablation study on my work with ImageNet. ✓indicates that the corresponding element is used. $\mathcal{T}$ denotes types of augmentation, and $Eud.$ is the abbreviation of Euclidean distance.



(a) CIFAR-10 (b) FLICKR25K (c) NUS-WIDE

Figure 2.9: Precision-Recall curves on benchmarks with binary codes @ 64-bits.



(a) CIFAR-10 (b) FLICKR25K (c) NUS-WIDE

Figure 2.10: Precision@top-1000 curves on benchmarks with binary codes @ 64-bits.

Ablation study. In Table 2.6, I show the experimental results according to the presence or absence of my contributions. To minimize the dependency on the network architecture and explore my contribution correctly, I use AlexNet as the backbone. In (1), even when only $\mathcal{L}_{HP}$ is applied, it shows good results compared to the AlexNet backbone existing methods in Table 2.2. In (2, 6, 7), search accuracy is improved in all

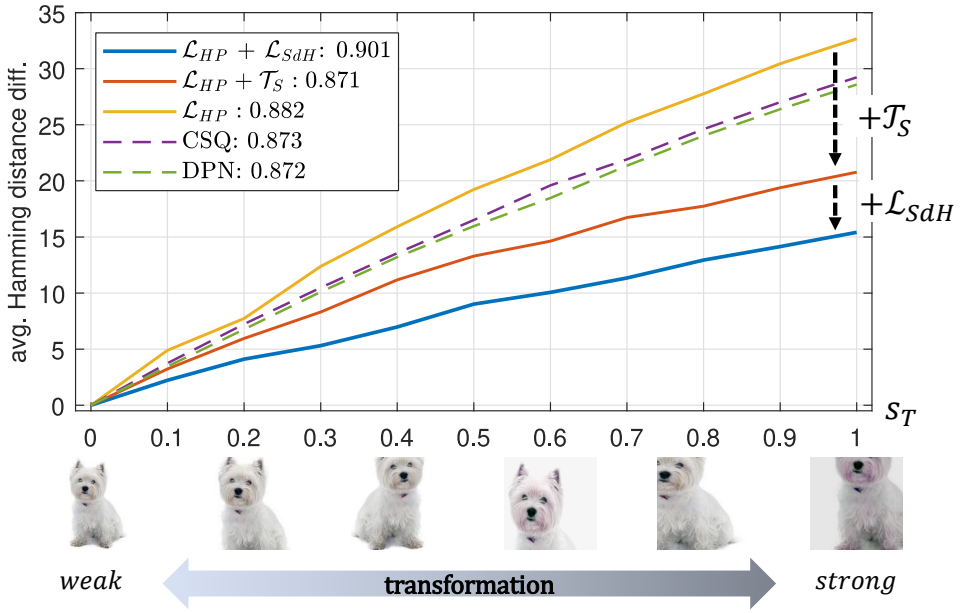


Figure 2.11: Average Hamming distance difference between the original and transformed images of ImageNet query set. By varying the s_T, I measure the sensitivity to transformation of ResNet backbone methods where the numbers in legend indicate mAP. Solid lines present DHD variants, and dotted lines present others. $+\mathcal{T}_S$ denotes strong student augmentation is applied during training. A low slope indicates insensitivity to various transformations.

cases where $\mathcal{L}_{bce\text{-}Q}$ is applied. In (3, 4) the results with strong student augmentation $\mathcal{T}_S$ show that simply applying it to training degrades the performance due to the emerged discrepancy in representation between Hamming and real space, while SdH mitigates this problem and improves performance by properly exploiting the power of data augmentation, as shown in (5, 6, 7). In order to check whether the cosine distance is the best choice for SdH or not, I replace the cosine distance in Eqn 2.8 with the Euclidean distance and report the results in (6), but the results in (7) demonstrates that the cosine distance is better. To summarize the ablation, every proposal affects the performance improvement, and the best is achieved when all of them are combined as in (7).

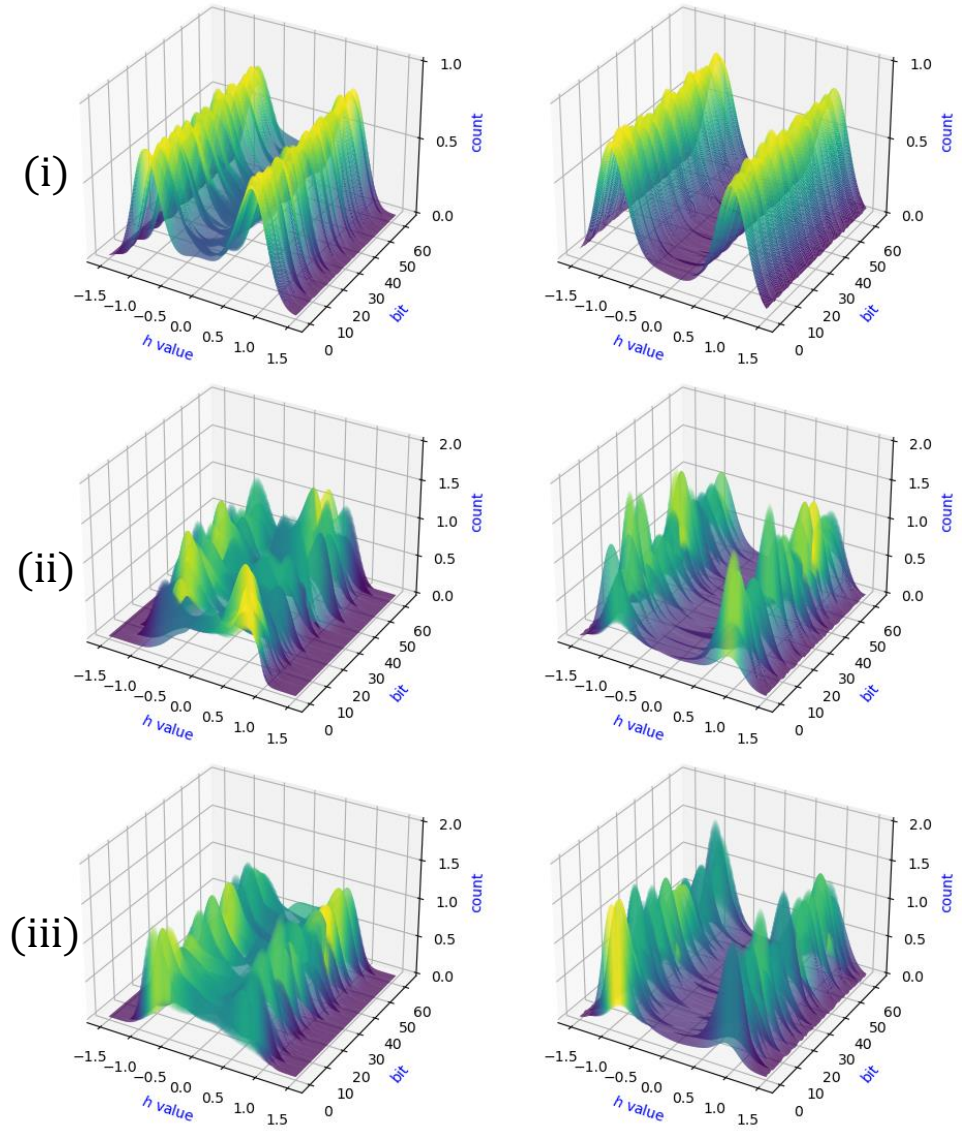
Method	ImageNet				NUS-WIDE				MS COCO			
	with SdH		without SdH		with SdH		without SdH		with SdH		without SdH	
	16-bit	64-bit	16-bit	64-bit	16-bit	64-bit	16-bit	64-bit	16-bit	64-bit	16-bit	64-bit
HashNet [2]	0.501	0.661	0.337	0.502	0.745	0.769	0.705	0.762	0.695	0.753	0.655	0.727
DCH [94]	0.640	0.673	0.571	0.597	0.754	0.771	0.748	0.767	0.703	0.746	0.669	0.697
DPN [95]	0.630	0.708	0.562	0.656	0.757	0.801	0.753	0.787	0.710	0.772	0.672	0.760
CSQ [87]	0.634	0.711	0.570	0.662	0.759	0.804	0.757	0.793	0.707	0.765	0.670	0.752
DHD(Mine)	**0.657**	**0.721**	**0.583**	**0.671**	**0.780**	**0.820**	**0.775**	**0.806**	**0.749**	**0.792**	**0.731**	**0.766**

Table 2.7: mAP scores with or without Self-distilled Hashing (SdH).

2.7.4 Analysis

Insensitivity to transformations. To investigate the sensitivity to transformations, I examine how the binary code shifts when transformed images are fed to ResNet backbone methods, by using ImageNet query set. I measure the average Hamming distance between the untransformed ($s_T = 0$) binary codes and the transformed (s_T in (0, 1]), binary codes. As observed in Figure 2.3, CSQ DPN, and a model learned with $\mathcal{L}_{HP}$ are trained with $\mathcal{T}_T$, showing sensitivity to transformations due to barely used augmentation. When the augmentation is applied ($\mathcal{L}_{HP} + \mathcal{T}_S$), model is improved to be more robust to transformations, however, the map score decreases due to the discrepancy in representation between Hamming and Real space during training. On the other hand, the combined $\mathcal{L}_{HP} + \mathcal{L}_{SdH}$ exhibits the highest robustness while achieving the best mAP score, by minimizing discrepancy and successfully exploring the potential of strong augmentation.

Self-distilled Hashing with other methods. In order to prove that SdH can be applied to other deep hashing baselines [2, 94, 95, 87], I perform retrieval experiments with AlexNet backbone and show the results in Table 3.1. With SdH setup, I employ $\mathcal{T}_T$ and $\mathcal{T}_S$ groups to produce input views, and for without SdH setup, I only use $\mathcal{T}_S$ to generate input views. By comparing the reported results of [2, 94] and DHD in Table 2.2 and the results without SdH in Table 3.1, I can see that the deep hashing model learned with $\mathcal{T}_S$ is inferior to the model learned with $\mathcal{T}_T$, which shows that the direct use of $\mathcal{T}_S$ degrades performance. Otherwise, if the model uses SdH to leverage the



(a) without $\mathcal{L}_{bce\text{-}Q}$ (b) with $\mathcal{L}_{bce\text{-}Q}$

Figure 2.12: 3D visualized histograms to verify the impact of $\mathcal{L}_{bce\text{-}Q}$. Gallery sets of (i) ImageNet, (ii) NUS-WIDE, and (iii) MS COCO are utilized. x-axis presents value of hash element h, y-axis presents bit position, and z-axis presents frequency counts.

power of strong augmentation, then the retrieval performance can be improved.

Quantization. The effect of $\mathcal{L}_{bce\text{-}Q}$ is plotted in Figure 2.12. The binary bits are observed to be distributed more evenly and binary-like when $\mathcal{L}_{bce\text{-}Q}$ is applied. This implies that the bit entropy is much higher, which shows better retrieval quality by representing diverse binary codes.

Deformation	with SdH	without SdH
None	**0.891** (2.3% ↑)	0.871
Cutout	**0.862** (3.7% ↑)	0.827
Dropout	**0.810** (7.9% ↑)	0.765
Zoom in	**0.658** (19.0% ↑)	0.552
Zoom out	**0.816** (1.4% ↑)	0.805
Rotation	**0.856** (2.4% ↑)	0.836
Shearing	**0.842** (2.7% ↑)	0.815
Gaussian noise	**0.768** (10.5% ↑)	0.673

Table 2.8: mAP scores on unseen deformations.

Robustness to unseen deformations. To further examine the generalization capacity of DHD, I conduct experiments with unseen (not seen during training) transformations to inputs following the evaluation protocol utilized in [98]. As reported in Table 3.2, deep hashing model with SdH significantly outperforms the model without SdH at all deformations, showing a performance difference of up to 19% (zoom in). In particular, SdH makes deep hashing model robust to per-pixel deformations such as dropout and Gaussian noise, even though SdH has not included any pixel-level transformations.

Qualitative Results. In order to see whether the difficulty of transformation is really different according to s_T, I illustrate Figure 2.13. For $\mathcal{T}_T$, images do not significantly deviated from the original. However, for $\mathcal{T}_S$, some images are distorted to the point of being hard to recognize. Figure 2.14 shows what images are actually retrieved as results when transformation is applied. I can confirm that DHD is robust to transformation and achieves high quality results.

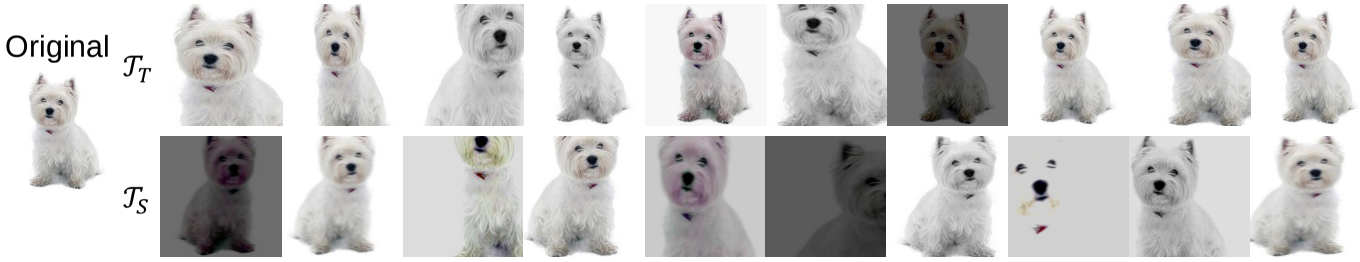


Figure 2.13: Visualized augmentation results of different groups: weakly-transformed teacher $\mathcal{T}_T$ and strongly-transformed student $\mathcal{T}_S$.

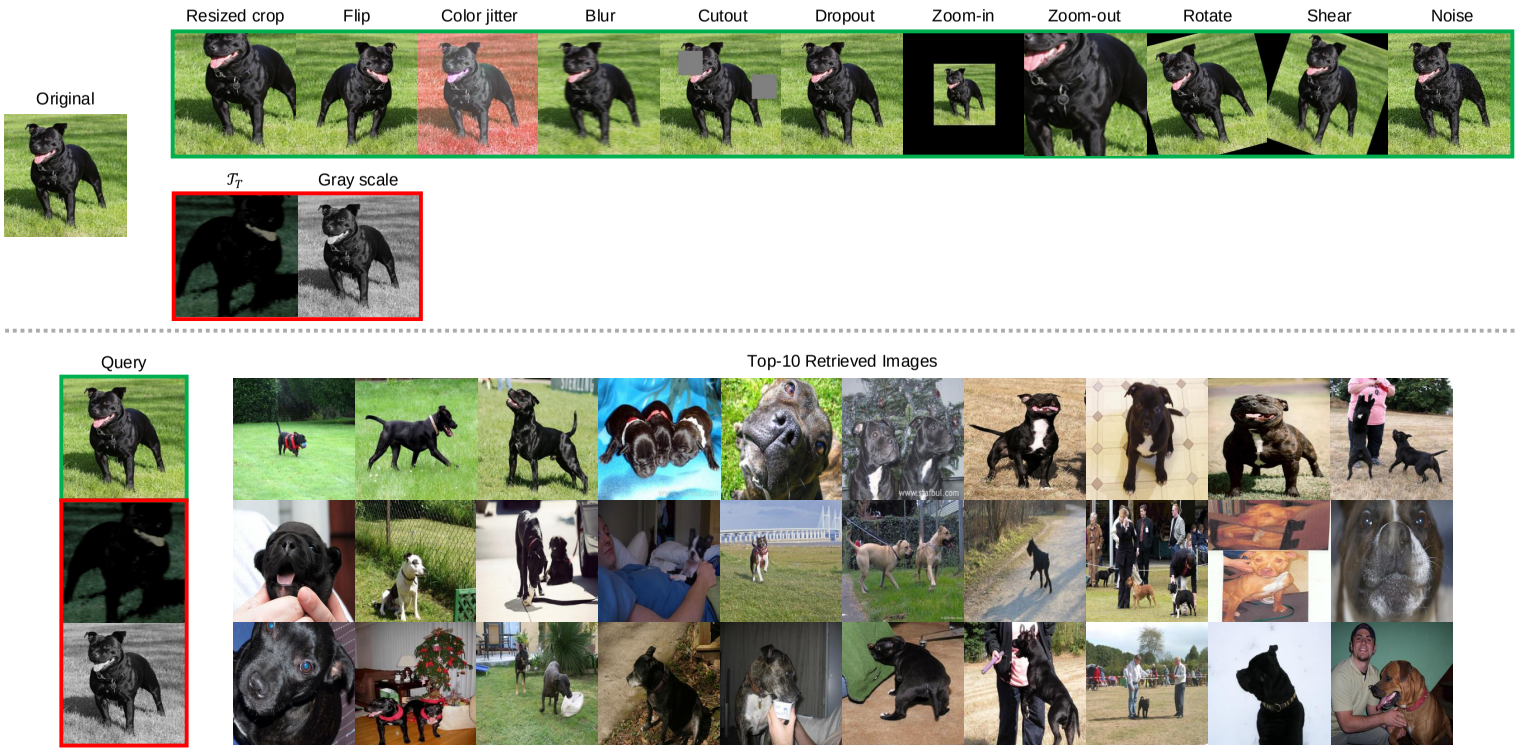


Figure 2.14: Above: Examples of various transformations applied to the original image. The green box indicates the same search result as the original, and the red box indicates otherwise. Below: Retrieved images on ImageNet dataset. The image output form the strongly transformed $\mathcal{T}_T$ and the image transformed to gray scale show different retrieval results. Nevertheless, it can be seen that visually similar images of the same content are retrieved well.

Chapter 3

Semi-supervised Learning for Product Quantization: Generalized Product Quantization Network for Semi-supervised Image Retrieval

3.1 Motivation and Overview

The amount of multimedia data, including images and videos, increases exponentially on a daily basis. Hence, retrieving relevant content from a large-scale database has become a more complicated problem. There have been many kinds of fast and accurate search algorithms, and the Approximate Nearest Neighbor (ANN) search is known to have high retrieval accuracy and computational efficiency. Recent ANN methods mainly focused on *Hashing* scheme [27], because of its low storage cost and fast retrieval speed. To be specific, an image is represented by a binary-valued compact hash code (*binary code*) with only a few tens of bits, and it is utilized to build database and distance computation.

The methods using binary code representation can be categorized as *Binary Hashing* (BH) and *Product Quantization* (PQ) [45]. BH-based methods [28, 5, 22] employ a hash function that maps a high-dimensional vector space to a Hamming space, where the distance between two codes can be measured extremely fast via bitwise XOR op-

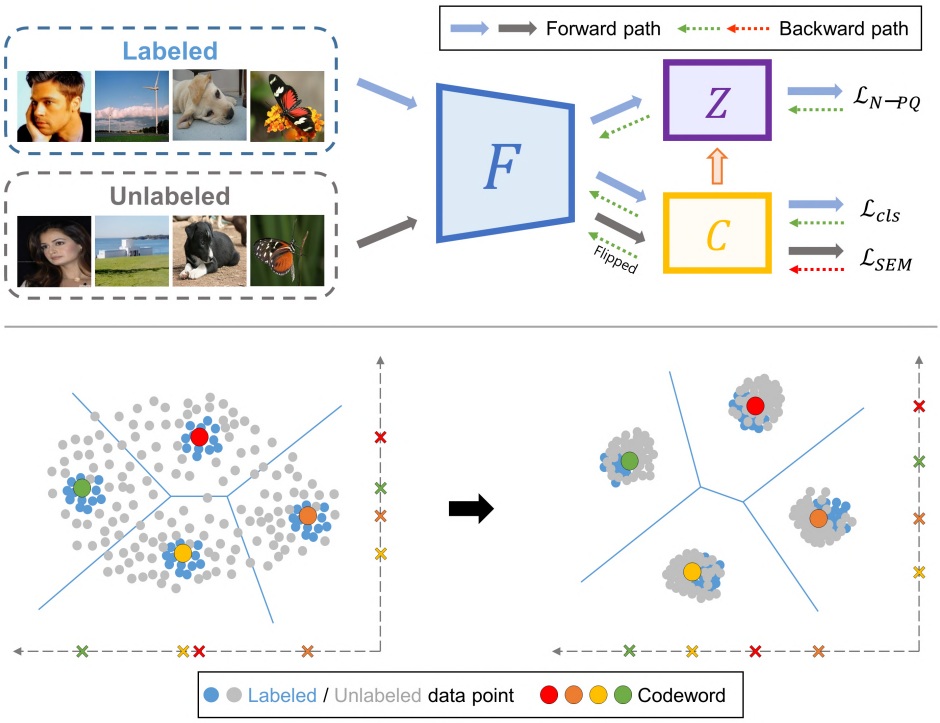


Figure 3.1: Above: an illustration of the overall framework of GPQ and its three components: feature extractor F, PQ table Z, and classifier C, where C contributes to build Z.

eration. However, BH has a limitation in describing the distance between data points because it can produce only a limited number of distinct values. PQ, which is a kind of vector quantization, has been introduced to alleviate this problem in information retrieval [45, 40, 47].

To perform PQ, I first need to decompose the input feature space into a Cartesian product of several disjoint subspaces (*codebooks*) and find the centroid (*codeword*) of each subspace. Then, from the sub-vectors of the input feature vector, sub-binary code is obtained by replacing each sub-vector with the index of the nearest codeword in the codebook. Since codeword consists of real numbers, PQ allows asymmetric distance calculation in real space using the binary codes, making many PQ-based approaches outperform BH-based ones.

Along with millions of elaborately labeled data, *deep Hashing* for both BH [30, 67, 43, 44, 46] and PQ [37, 49, 64, 48] has been introduced to take advantage of deep representations for image retrieval. By employing supervised deep neural networks, deep Hashing outperforms conventional ones on many benchmark datasets. Nevertheless, there still is a great deal of potential for improvement, since a significant amount of unlabeled data with abundant knowledge is not utilized. To resolve these issue, some recent methods are considering the *Deep Semi-Supervised Hashing*, based on BH [66, 62, 42]. However, even if PQ generally outperforms BH for both supervised and unsupervised settings, it has not yet been considered for learning in a semi-supervised manner. In this dissertation, I propose the first PQ-based deep semi-supervised image retrieval approach: Generalized Product Quantization (GPQ) network, which significantly improves the retrieval accuracy with lots of image data and just a few labels per category (class).

Existing deep semi-supervised BH methods construct graphs [66, 62] or apply additional generative models [42] to encode unlabeled data into the binary code. However, due to the fundamental problem in BH; a deviation that occurs when embedding a continuous deep representation into a discrete binary code restricts extensive information of unlabeled data. In my GPQ framework, this problem is solved by including quantization process into the network learning. I adopt intra-normalization [36] and soft assignment [64] to quantize real-valued input sub-vectors, and introduce an efficient metric learning strategy; *N-pair Product Quantization loss* inspired from [57]. By this, I can embed multiple pair-wise semantic similarity between every feature vectors in a training batch into the codewords. It also has an advantage of not requiring any complex batch configuration strategy to learn pairwise relations.

The key point of deep semi-supervised retrieval is to avoid overfitting to labeled data and increase the generalization toward unlabeled one. For this, I suggest a *Subspace Entropy Mini-max Loss* for every codebook in GPQ, which regularizes the network using unlabeled data. Precisely, I first learn a cosine similarity-based classifier,

which is commonly used in few-shot learning [56, 60]. The classifier has as many weight matrices as the number of codebooks, and each matrix contains class-specific weight vector, which can be regarded as a *sub-prototype* that indicates class representative centroid of each codebook. Then, I compute the entropy between the distributions of sub-prototypes and unlabeled sub-vectors by measuring their cosine similarity. By maximizing the entropy, the two distributions become similar, allowing the sub-prototypes to move closer to the unlabeled sub-vectors. At the same time, I also minimize the entropy of the distribution of the unlabeled sub-vectors, making them assemble near the moved sub-prototypes. With the gradient reversal layer generally used for deep domain adaptation [39, 54], GPQ can simultaneously minimize and maximize the entropy during the network training.

3.1.1 Related Work

Existing Hashing Methods Referring to the survey [27], early works in Binary Hashing (BH) [28, 5, 22, 59] and Product Quantization (PQ) [45, 40, 47, 41, 51, 63, 58] mainly focused on unsupervised settings. Specifically, Spectral Hashing (SH) [28] considered correlations within hash functions to obtain balanced compact codes. Iterative Quantization (ITQ) [5] addressed the problem of preserving the similarity of original data by minimizing the quantization error in hash functions. There were several studies to improve PQ, for example, Optimized Product Quantization (OPQ) [40] tried to improve the space decomposition and codebook learning procedure to reduce the quantization error. Locally Optimized Product Quantization (LOPQ) [47] employed a coarse quantizer with locally optimized PQ to explore more possible centroids. These methods might reveal some distinguishable results, however, they still have disadvantage of not exploiting expensive label signals.

Deep Hashing Methods After Supervised Discrete Hashing (SDH) [22] has shown the capability to improve using labels, supervised Convolutional Neural Network (CNN)-

based BH approaches [30, 67, 43, 44, 46] are leading the mainstream. For examples, CNN Hashing (CNNH) [30] utilized a CNN to simultaneously learn feature representation and hash functions with the given pairwise similarity matrix. Network in Network Hashing (NINH) [67] introduced a sub-network, divide-and-encode module, and a triplet ranking loss for the similarity-preserving hashing. A Supervised, Structured Binary Code (SUBIC) [43], used a block-softmax nonlinear function and computed batch-based entropy error to embed the structure into a binary form. There also have been researches on supervised learning that uses PQ with CNN [37, 49, 64, 48]. Precisely, Deep Quantization Network (DQN) [37] simultaneously optimizes a pairwise cosine loss on semantic similarity pairs to learn feature representations and a product quantization loss to learn the codebooks. Deep Triplet Quantization (DTQ) [49] designed a group hard triplet selection strategy and trained triplets by triplet quantization loss with weak orthogonality constraint. Product Quantization Network (PQN) [64] applied the asymmetric distance calculation mechanism to the triplets and exploited softmax function to build a differentiable soft product quantization layer to train the network in an end-to-end manner. My method is also based on PQ, but I try a *semi*-supervised PQ scheme that had not been considered previously.

Deep Semi-supervised Image Retrieval Assigning labels to images is not only expensive but also has the disadvantage of restricting the data structure to the labels. Deep semi-supervised hashing based on BH is being considered in the image retrieval community to alleviate this problem, with the use of a small amount of labeled data and a large amount of unlabeled data. For example, Semi-supervised Deep Hashing (SSDH) [66] employs an online graph construction strategy to train a network using unlabeled data. Deep Hashing with a Bipartite Graph (BGDH) [62] improved SSDH by using the bipartite graph, which is more efficient in building a graph and learning the embeddings. Since Generative Adversarial Network (GAN) had been used for BH and showed good performance as in [46], Semi-supervised Generative Adversar-

ial Hashing (SSGAH) also employed the GAN to fully utilize triplet-wise information of both labeled and unlabeled data. In this dissertation, I propose GPQ, the first deep semi-supervised image retrieval method applying PQ. In my work, I endeavor to generalize the whole network by preserving the semantic similarity with N-pair Product Quantization loss and extracting the underlying structure of unlabeled data with the Subspace Entropy Mini-max loss.

3.2 Generalized Product Quantization

Given a dataset $\mathcal{X}$ that is composed of individual images, I split this into two subsets as a labeled dataset $\mathcal{X}^L = \{(I_i^L, y_i)|i = 1, ..., N^L\}$ and an unlabeled dataset $\mathcal{X}^U = \{I_i^U|i = 1, ..., N^U\}$ to establish a semi-supervised environment. The goal of my work is learning a quantization function $\mathrm{q} : I \rightarrow \hat{\mathbf{b}} \in \{0, 1\}^{\mathcal{B}}$ which maps a high-dimensional input I to a compact $\mathcal{B}$-bits binary code $\hat{\mathbf{b}}$, by utilizing both labeled and unlabeled datasets. I propose a semi-supervised deep Hashing framework: GPQ, which integrates q into the deep network as a form of Product Quantization (PQ) [45] to learn the deep representations and the codewords jointly. In the learning process, I aim to preserve the semantic similarity of labeled data and simultaneously explore the structures of unlabeled data to obtain high retrieval accuracy.

GPQ contains three trainable components: 1) a standard deep convolutional neural network-based feature extractor F, e.g. AlexNet [14], CNN-F [38] or modified version of VGG [44] to learn deep representations; 2) a PQ table Z that collects codebooks which are used to map an extracted feature vector to a binary code; 3) a cosine similarity-based classifier C to classify both labeled and unlabeled data. GPQ network is designed to train all these components in an end-to-end manner. In this section, I will describe each component and how GPQ is learned in a semi-supervised way.

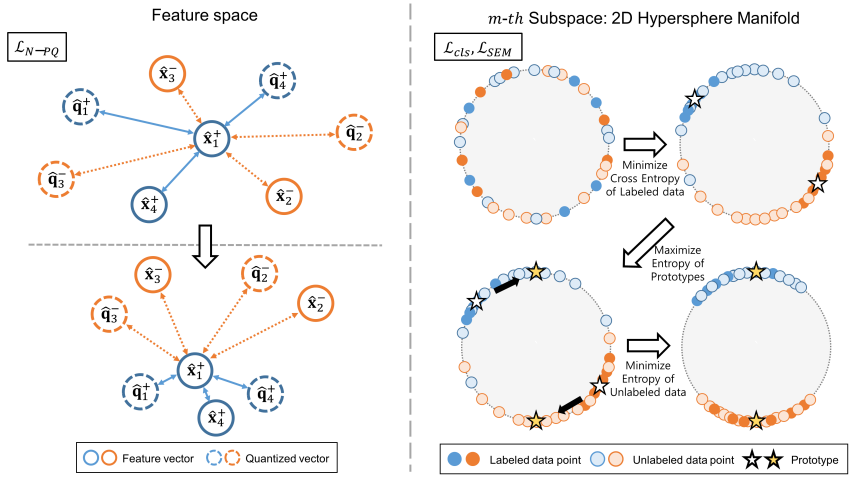


Figure 3.2: A two class (+: blue, −: orange) visualized examples of my training objectives. 1) The left part shows the learning process of N-pair Product Quantization loss $\mathcal{L}_{N\text{-}PQ}$. When I define an anchor as $\hat{\mathbf{x}}_1^+$, the semantically similar points $(\hat{\mathbf{q}}_1^+, \hat{\mathbf{x}}_4^+, \hat{\mathbf{q}}_4^+)$ are pulled together while the semantically dissimilar points $(\hat{\mathbf{x}}_2^-, \hat{\mathbf{q}}_2^-, \hat{\mathbf{x}}_3^-, \hat{\mathbf{q}}_3^-)$ are pushing the anchor. 2) The right part shows the learning process of classification loss $\mathcal{L}_{cls}$ and subspace entropy mini-max loss $\mathcal{L}_{SEM}$. For the data points constrained on the unit hypersphere, the cross entropy of labeled data points is minimized to find prototypes (white stars). Then, the entropy between the prototypes and the unlabeled data points is maximized to move prototypes toward unlabeled data points and find new prototypes (yellow stars). Finally, the entropy of the unlabeled data points is minimized to cluster them near the new prototypes.

3.2.1 Semi-Supervised Learning

The feature extractor F generates D-dimensional feature vector $\hat{\mathbf{x}} \in R^D$. Under semi-supervised learning condition, I aim to train the F to extract discriminative $\hat{\mathbf{x}}^L$ and $\hat{\mathbf{x}}^U$ from labeled image I^L and unlabeled image I^U, respectively. Also, I leverage the PQ concept to utilize these feature vectors for image retrieval, which requires appropriate codebooks with distinct codewords to replace and store the feature vectors. I introduce three training objectives for my GPQ approach to fully exploit the data structure of labeled and unlabeled images, and I illustrate a conceptual visualization of each loss function in Figure 3.2 for better understanding.

Following the observation of [56, 60, 64, 49], I normalize the feature vectors and

constrain them on a unit hypersphere to focus on the angle rather than the magnitude in measuring the distance between two different vectors. In this way, every data is mapped to the nearest class representative direction, and better performs for the semi-supervised scheme because the distribution divergence between labeled and unlabeled data can be reduced within the constraint. Especially for PQ, I apply intra-normalization [36] for a feature vector $\hat{\mathbf{x}}$ by dividing it into M-sub-vectors $\hat{\mathbf{x}} = [\mathbf{x}_1, ..., \mathbf{x}_M]$, where $\mathbf{x}_m \in R^d, d = D/M$, and $l2$-normalize each sub-vector as: $\mathbf{x}_m \leftarrow \mathbf{x}_m / ||\mathbf{x}_m||_2$. In the rest of the dissertation, $\hat{\mathbf{x}}$ for GPQ denotes the intra-normalized feature vector.

N-pair Product Quantization The PQ table Z collects M-codebooks $Z = [\mathbf{Z}_1, ..., \mathbf{Z}_M]$ and each codebook has K-codewords $\mathbf{Z}_m = [\mathbf{z}_{m1}, ..., \mathbf{z}_{mK}]$ where $\mathbf{z}_{mk} \in R^d$, which are used to replace $\hat{\mathbf{x}}$ with the quantized vector $\hat{\mathbf{q}}$. Every codeword is $l2$-normalized to simplify the measurement of cosine similarity as multiplication. I employ the soft assignment $s_m(\cdot)$ [64] to obtain a $\mathbf{q}_m$ from $\mathbf{x}_m$ as:

$$\mathbf{q}_m = \sum_{k}^{K} \frac{e^{-\alpha(\mathbf{x}_m \cdot \mathbf{z}_{mk})}}{\sum_{k'}^{K} e^{-\alpha(\mathbf{x}_m \cdot \mathbf{z}_{mk'})}} \mathbf{z}_{mk} \tag{3.1}$$

where α represents a scaling factor to approximate hard assignment, and $\mathbf{q}_m = s_m(\mathbf{x}_m; \alpha, \mathbf{Z}_m)$ is the sub-quantized vector of $\hat{\mathbf{q}} = [\mathbf{q}_1, ..., \mathbf{q}_M]$. I multiply $\mathbf{x}_m$ with the every codeword in $\mathbf{Z}_m$ to measure the cosine similarity between them.

Quantization error occurs through the encoding process; therefore, I need to find the codewords that will minimize the error. In addition, the conventional PQ scheme has a limitation that it ignores label information since the sub-vectors are clustered to find codewords without any supervised signals. To fully exploit the semantic labels and reduce the quantization error, I revise the metric learning strategy proposed in [57] from *N-pair Product Quantization loss*: $\mathcal{L}_{N\text{-}PQ}$ to learn the F and Z, and set it as one of the training objectives.

Deep metric learning [61, 55] aims to learn an embedding representation of the data with the semantic labels. From a labeled image I^L, I can generate a unique feature vector $\hat{\mathbf{x}}^L$ and its nearest quantized vector $\hat{\mathbf{q}}^L$, which can be regarded as sharing the same semantic information. Thus, for randomly sampled B training examples $\{(I_1^L, y_1), ..., (I_B^L, y_B)\}$, the objective function which is based on a standard cross entropy loss $\mathcal{L}_{CE}$, can be formulated as:

$$\mathcal{L}_{N\text{-}PQ} = \frac{1}{B}\sum_{b=1}^{B}\mathcal{L}_{CE}(\mathcal{S}_b, \mathbf{Y}_b) \tag{3.2}$$

where $\mathcal{S}_b = \left[(\hat{\mathbf{x}}_b^L)^T\hat{\mathbf{q}}_1^L, ..., (\hat{\mathbf{x}}_b^L)^T\hat{\mathbf{q}}_B^L\right]$ denotes a cosine similarity between b-th feature vector and every quantized vector, and $\mathbf{Y}_b = \left[(\mathbf{y}_b)^T\mathbf{y}_1, ..., (\mathbf{y}_b)^T\mathbf{y}_B\right]$ denotes a similarity between b-th label and every label in a batch. In this case, $\mathbf{y}$ represents one-hot-encoded semantic label, and column-wise normalization is applied to $\mathbf{Y}_B$. $\mathcal{L}_{N\text{-}PQ}$ has an advantage in that no complicated batch construction method is required, and it also allows us to jointly learn the deep feature representation for both feature vectors and the codewords on the same embedding space.

Cosine Similarity-based Classification To embed semantic information into the codewords while reducing correlation between each codebook, I learn a cosine similarity-based classifier C containing M-weight matrices $[\mathbf{W}_1, ..., \mathbf{W}_M]$, where each matrix includes sub-prototypes as $\mathbf{W}_m = [\mathbf{c}_{m1}, ..., \mathbf{c}_{mN^c}]$, $\mathbf{W}_m \in R^{d\times N^c}$ and N^c is the number of class. Every sub-prototype is $l2$-normalized to hold a class-specific angular information. With the m-th sub-vector $\mathbf{x}_m^L$ of $\mathbf{x}^L$ and m-th weight matrix $\mathbf{W}_m$, I can obtain the labeled class prediction as: $\mathbf{p}_m^L = \mathbf{W}_m^T\mathbf{x}_m^L$. I use $\mathcal{L}_{CE}$ again to train the F and C for classification using $\hat{\mathbf{p}}^L = [\mathbf{p}_1^L, ..., \mathbf{p}_M^L]$ computed from x^L with the corresponding semantic label y as:

$$\mathcal{L}_{cls} = \frac{1}{M} \sum_{m=1}^{M} \mathcal{L}_{CE}(\beta \cdot \mathbf{p}_m^L, y) \tag{3.3}$$

where β is a scaling factor and y is a label corresponding to x^L. This classification loss ensures the feature extractor to generate the discriminative features with respect to the labeled examples. In addition, each weight matrix in the classifier is derived to include the class-specific representative sub-prototypes of related subspace.

Subspace Entropy Mini-max On the assumption that the distribution is not severely different between the labeled data and the unlabeled data, I aim to propagate the gradients derived from the divergence between them. To calculate the error arising from the distribution difference, I adopt an entropy in information theory. In particular for PQ setting, I compute the entropy for each subspace to balance the amount of gradients propagating into each subspace. With the m-th sub-vector $\mathbf{x}_m^U$ of the unlabeled feature vector $\mathbf{x}^U$, and the m-th weight matrix of C that embraces sub-prototypes, I can obtain a class prediction using cosine similarity as: $\mathbf{p}_m^U = \mathbf{W}_m^T \mathbf{x}_m^U$. By using it, the subspace entropy mini-max loss is calculated as:

$$\mathcal{L}_{SEM} = -\frac{1}{M} \sum_{m=1}^{M} \sum_{l=1}^{N^c} (\beta \cdot p_{ml}^U) \log(\beta \cdot p_{ml}^U) \tag{3.4}$$

where β is the same as that in Equation 3.3 and p_{ml}^U denotes the probability of prediction to k'-th class; l-th element of $\mathbf{p}_m^U$. The generalization capacity of the network can increase by maximizing the $\mathcal{L}_{SEM}$ because high entropy ensures that the sub-prototypes are regularized toward unlabeled data. Explicitly, entropy maximization makes sub-prototypes have a similar distribution with the unlabeled sub-vectors, moving sub-prototypes near the unlabeled sub-vectors. To further improve, I aim to cluster unlabeled sub-vectors near the moved sub-prototype, by applying gradient reversal

layer [39, 54] before intra-normalization. Flipped gradients induce F to be learned in the direction of minimizing the entropy, resulting in a skewed distribution of unlabeled data.

From the Equations 4.2 to 3.4, total objective function $\mathcal{L}_T$ for B randomly sampled training pairs of I^L and I^U, can be formulated as:

$$\mathcal{L}_T(\mathbf{B}) = \mathcal{L}_{N\text{-}PQ} + \frac{1}{B}\sum_{b=1}^{B}(\lambda_1\mathcal{L}_{cls} - \lambda_2\mathcal{L}_{SEM}) \tag{3.5}$$

where $\mathbf{B} = \{(I_1^L, y_1, I_1^U), ..., (I_B^L, y_B, I_B^U)\}$ and λ_1 and λ_2 are the hyper-parameters that balance the contribution of each loss function. I force training optimizer to minimize the $\mathcal{L}_T$, so that the $\mathcal{L}_{N-PQ}$ and $\mathcal{L}_{cls}$ is minimized while the $\mathcal{L}_{SEM}$ is maximized, simultaneously. In this way, F can learn the deep representation of both labeled and unlabeled data. However, to make the codewords robust against unlabeled data, it is necessary to reflect the unlabeled data signals directly into Z. Accordingly, I apply another soft assignment to embed sub-prototype intelligence of $\mathbf{W}_m$ into the of m-th codebook by updating codewords as $\mathbf{z}'_{mk} = s_m(\mathbf{z}_{mk}; \alpha, \mathbf{W}_m)$. As a result, high retrieval performance can be expected by exploiting the potential of unlabeled data for quantization.

3.2.2 Retrieval

Building Retrieval Database After learning the entire GPQ framework, I can build a retrieval database using images in $\mathcal{X}^U$. Given an input image $I^R \in \mathcal{X}^U$, I first extract $\hat{\mathbf{x}}^R$ from F. Then, I find the nearest codeword $\mathbf{z}_{mk^*}$ of each sub-vector $\mathbf{x}_m^R$ from the corresponding codebook $\mathbf{Z}_m$, by computing the cosine similarity. After that, formatting a index k^* of the nearest codeword as binary to generate a sub-binary code $\mathbf{b}^R$. Finally, concatenate all the sub-binary codes to obtain a $M \cdot \log_2(K)$-bits binary code $\hat{\mathbf{b}}^R$, where $\hat{\mathbf{b}}^R = [\mathbf{b}_1^R, ..., \mathbf{b}_M^R]$. This procedure is repeated for all images to store

them as binary, and Z is also stored for distance calculation.

Asymmetric Search For a given query image I^Q, $\hat{\mathbf{x}}^Q$ is extracted from F. To conduct the image retrieval, I take the m-th sub-vector $\mathbf{x}_m^Q$ as an example, compute the cosine similarity between $\mathbf{x}_m^Q$ and every codeword belonging to the m-th codebook, and store measured similarities on the look-up-table (LUT). Similarly, the same operation is done for the other sub-vectors, and the results are also stored on the LUT. The distance between the query image and a binary code in the database can be calculated asymmetrically, by loading the pre-computed distance from the LUT using a sub-binary codes, and aggregating all the loaded distances.

3.3 Experiments

GPQ is evaluated for two semi-supervised image retrieval protocols against several Hashing approaches. I conduct experiments on the two most popular image retrieval benchmark datasets. Extensive experimental results show that GPQ achieves superior performance to existing methods.

3.3.1 Setup

Evaluation Protocols and Metrics Following the semi-supervised retrieval experiments in [66, 62, 42], I adopt two protocols as follows.

- **Protocol 1: Single Category Image Retrieval** Assuming all categories (classes) used for image retrieval are known and only a small number of label data is provided for each class. The labeled data is used for training, and the unlabeled data is used for building the retrieval database and the query dataset. In this case, labeled data and the unlabeled data belonging to the retrieval database are used for semi-supervised learning.

- **Protocol 2: Unseen Category Image Retrieval** In line with semi-supervised learning, suppose that the information of the categories in the query dataset is unknown,

and consider building a retrieval database using both known and unknown categories. For this situation, I divide image dataset into four parts: train75, test75, train25, and test25, where train75 and test75 are the data of 75% categories, while train25 and test25 are the data of the remaining 25% categories. I use train75 for training, train25 and test75 for the retrieval database, and test25 for the query dataset. In this case, train75 with labels, and train25, test75 without labels are used for semi-supervised learning.

The retrieval performance of Hashing method is measured by mAP (mean Average Precision) with bit lengths of 12, 24, 32, and 48 for all images in the query dataset. In particular, I set Protocol 1 as the primary experiment and observe the contribution of each training objective.

Datasets I set up two benchmark datasets, different for each protocol as shown in the Table 3.1, and each dataset is configured as follows.

	CIFAR-10		NUS-WIDE	
	Protocol 1	Protocol 2	Protocol 1	Protocol 2
Query	1,000	9,000	2,100	35,272
Training	5,000	21,000	10,500	48,956
Retrieval Database	54,000	30,000	157,043	85,415

Table 3.1: Detailed composition of two benchmark datasets.

- **CIFAR-10** is a dataset containing 60,000 color images with the size of 32×32. Each image belongs to one of 10 categories, and each category includes 6,000 images.

- **NUS-WIDE** [**?**] is a dataset consisting nearly 270,000 color images with various resolutions. Images in the dataset associate with one or more class labels of 81 semantic concepts. I select the 21 most frequent concepts for experiments, where each concept has more than 5,000 images, with a total of 169,643.

Concept	Method	CIFAR-10				NUS-WIDE			
		12-bits	24-bits	32-bits	48-bits	12-bits	24-bits	32-bits	48-bits
Deep Semi-supervised	GPQ (Mine)	**0.858**	**0.869**	**0.878**	**0.883**	**0.852**	**0.865**	**0.876**	**0.878**
	SSGAH [42]	0.819	0.837	0.847	0.855	0.838	0.849	0.863	0.867
	BGDH [62]	0.805	0.824	0.826	0.833	0.810	0.821	0.825	0.829
	SSDH [66]	0.801	0.813	0.812	0.814	0.783	0.788	0.791	0.794
Deep Quantization	PQN [64]	0.795	0.819	0.823	0.830	0.803	0.818	0.822	0.824
	DTQ [49]	0.785	0.789	0.790	0.792	0.791	0.798	0.808	0.811
	DQN [37]	0.527	0.551	0.558	0.564	0.764	0.778	0.785	0.793
Deep Binary Hashing	SUBIC [43]	0.635	0.689	0.713	0.721	0.652	0.783	0.792	0.796
	NINH [67]	0.600	0.667	0.689	0.702	0.597	0.627	0.647	0.651
	CNNH [30]	0.496	0.580	0.582	0.583	0.536	0.522	0.533	0.531
Product Quantization	LOQP [47]	0.279	0.324	0.366	0.370	0.436	0.452	0.463	0.468
	OPQ [40]	0.265	0.315	0.323	0.345	0.429	0.433	0.450	0.458
	PQ [45]	0.237	0.265	0.268	0.266	0.398	0.406	0.413	0.422
Binary Hashing	SDH [22]	0.255	0.330	0.344	0.360	0.414	0.465	0.451	0.454
	ITQ [5]	0.158	0.163	0.168	0.169	0.428	0.430	0.432	0.435
	SH [28]	0.124	0.125	0.125	0.126	0.390	0.394	0.393	0.396

Table 3.2: mAP scores of different Hashing algorithms on experimental protocol 1.

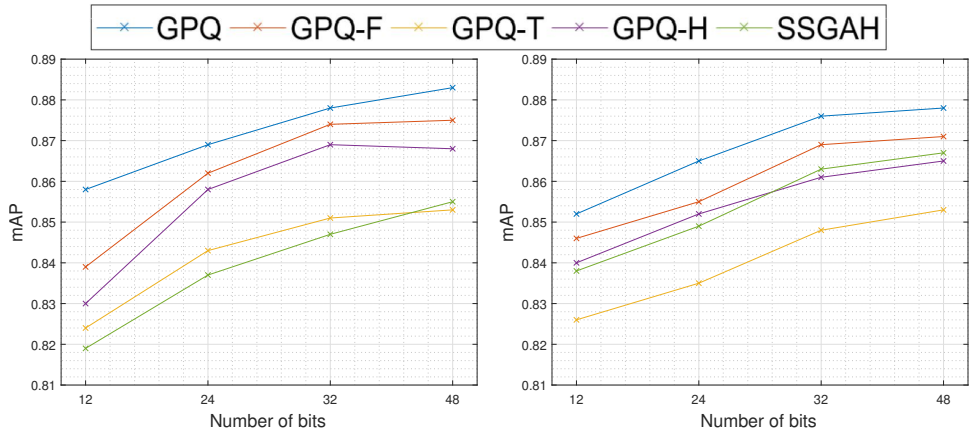
Concept	Method	CIFAR-10				NUS-WIDE			
		12-bits	24-bits	32-bits	48-bits	12-bits	24-bits	32-bits	48-bits
Deep Hashing	GPQ (Mine)	**0.321**	**0.333**	**0.350**	**0.358**	**0.554**	**0.565**	**0.578**	**0.586**
	SSGAH [42]	0.309	0.323	0.341	0.339	0.539	0.553	0.565	0.579
	SSDH [66]	0.285	0.291	0.311	0.325	0.510	0.533	0.549	0.551
	NINH [67]	0.241	0.249	0.253	0.272	0.484	0.483	0.485	0.487
	CNNH [30]	0.210	0.225	0.227	0.231	0.445	0.463	0.471	0.477
CNN features + non-Deep Hashing	SDH [22]	0.185	0.193	0.199	0.213	0.471	0.490	0.489	0.507
	ITQ [5]	0.157	0.165	0.189	0.201	0.488	0.493	0.508	0.503
	LOPQ [47]	0.134	0.127	0.126	0.124	0.416	0.386	0.380	0.379
	OPQ [40]	0.107	0.119	0.125	0.138	0.341	0.358	0.371	0.373

Table 3.3: mAP scores of different Hashing algorithms on experimental protocol 2.

3.3.2 Results and Analysis

Overview Experimental results for protocol 1 and protocol 2 are shown in Tables 3.2 and 3.3, respectively. In each Table, the methods are divided into several basic concepts and listed by group. I investigate the variants of GPQ for ablation study, and the results can be seen in Figures 3.3 to 3.5. From the results, I can observe that my GPQ scheme outperforms other Hashing methods, demonstrating that the proposed loss functions effectively improve the GPQ network by training it in a semi-supervised fashion.

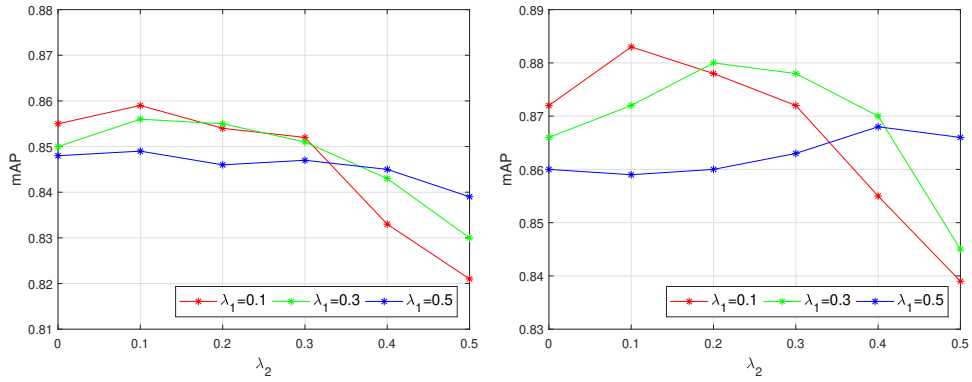
Comparison with Others As shown in Table 3.2, the proposed GPQ performs substantially better over all bit lengths than compared methods. Specifically, when I averaged mAP scores for all bit lengths, GPQ are 4.8%p and 4.6%p higher than the previous semi-supervised retrieval methods on CIFAR-10 and NUS-WIDE respectively. In particular, the performance gap is more pronounced as the number of bits decreases.



(a) CIFAR-10 (b) NUS-WIDE

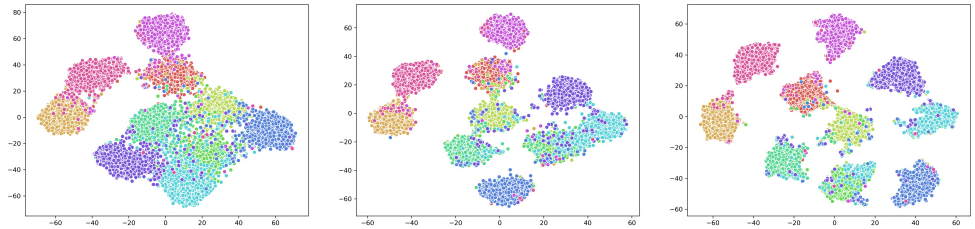
Figure 3.3: The comparison results of GPQ and its variants.

This tendency is intimately related to the baseline Hashing concepts. Comparing the results of the PQ-based and BH-based methods, I can identify that PQ-based ones are generally superior for both deep and non-deep cases especially for smaller bits. This is because unlike BH, PQ-based methods have the codewords of real values which enable mitigating the deviations generated during the encoding time, and they also allow more diverse distances through asymmetric calculation between database and query inputs. For the same reason, PQ-based GPQ with these advantages is able to achieve the state-of-the-art results in semi-supervised image retrieval.



(a) 12-bits (b) 48-bits

Figure 3.4: The sensitivity investigation of two balancing parameters: λ_1 and λ_2.



(a) GPQ-T (b) GPQ-H (c) GPQ

Figure 3.5: The t-SNE visualization of deep representations learned by GPQ-T, GPQ-H and GPQ on CIFAR-10 dataset respectively.

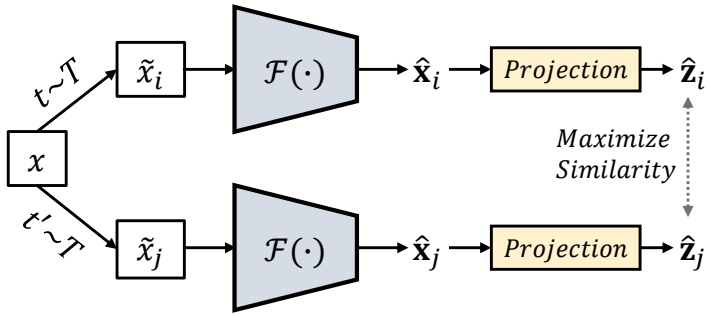
Chapter 4

Unsupervised Learning for Product Quantization: Self-supervised Product Quantization for Deep Unsupervised Image Retrieval

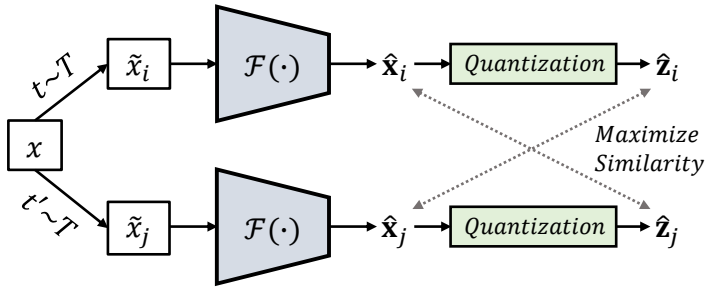
4.1 Motivation and Overview

Approximate Nearest Neighbor (ANN) search has received much attention in image retrieval research due to its low storage cost and fast search speed. There are two mainstream approaches in the ANN research community, where one is *Hashing* [27], and the other is *Vector Quantization* (VQ). Both methods aim to transform high-dimensional data into compact binary codes while preserving the semantic similarity, and the difference lies in measuring the distance between two different binary codes.

In the case of Hashing methods [91, 28, 75, 5, 78], the distance between two different binary codes is calculated using Hamming distance, *i.e.,* simple XOR operation. However, there exists a limitation that the distance between data points can be represented with only a few discrete values. To alleviate this problem, VQ based methods [45, 40, 47, 68, 88, 69, 89] have been proposed, which exploits quantized real-valued vectors in distance measurement. Among these, *Product Quantization* (PQ) [45] is the most prototypical method due to its fast retrieval speed and high accuracy.



(a) Contrastive Learning



(b) Cross Quantized Contrastive Learning (Mine)

Figure 4.1: A simple framework comparison between contrastive learning (a) and cross quantized contrastive learning (b). The separately sampled two transformations ($t, t' \sim \mathcal{T}$) are applied on an image x to generate two different views $\tilde{x}_i$ and $\tilde{x}_j$, and corresponding deep descriptor $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{x}}_j$ are obtained with the feature extractor $\mathcal{F}(\cdot)$, respectively. The feature representations in contrastive learning are achieved by comparing the similarity between the projection head outputs $\hat{\mathbf{z}}_i$ and $\hat{\mathbf{z}}_j$. Instead of projection, I introduce the quantization head, which collects codebooks of product quantization. By maximizing cross-similarity between the deep descriptor of one view and the product quantized descriptor of the other, both codewords and deep descriptors are jointly trained to contain discriminative image content representations.

The essence of PQ is to decompose the high-dimensional vector space of the representative feature vector (image descriptor) into the Cartesian product of several subspaces. To be specific, the image descriptor is divided into several subvectors according to the subspaces, and the subvectors are clustered to form centroids. *Codebook* of each subspace is configured with corresponding centroids; *codewords*, which are regarded as quantized representations of the images. The distance calculation in PQ is asymmetrically approximated by utilizing the real-valued codewords, resulting in

richer distance representations than the Hashing.

Recently, supervised deep image retrieval is the most popular approach to implementing image retrieval systems. Deep Hashing methods [30, 2, 44, 17, 87] show promising results, however, since binary codes for Hashing cannot be directly applied to learn deep continuous representations, there occurs an inevitable deviation. To address this problem, quantization-based deep image retrieval approaches have been proposed [37, 64, 49, 48, 76], showing superior performance. By introducing differentiable quantization methods on continuous deep image feature vectors (deep descriptors), the direct learning of deep representations is allowed in the real-valued space.

Despite the outstanding performance of the deep supervised learning-based image retrieval schemes, the problem remains that expensive label information is indispensable. In this respect, deep unsupervised Hashing methods have been proposed [77, 73, 84, 90, 85, 74, 81, 86, 82], which investigate the image similarity to discover semantically distinguishable binary codes without annotations. However, while quantization-based methods have advantages over Hashing-based ones in deep image retrieval, only limited studies exist that adopt quantization for deep unsupervised retrieval [80], which employs pre-extracted visual descriptors instead of images.

In this dissertation, I propose the first *unsupervised* end-to-end deep quantization-based image retrieval method; *Self-supervised Product Quantization* (SPQ) network, which jointly learns the feature extractor and the codewords. As shown in Fig 4.1, the main idea of SPQ is based on self-supervised learning. Similar to a concept of contrastive learning [4, 15, 70], I regard that two different "views" (individually transformed outputs) of a single image are correlated, and conversely, the views generated from different images are uncorrelated. Especially to train PQ codewords, I introduce *Cross Quantized Contrastive learning*, which maximizes the cross-similarity between the correlated deep descriptor and the product quantized descriptor. This strategy leads both deep descriptors and PQ codewords to become discriminative, allowing the SPQ

framework to achieve high retrieval accuracy.

To demonstrate the efficiency of my proposal, I conduct experiments under various training conditions. Specifically, unlike previous methods that utilize pretrained model weights learned from a large labeled dataset, I carry out experiments with "truly"-unsupervised settings where human supervision is excluded. Despite the absence of label information, SPQ achieves the state-of-the-art performance.

4.1.1 Related Works

This section categorizes image retrieval algorithms regarding whether or not deep learning is utilized (conventional versus deep methods) and briefly explains the approaches. For a more comprehensive understanding, refer to a survey paper [27].

Conventional methods. One of the most common strategies for fast image retrieval is hashing. For some examples, Locality Sensitivity Hashing (LSH) [91] employed random linear projections to hash. Spectral Hashing (SH) [28] and Discrete Graph Hashing (DGH) [78] exploited graph-based approaches to preserve data similarity of the original feature space. K-means Hashing (KMH) [75] and Iterative Quantization (ITQ) [5] focused on minimizing quantization errors that occur when mapping the original feature to discrete binary codes. Another fast image retrieval strategy is vector quantization. There are Product Quantization (PQ) [45] and its improved variants; Optimized PQ (OPQ) [40], Locally Optimized PQ (LOPQ) [47], and methods with different quantizers, such as Additive [68], Composite [88], Tree [69], and Sparse Composite Quantizers [89]. My SPQ belongs to the PQ family, where the deep feature data space is divided into several disjoint subspaces. The divided deep subvectors are then trained with the proposed loss function to find the optimal codewords.

Deep methods. Supervised deep convolutional neural network (CNN)-based hashing approaches [30, 2, 44, 17, 87] have shown superior performance in many image retrieval tasks. There are also quantization-based deep image retrieval methods [37, 48],

which use pretrained CNNs and fine-tune the network to train robust codewords together. For improvement, the metric learning schemes are applied in [64, 49, 76] to learn codewords and deep representations together with the pairwise semantic similarity. Note that I also utilize a type of metric learning, *i.e.,* contrastive learning; however, my method requires no label information in learning the codewords. Regarding unsupervised deep image retrieval, most works are based on hashing. To be specific, generative mechanisms are utilized in [73, 84, 90, 74], and graph-based techniques are employed in [81, 82]. Notably, DeepBit [77] has a similar concept to SPQ in that the distance between the transformed image and the original one is minimized. However, the hash code representation has a limitation in that only a simple rotational transformation is exploited. In terms of deep quantization, there only exists a study dubbed Unsupervised Neural Quantization (UNQ) [80], which uses pre-extracted visual descriptors instead of employing the image itself to find the codewords.

To improve the quality of image descriptors and codewords for unsupervised deep PQ-based retrieval, I configure SPQ with a feature extractor to explore the entire image information. Then, I jointly learn every component of SPQ in a self-supervised fashion. Similar to [4, 70], the full knowledge of the dataset is augmented with several transformations such as crop and resize, flip, color distortion, and Gaussian blurring. By cross-contrasting differently augmented images, both image descriptors and codewords become discriminative to achieve a high retrieval score.

4.2 Self-supervised Product Quantization

4.2.1 Overall Framework

The goal of an image retrieval model is to learn a mapping $\mathcal{R} : x \rightarrow \mathbf{b}$ where $\mathcal{R}$ denotes the overall system, x is an image included in a dataset $\mathcal{X} = \{x_n\}_{n=1}^{N}$ of N training samples, and $\hat{\mathbf{b}}$ is a B-bits binary code $\hat{\mathbf{b}} \in \{0, 1\}^{\mathrm{B}}$. As illustrated in Figure 4.2,

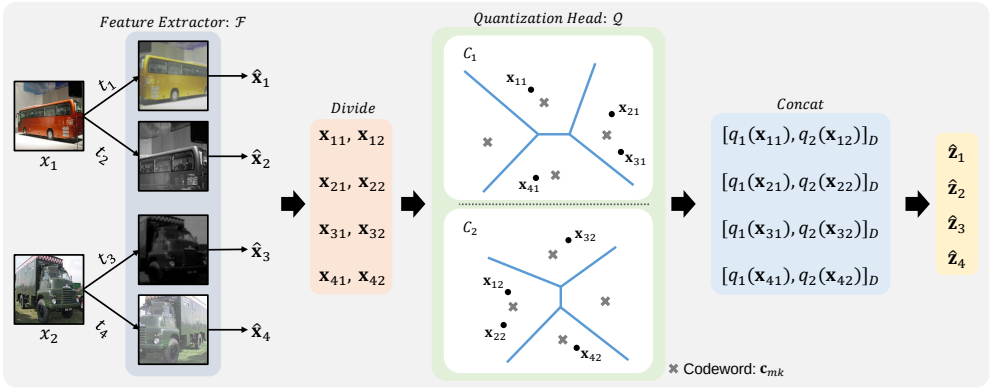


Figure 4.2: An illustration of feature extraction and quantization procedure in SPQ. Randomly sampled data augmentation techniques ($t_n \sim T$) are applied on x_1 and x_2 to produce transformed images (different views). There are two trainable components; 1) CNN-based feature extractor $\mathcal{F}$, and 2) quantization head $\mathcal{Q}$, which collects multiple codebooks to conduct product quantization. For example, I set up two codebooks C_1 and C_2, and illustrate 2D conceptual Voronoi diagram in $\mathcal{Q}$. The original feature space of deep descriptor (feature vector $\hat{\mathbf{x}}_n \in R^D$) is divided into two subspaces and generates subvectors; $\mathbf{x}_{nk}$ where $k = \{1, 2\}$ and $\mathbf{x}_{nk} \in R^{D/2}$. By employing soft quantizer $q_k(\cdot)$ on each $\mathbf{x}_{nk}$, the sub-quantized descriptor $\mathbf{z}_{nk} = q_k(\mathbf{x}_{nk})$ is approximated with the combination of the codewords. Notably, subvectors representing similar features are allocated to the same codeword. The output product quantized descriptor $\hat{\mathbf{z}}_n \in R^D$ is obtained by $[\cdot]_D$ operation which concatenates the sub-quantized descriptors along the D-dimension.

$\mathcal{R}$ of Self-supervised Product Quantization (SPQ) contains deep convolutional neural network based feature extractor $\mathcal{F}(x; \theta_{\mathcal{F}})$ which outputs a compact deep descriptor (feature vector) $\mathbf{x} \in R^D$. Any CNN architecture can be exploited as a feature extractor, as long as it can handle length-scalable fully connected layer, e.g. AlexNet [14], VGG [83], or ResNet [6]. I configure the baseline network architecture with ResNet50 that generally shows outstanding performance in image representation learning.

In terms of applying quantization for fast image retrieval, $\mathcal{R}$ employs K codebooks in the quantization head $\mathcal{Q}(\hat{\mathbf{x}}; \theta_{\mathcal{Q}})$ of $\{C_1, ..., C_K\} \subset \mathcal{Q}$, where C_i consists of M codewords $\mathbf{c} \in R^{D/K}$ as $C_k = \{\mathbf{c}_{1k}, ..., \mathbf{c}_{Mk}\}$. Product Quantization (PQ) is conducted in $\mathcal{Q}$ by dividing the deep feature space into the Cartesian product of multiple subspaces. Every codebook of corresponding subspace induces to exhibit several

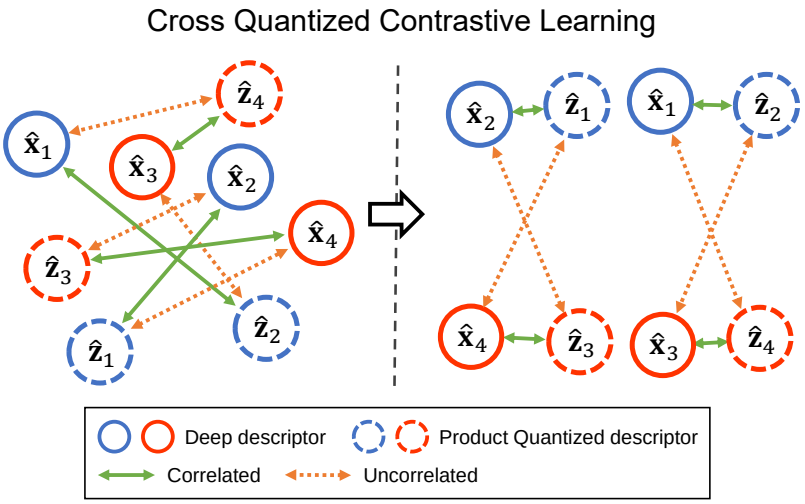


Figure 4.3: A visualized example of proposed cross quantized contrastive learning strategy. For simplicity, I take the examples $x_1 \mapsto \{\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2\}$, and $x_2 \mapsto \{\hat{\mathbf{x}}_3, \hat{\mathbf{x}}_4, \hat{\mathbf{z}}_3, \hat{\mathbf{z}}_4\}$ from Figure 2.2, and paint the feature representations related to x_1 in blue, and x_2 in red. Taking into account the cross-similarity between $\hat{\mathbf{x}}$ and $\hat{\mathbf{z}}$ as: $\hat{\mathbf{x}}_1 \leftrightarrow \{\hat{\mathbf{z}}_2, \hat{\mathbf{z}}_4\}$, $\hat{\mathbf{x}}_2 \leftrightarrow \{\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_3\}$, $\hat{\mathbf{x}}_3 \leftrightarrow \{\hat{\mathbf{z}}_2, \hat{\mathbf{z}}_4\}$,and $\hat{\mathbf{x}}_4 \leftrightarrow \{\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_3\}$, the network is trained to understand the discriminative image contents, while simultaneously collecting frequently occurring local patterns into the codewords.

distinctive characteristics representing the image dataset $\mathcal{X}$. Each codeword belonging to the codebook infers a clustered centroid of a divided deep descriptor, which aims to hold a local pattern that frequently occurs. During quantization, similar properties between images are shared by being assigned to the same codeword, whereas distinguishable features have different codewords. As a result, various distance representations for efficient image retrieval are achieved.

4.2.2 Self-supervised Training

First of all, to conduct deep learning with $\mathcal{F}$ and $\mathcal{Q}$ in an end-to-end manner, and make the whole codewords contribute to training, I need to solve the infeasible derivative calculation of hard assignment quantization. Therefore, following [64], I introduce soft quantization on the quantization head with the soft quantizer $q_k(\cdot)$ as:

$$\mathbf{z}_{nk} = \sum_{m}^{M} \frac{\exp(-\|\mathbf{x}_{nk} - \mathbf{c}_{mk}\|_2^2/\tau_q)}{\sum_{m'}^{M} \exp(-\|\mathbf{x}_{nk} - \mathbf{c}_{m'k}\|_2^2/\tau_q)} \mathbf{c}_{mk} \tag{4.1}$$

where τ_q is a non-negative temperature parameter which scales the input of the softmax, and $\|\cdot\|_2^2$ denotes the squared Euclidean distance to measure the similarity between inputs. In this fashion, the sub-quantized descriptor $\mathbf{z}_{nk} = q_k(\mathbf{x}_n; \tau_q, C_k)$ can be regarded as an exponential weighted sum of the codewords that belongs to the C_k. Note that the entire codewords in the codebook are utilized to approximate the quantized output, where the closest codeword contributes the most.

Besides, unlike previous deep PQ approaches [64, 76], I exclude intra-normalization which is known to minimize the impact of burst visual features when concatenating sub-quantized descriptors to obtain the full product quantized descriptor $\hat{\mathbf{z}}$. Since my SPQ is trained without any human supervision which assists to find distinct features, I focus on catching dominant visual features rather than balancing the influence of every codebook.

To learn the deep descriptors and the codewords together, I propose *cross quantized contrastive learning* scheme, as observed in the Figure 4.1(b). Inspired from the contrastive learning [4, 70], I attempt to comparing the deep descriptors and the product quantized descriptors of various views (transformed images). As shown in Figure 4.3, the deep descriptor and the product quantized descriptor are treated as correlated if the views are originated from the same image, whereas uncorrelated if the views are originated from the different images. Note that, to increase the generalization capacity of the codewords, the correlation between the deep descriptor and the quantized descriptor of itself ($\hat{\mathbf{x}}_n$ and $\hat{\mathbf{z}}_n$) is ignored. This is because the contribution of other codewords decreases when the agreement between the subvector and the nearest codeword is maximized.

For a given mini-batch of size N_B, we randomly sample N_B examples from the

database $\mathcal{X}$ and apply a random combination of augmentation techniques to each image twice to generate $2N_B$ data points (views). Inspired from [71, 4, 70], we take into account that two separate views of the same image $(\tilde{x}_i, \tilde{x}_j)$ are correlated, and the other $2(N_B - 1)$ views originating from different images within a mini-batch are uncorrelated. On this assumption, we design a cross quantized contrastive loss function to learn the correlated pair of examples (i, j) as:

$$\ell_{(i,j)} = -\log \frac{\exp\left(\mathcal{S}(i,j)/\tau_{cqc}\right)}{\sum_{n=1}^{N_B} \mathbb{1}_{[n' \neq j]} \exp\left(\mathcal{S}(i,n')/\tau_{cqc}\right)} \tag{4.2}$$

where $n' = \begin{cases} 2n-1 & \text{if } j \text{ is odd} \\ 2n & \text{else} \end{cases}$, $\mathcal{S}(i,j)$ denotes a cosine similarity between $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{z}}_j$, τ_{cqc} is a non-negative temperature parameter, and $\mathbb{1}_{[n' \neq j]} \in \{0, 1\}$ is an indicator that evaluates to 1 iff $n' \neq j$. Notably, to reduce redundancy between $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{z}}_i$ which are similar to each other, the loss is computed for the half of the uncorrelated samples in the batch. The cosine similarity is used as a distance measure to avoid the norm deviations between $\hat{\mathbf{x}}$ and $\hat{\mathbf{z}}$.

Concerning data augmentation for generating various views, I employ five popular techniques as: 1) resized crop to treat local, global, and adjacent views, 2) horizontal flip to handle mirrored inputs, 3) color jitter to deal with color distortions, 4) grayscale to focus more on intensity, and 5) Gaussian blur to cope with noise in image. Default setup is directly taken from [4], where all transformations are randomly applied in a sequential manner (1-5). Exceptionally, I modify color jitter strength as 0.5 to fit in SPQ, following the empirical observation. In the end, SPQ is capable of interpreting contents in the image by contrasting different views in a self-supervised way.

4.3 Experiments

4.3.1 Datasets

To evaluate the performance of SPQ, I conduct comprehensive experiments on three public benchmark datasets, following experimental protocols in recent unsupervised deep image retrieval methods [90, 86, 82].

CIFAR-10 contains 60,000 images with the size of 32×32 in 10 class labels, and each class has 6,000 images. I select 5,000 images per class as a training set, 100 images per class as a query set. The entire training set of 50,000 images are utilized to build a retrieval database.

FLICKR25K consists 25,000 images with various resolutions collected from the Flickr website. Every image is manually annotated with at least one of the 24 semantic labels. I randomly take 2,000 images as a query set and employ the remaining 23,000 images to build a retrieval database, of which 5,000 images are utilized for training.

NUS-WIDE has nearly 270,000 images with various resolutions in 81 unique labels, where each image belongs to one or more labels. I pick out images containing the 21 most frequent categories to perform experiments with a total of 169,643. I randomly choose a total of 10,500 images as a training set with each category being at least 500, a total of 2,100 images as a query set with each category being at least 100, and the rest images as a retrieval database.

Table 4.1: Detailed composition of three benchmark datasets.

Dataset	# Train	# Query	# Retrieval	# Class
CIFAR-10	50,000	10,000	50,000	10
FLICKR25K	5,000	2,000	23,000	24
NUS-WIDE	10,500	2,100	157,043	21

Table 4.2: mAP scores of different retrieval methods on three benchmark datasets.

Method	CIFAR-10			FLICKR25K			NUS-WIDE		
	16-bits	32-bits	64-bits	16-bits	32-bits	64-bits	16-bits	32-bits	64-bits
Shallow Methods without Deep Learning									
LSH [91]	0.132	0.158	0.167	0.583	0.589	0.593	0.432	0.441	0.443
SH [28]	0.272	0.285	0.300	0.591	0.592	0.602	0.510	0.512	0.518
ITQ [5]	0.305	0.325	0.349	0.610	0.622	0.624	0.627	0.645	0.664
PQ [45]	0.237	0.259	0.272	0.601	0.612	0.626	0.452	0.464	0.479
OPQ [40]	0.297	0.314	0.323	0.620	0.626	0.629	0.565	0.579	0.598
LOPQ [47]	0.314	0.320	0.355	0.614	0.634	0.635	0.620	0.655	0.670
Deep Semi-unsupervised Methods									
DeepBit [77]	0.220	0.249	0.277	0.593	0.593	0.620	0.454	0.463	0.477
GreedyHash [85]	0.448	0.473	0.501	0.689	0.699	0.701	0.633	0.691	0.731
DVB [81]	0.403	0.422	0.446	0.614	0.655	0.658	0.677	0.632	0.665
DistillHash [86]	0.454	0.469	0.489	0.696	0.706	0.708	0.667	0.675	0.677
TBH [82]	0.532	0.573	0.578	0.702	0.714	0.720	0.717	0.725	0.735
Deep Truly-unsupervised Methods									
SGH [73]	0.435	0.437	0.433	0.616	0.628	0.625	0.593	0.590	0.607
HashGAN [74]	0.447	0.463	0.481	-	-	-	-	-	-
BinGAN [90]	0.476	0.512	0.520	0.663	0.679	0.688	0.654	0.709	0.713
BGAN [84]	0.525	0.531	0.562	0.671	0.686	0.695	0.684	0.714	0.730
SPQ (Mine)	**0.768**	**0.793**	**0.812**	**0.764**	**0.789**	**0.801**	**0.759**	**0.781**	**0.792**

4.3.2 Experimental Settings

Evaluation Metrics. I employ mean Average Precision (mAP) to evaluate the retrieval performance. Specifically, in the case of multi-label image retrieval on FLICKR25K and NUS-WIDE dataset, it is considered relevant even if only one of the labels matches. I vary the number of bits allocated to the binary code as $\{16, 32, 64\}$ to measure the mAP scores of the retrieval approaches, mAP@1,000 for CIFAR-10 dataset and mAP@5,000 for FLICKR25K and NUS-WIDE datasets following [82, 86]. In addition, by employing 64-bits hash codes of different algorithms, I draw Precision-Recall curves (PR) to compare the precisions at different recall levels and report Precision curves with respect to 1,000 top returned samples (P@1,000) to contrast the ratio of

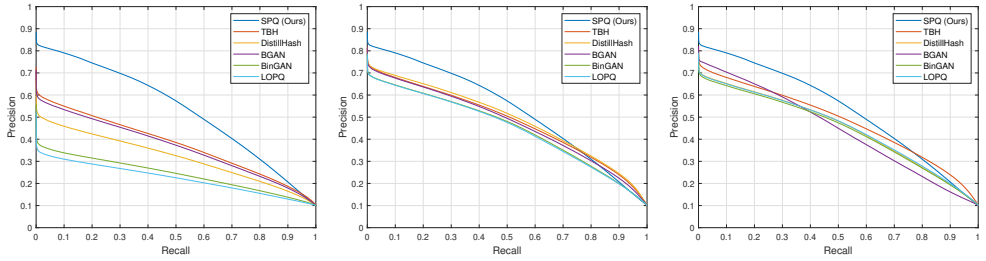
results retrieved correctly.

Implementation details. There are three baseline approaches that I categorize to make comparison: 1) shallow methods without deep learning, based on Hashing for LSH [91], SH [28], ITQ [5], and based on product quantization for PQ [45], OPQ [40] LOPQ [47], 2) deep semi-unsupervised methods for DeepBit [77], GreedyHash [85], DVB [81], DistillHash [86], TBH [82], and 3) deep truly-unsupervised methods for SGH [73], HashGAN [74], BinGAN [90], and BGAN [84]. The terms "semi" and "truly" indicate whether the pretrained model weights are utilized or not. Both semi and truly training conditions can be applied to SPQ; however, I take the truly-unsupervised model that has the advantage of not requiring human supervision as the baseline.

To evaluate the shallow and deep semi-unsupervised methods, I employ ImageNet pretrained model weights of AlexNet [14] or VGG16 [83] to utilize fc_7 features, following the experimental settings of [86, 81, 82]. Since those models take only fixed-size inputs, I need to resize all images to 224×224, by upscaling the small images and downsampling the large ones. In case of evaluating deep truly-unsupervised methods including SPQ, the same resized images of FLICKR25K and NUS-WIDE datasets are used for simplicity, and the original resolution images of CIFAR-10 are used to reduce computational load.

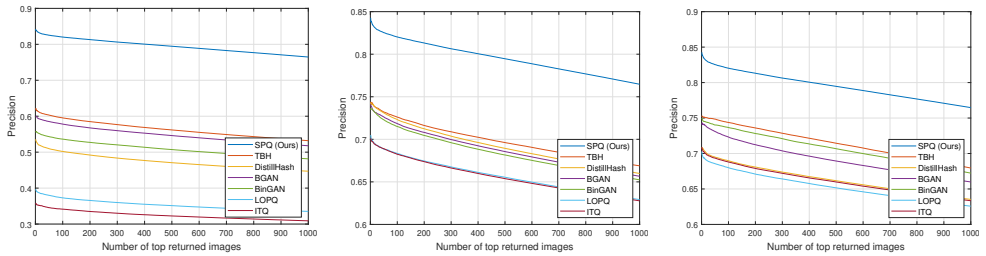
My implementation of SPQ is based on PyTorch with NVIDIA Tesla V100 32GB Tensor Core GPU. Following the observations in recent self-supervised learning studies [4, 70], I set the baseline network architecture as a standard ResNet50 [6] for FLICKR25K and NUS-WIDE datasets. In case of CIFAR-10 dataset with the much smaller images, I set the baseline as a standard ResNet18 [6], and modify the number of filters as same as ResNet50.

For network training, I adopt Adam [13] and decay the learning rate with the cosine scheduling without restarts [79], and set the batch size N_B as 256. I fix the dimension



(a) CIFAR-10 (b) FLICKR25K (c) NUS-WIDE

Figure 4.4: Precision-Recall curves on benchmarks with binary codes @ 64-bits.



(a) CIFAR-10 (b) FLICKR25K (c) NUS-WIDE

Figure 4.5: Precision@top-1000 curves on benchmarks with binary codes @ 64-bits.

of the subvector x and the codeword c to $D/K = 16$, and also the number of codewords to $M = 2^4$. Consequently, the number codebooks K is changed to $\{4, 8, 16\}$ because $K \cdot \log_2(M)$-bits are needed to obtain $\{16, 32, 64\}$-bits binary code. The temperature parameter τ_q and τ_{cqc} are set as 5 and 0.5, respectively. Data augmentation is operated with Kornia [72] library, and each transformation is applied with the same probability as the settings in [4]. I will make my code publicly open for further research and comparison.

4.3.3 Results

The mAP results on three different image retrieval datasets are listed in Table 4.2, showing that SPQ substantially outperforms all the compared methods in every bit-length. Additionally, referring to Figures 4.4 and 4.5, SPQ is demonstrated to be the most desirable retrieval system.

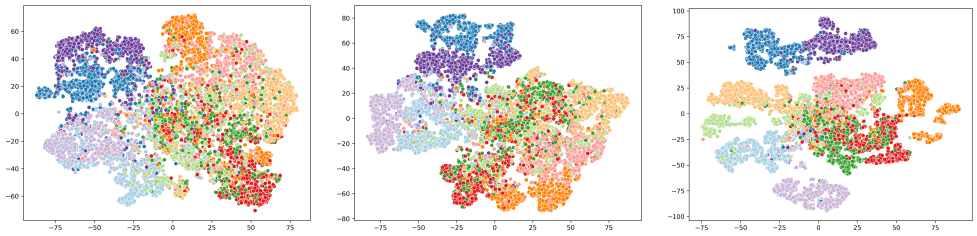
First of all, compared to the best shallow method LOPQ [47], SPQ reveals a performance improvement of more than 46%p 16%p and 13%p in the average mAP on CIFAR-10, FLICKR25K and NUS-WIDE, respectively. The reason for the more pronounced difference for CIFAR-10 is because the shallow methods involve an unnecessary upscaling process to utilize the ImageNet pretrained deep features. SPQ has an advantage over shallow methods in that various and suitable neural architectures can be accommodated for feature extraction and end-to-end learning.

Second, in contrast to the best deep semi-unsupervised method TBH [82], SPQ yields 23%p 7.2%p 5.2%p higher average mAP scores on CIFAR-10, FLICKR25K and NUS-WIDE, respectively. Even if there is no prior information, SPQ distinguishes contents within the image properly for retrieval, by comparing multiple views of training samples.

Lastly, even with the truly unsupervised setup, SPQ achieves state-of-the-art retrieval accuracy. Specifically, unlike previous Hashing-based truly unsupervised methods, SPQ introduces differentiable product quantization to the unsupervised image retrieval system for the first time. By considering cross-similarity between different views in a self-supervised way, deep descriptors and codewords are allowed to be discriminative.

4.3.4 Empirical Analysis

Ablation Study. I configure five variants of SPQ to investigate as: 1) *SPQ-C* replaces cross-quantized contrastive learning with a contrastive learning by comparing $\hat{\mathbf{z}}_i$ and $\hat{\mathbf{z}}_j$, 2) *SPQ-H* employs hard quantization instead of soft quantization, 3) *SPQ-Q*



(a) BinGAN [90] (b) TBH [82] (c) SPQ (Mine)

Figure 4.6: t-SNE visualization of deep representations learned by BinGAN, TBH, and SPQ on CIFAR-10 query set respectively.

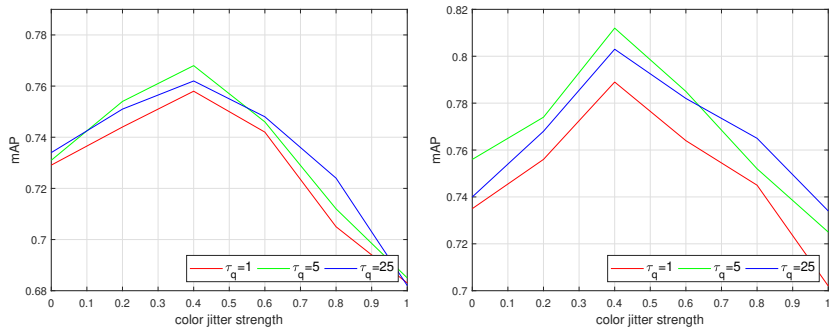
Table 4.3: mAP scores of SPQ and its variants on three benchmark datasets @ 32-bits.

Method	CIFAR-10	FLICKR25K	NUS-WIDE
SPQ-C	0.722	0.748	0.743
SPQ-H	0.735	0.754	0.746
SPQ-Q	0.704	0.733	0.735
SPQ-S	0.814	0.817	0.809
SPQ-V	0.776	0.779	0.765
SPQ	**0.793**	**0.789**	**0.781**

uses standard vector quantization, which does not divide the feature space and directly utilize entire feature vector to build the codebook, 4) *SPQ-S* exploits pretrained model weights to conduct deep semi-unsupervised image retrieval, and 5)*SPQ-V* utilizes VGG16 network architecture as the baseline.

As reported in Table 4.3, I can observe that each component of SPQ contributes sufficiently to performance improvement. Through the comparison with SPQ-C, it is confirmed that considering cross-similarity between the deep descriptor and the product quantized descriptor of different views, rather than comparing quantized outputs, resulting in more efficient image retrieval results. From the results of SPQ-H, I find that soft quantization is more suitable for learning codewords. The retrieval outcomes with SPQ-Q, which shows the biggest performance gap with SPQ, explain that product quantization leads to accomplishing precise search results by increasing the amount of distance representation.

Notably, SPQ-S, which utilizes ImageNet pretrained model weights for network



(a) @16-bits (b) @64-bits

Figure 4.7: Sensitivity investigation of two temperature hyper-parameters: τ_q and τ_{cqc} on CIFAR-10 dataset.

initialization, outperforms truly-unsupervised SPQ. In this observation, I can see that although SPQ demonstrates the best retrieval accuracy without any human guidance, better results can be obtained with some label information. Although SPQ-V is inferior to ResNet-based SPQ, its performance still surpasses existing state-of-the-art retrieval algorithms, which proves the excellence of PQ-based self-supervised learning scheme.

Hyper-parameter sensitivity. I explore the performance difference according to the changes of the hyper-parameters τ_q and τ_{cqc} in Figure 4.7. In general, since the larger τ_q is closer to hard quantization, the influence from the changes in color jitter strength is reduced. However, to cope with the color distortion properly, it is more helpful to learn codewords with soft quantization to some extent.

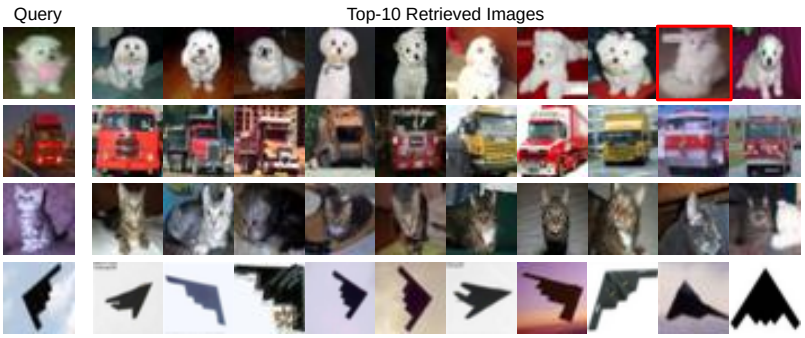


Figure 4.8: SPQ retrieval results on CIFAR-10 @ 32-bits.

Visualization. As illustrated in Figure 4.6, I employ t-SNE [50] to examine the distribution of deep representations of BinGAN, TBH, and my SPQ, where BinGAN and SPQ are trained under the truly-unsupervised setting. Nonetheless, my SPQ scatters data samples most distinctly where each color denotes a different class label. Furthermore, I show the actual returned images in Figure 4.8. Interestingly, not only images of the same category, but also images with visually similar contents are retrieved, like cat appears in the dog retrieval results.

Chapter 5

Conclusion

In this dissertation, deep learning based Hashing and Product Quantization (PQ) algorithms are proposed to achieve fast and accurate image retrieval. For semantically similar image retrieval, deep Hashing methods are proposed that retrieve images of the same category at the top. For more complex semantically and visually similar image retrieval, I consider semi-supervised learning to exploit both labeled and unlabeled in training deep PQ based image retrieval system. To be more focused on visually similar image retrieval, unsupervised learning approach with self-supervision is proposed to train the deep PQ model without human supervision.

Similarity Guided Hashing The proposed Similarity Guided Hashing (SGH) network for face image retrieval exploits an end-to-end supervised learning strategy. With the randomly transformed face images, the self and pairwise-similarity between the original image and the transformed one in the latent space are learned to find better image representations. In addition, I employed quantization and classification training objectives on hashing head to appropriately encode learned representations into the hash space while minimizing the information loss. Retrieval results on the large scale face image datasets with various resolutions verify the effectiveness of proposed approach with the state-of-the-art performances.

Deep Hash Distillation The proposed Self-distilled Hashing (SdH) scheme for deep hashing learning is demonstrated to generate robust hash codes from transformations. By maximizing the cosine similarity between hash codes of different views of one image, SdH minimizes the discrepancy in the representation due to augmentation and exploits its power in training. Besides, I optimized deep hashing model with proxy-based hash similarity learning and quantization loss of maximum likelihood manner. With all these proposals, I configured Deep Hash Distillation (DHD) framework that yields discriminative hash codes for hashing-based image retrieval systems. Experimental results on popular benchmarks validate the effectiveness of proposed approach with the state-of-the-art performance.

Generalized Product Quantization The first quantization based deep semi-supervised image retrieval technique, named Generalized Product Quantization (GPQ) network is proposed. I employed a metric learning strategy that preserves semantic similarity within the labeled data for the discriminative codebook learning. Further, I compute an entropy for each subspace and simultaneously maximize and minimize it to embed underlying information of the unlabeled data for the codebook regularization. Comprehensive experimental results justify that the GPQ yields state-of-the-art performance on large-scale image retrieval benchmark datasets.

Self-supervised Product Quantization A novel deep self-supervised learning-based fast image retrieval method, Self-supervised Product Quantization (SPQ) network is proposed. By employing a PQ scheme, I built the first end-to-end unsupervised learning framework for image retrieval. I introduced a cross quantized contrastive learning strategy to learn the deep representations and codewords to discriminate the image contents while clustering local patterns at the same time. Despite the absence of any supervised label information, SPQ yields state-of-the-art retrieval results.

Bibliography

[1] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *IEEE International Conference on Automatic Face & Gesture Recognition*, pages 67–74. IEEE, 2018.

[2] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S Yu. Hashnet: Deep learning to hash by continuation. In *CVPR*, pages 5608–5617, 2017.

[3] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014.

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[5] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *PAMI*, 35(12):2916–2929, 2012.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[7] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015.

[8] Young Kyun Jang and Nam Ik Cho. Generalized product quantization network for semi-supervised image retrieval. In *CVPR*, pages 3420–3429, 2020.

[9] Young Kyun Jang, Dong-ju Jeong, Seok Hee Lee, and Nam Ik Cho. Deep clustering and block Hashing network for face image retrieval. In *ACCV*, pages 325–339. Springer, 2018.

[10] Dong-ju Jeong, Sung-Kwon Choo, Wonkyo Seo, and Nam Ik Cho. Classification-based supervised Hashing with complementary networks for image search. In *BMVC*, page 74, 2018.

[11] L. Jing and Y. Tian. Self-supervised visual feature learning with deep neural networks: A survey. *PAMI*, pages 1–1, 2020.

[12] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*, 2020.

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2015.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, pages 1097–1105, 2012.

[15] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *ICML*, volume 119, 2020.

[16] Jian Li, Yabiao Wang, Changan Wang, Ying Tai, Jianjun Qian, Jian Yang, Chengjie Wang, Jilin Li, and Feiyue Huang. Dsfd: dual shot face detector. In *CVPR*, pages 5060–5069, 2019.

[17] Qi Li, Zhenan Sun, Ran He, and Tieniu Tan. Deep supervised discrete Hashing. In *NeurIPS*, pages 2482–2491, 2017.

[18] Jie Lin, Zechao Li, and Jinhui Tang. Discriminative deep Hashing for scalable face image retrieval. In *IJCAI*.

[19] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. Deep supervised Hashing for fast image retrieval. In *CVPR*, pages 2064–2072, 2016.

[20] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. Supervised Hashing with kernels. In *CVPR*, pages 2074–2081. IEEE, 2012.

[21] Hong-Wei Ng and Stefan Winkler. A data-driven approach to cleaning large face datasets. In *ICIP*, pages 343–347. IEEE, 2014.

[22] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. Supervised discrete Hashing. In *CVPR*, pages 37–45, 2015.

[23] Yuming Shen, Jie Qin, Jiaxin Chen, Mengyang Yu, Li Liu, Fan Zhu, Fumin Shen, and Ling Shao. Auto-encoding twin-bottleneck Hashing. In *CVPR*, pages 2818–2827, 2020.

[24] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *NeurIPS*, pages 1857–1865, 2016.

[25] Jinhui Tang, Zechao Li, and Xiang Zhu. Supervised deep Hashing for scalable face image retrieval. *PR*, 75:25–32, 2018.

[26] Jinhui Tang, Jie Lin, Zechao Li, and Jian Yang. Discriminative deep quantization Hashing for face image retrieval. *TNNLS*, 29(12):6154–6162, 2018.

[27] Jingdong Wang, Ting Zhang, Nicu Sebe, Heng Tao Shen, et al. A survey on learning to hash. *PAMI*, 40(4):769–790, 2017.

[28] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral Hashing. In *NeurIPS*, pages 1753–1760, 2009.

[29] Lior Wolf, Tal Hassner, and Itay Maoz. Face recognition in unconstrained videos with matched background similarity. In *CVPR*, pages 529–534. IEEE, 2011.

[30] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan. Supervised Hashing for image retrieval via image representation learning. In *AAAI*, 2014.

[31] Zhi Xiong, Dayan Wu, Wen Gu, Haisu Zhang, Bo Li, and Weiping Wang. Deep discrete attention guided Hashing for face image retrieval. In *ICMR*, pages 136–144, 2020.

[32] Li Yuan, Tao Wang, Xiaopeng Zhang, Francis EH Tay, Zequn Jie, Wei Liu, and Jiashi Feng. Central similarity quantization for efficient image and video retrieval. In *CVPR*, pages 3083–3092, 2020.

[33] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S4l: Self-supervised semi-supervised learning. In *ICCV*, pages 1476–1485, 2019.

[34] Han Zhu, Mingsheng Long, Jianmin Wang, and Yue Cao. Deep Hashing network for efficient similarity retrieval. In *AAAI*, 2016.

[35] Jegou, Herve and Douze, Matthijs and Schmid, Cordelia. , "Product quantization for nearest neighbor search," *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, vol. 33, no.1, pp. 117-128, 2010.

[36] Relja Arandjelovic and Andrew Zisserman. All about vlad. In *CVPR*, pages 1578–1585, 2013.

[37] Yue Cao, Mingsheng Long, Jianmin Wang, Han Zhu, and Qingfu Wen. Deep quantization network for efficient image retrieval. In *AAAI*, 2016.

[38] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014.

[39] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. *ICML*, 2015.

[40] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. Optimized product quantization for approximate nearest neighbor search. In *CVPR*, pages 2946–2953, 2013.

[41] Jae-Pil Heo, Zhe Lin, and Sung-Eui Yoon. Distance encoded product quantization for approximate k-nearest neighbor search in high-dimensional space. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[42] Qinghao Hu, Jian Cheng, Zengguang Hou, et al. Semi-supervised generative adversarial Hashing for image retrieval. In *ECCV*, pages 491–507. Springer, 2018.

[43] Himalaya Jain, Joaquin Zepeda, Patrick Pérez, and Rémi Gribonval. Subic: A supervised, structured binary code for image search. In *ICCV*, pages 833–842, 2017.

[44] Young Kyun Jang, Dong-ju Jeong, Seok Hee Lee, and Nam Ik Cho. Deep clustering and block Hashing network for face image retrieval. In *ACCV*, pages 325–339. Springer, 2018.

[45] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2010.

[46] Dong-ju Jeong, Sung-Kwon Choo, Wonkyo Seo, and Nam Ik Cho. Classification-based supervised Hashing with complementary networks for image search. In *BMVC*, page 74, 2018.

[47] Yannis Kalantidis and Yannis Avrithis. Locally optimized product quantization for approximate nearest neighbor search. In *CVPR*, pages 2321–2328, 2014.

[48] Benjamin Klein and Lior Wolf. End-to-end supervised product quantization for image search and retrieval. In *CVPR*, pages 5041–5050, 2019.

[49] Bin Liu, Yue Cao, Mingsheng Long, Jianmin Wang, and Jingdong Wang. Deep triplet quantization. *ACM Multimedia*, 2018.

[50] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[51] Qingqun Ning, Jianke Zhu, Zhiyuan Zhong, Steven CH Hoi, and Chun Chen. Scalable image retrieval by sparse product quantization. *IEEE Transactions on Multimedia*, 19(3):586–597, 2016.

[52] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001.

[53] Alexandre Sablayrolles, Matthijs Douze, Nicolas Usunier, and Hervé Jégou. How should I evaluate supervised Hashing? In *ICASSP*, pages 1732–1736. IEEE, 2017.

[54] Kuniaki Saito, Shohei Yamamoto, Yoshitaka Ushiku, and Tatsuya Harada. Open set domain adaptation by backpropagation. In *ECCV*, pages 153–168, 2018.

[55] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823, 2015.

[56] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, pages 4077–4087, 2017.

[57] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *NeurIPS*, pages 1857–1865, 2016.

[58] Jingkuan Song, Lianli Gao, Li Liu, Xiaofeng Zhu, and Nicu Sebe. Quantization-based Hashing: a general framework for scalable image and video retrieval. *Pattern Recognition*, 75:175–187, 2018.

[59] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Sequential projection learning for Hashing with compact codes. In *ICML*, pages 1127??-1134, 2010.

[60] Yen-Cehng Wei-Yu and Jia-Bin Zsolt, Yu-Chiang. A closer look at few-shot classification. In *ICLR*, 2019.

[61] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.

[62] Xinyu Yan, Lijun Zhang, and Wu-Jun Li. Semi-supervised deep Hashing with a bipartite graph. In *IJCAI*, pages 3238–3244, 2017.

[63] Litao Yu, Zi Huang, Fumin Shen, Jingkuan Song, Heng Tao Shen, and Xiaofang Zhou. Bilinear optimized product quantization for scalable visual content analysis. *IEEE Transactions on Image Processing*, 26(10):5057–5069, 2017.

[64] Tan Yu, Junsong Yuan, Chen Fang, and Hailin Jin. Product quantization network for fast image retrieval. In *ECCV*, pages 186–201, 2018.

[65] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. 2017.

[66] Jian Zhang and Yuxin Peng. Ssdh: semi-supervised deep Hashing for large scale image retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(1):212–225, 2017.

[67] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. Simultaneous feature learning and hash coding with deep neural networks. In *CVPR*, pages 3270–3278, 2015.

[68] Artem Babenko and Victor Lempitsky. Additive quantization for extreme vector compression. In *CVPR*, pages 931–938, 2014.

[69] Artem Babenko and Victor Lempitsky. Tree quantization for large-scale similarity search and classification. In *CVPR*, pages 4240–4248, 2015.

[70] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020.

[71] Ting Chen, Yizhou Sun, Yue Shi, and Liangjie Hong. On sampling strategies for neural network-based collaborative filtering. In *ACM SIGKDD*, pages 767–776, 2017.

[72] et al. E. Riba. A survey on kornia: an open source differentiable computer vision library for pytorch. 2020.

[73] Bo Dai, Ruiqi Guo, Sanjiv Kumar, Niao He, and Le Song. Stochastic generative Hashing. In *ICML*, 2017.

[74] Kamran Ghasedi Dizaji, Feng Zheng, Najmeh Sadoughi, Yanhua Yang, Cheng Deng, and Heng Huang. Unsupervised deep generative adversarial Hashing network. In *CVPR*, pages 3664–3673, 2018.

[75] Kaiming He, Fang Wen, and Jian Sun. K-means Hashing: An affinity-preserving quantization method for learning binary compact codes. In *CVPR*, pages 2938–2945, 2013.

[76] Young Kyun Jang and Nam Ik Cho. Generalized product quantization network for semi-supervised image retrieval. In *CVPR*, 2020.

[77] Kevin Lin, Jiwen Lu, Chu-Song Chen, and Jie Zhou. Learning compact binary descriptors with unsupervised deep neural networks. In *CVPR*, pages 1183–1192, 2016.

[78] Wei Liu, Cun Mu, Sanjiv Kumar, and Shih-Fu Chang. Discrete graph Hashing. In *NeurIPS*, pages 3419–3427, 2014.

[79] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[80] Stanislav Morozov and Artem Babenko. Unsupervised neural quantization for compressed-domain similarity search. In *ICCV*, pages 3036–3045, 2019.

[81] Yuming Shen, Li Liu, and Ling Shao. Unsupervised binary representation learning with deep variational networks. *IJCV*, 127(11-12):1614–1628, 2019.

[82] Yuming Shen, Jie Qin, Jiaxin Chen, Mengyang Yu, Li Liu, Fan Zhu, Fumin Shen, and Ling Shao. Auto-encoding twin-bottleneck Hashing. In *CVPR*, pages 2818–2827, 2020.

[83] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.

[84] Jingkuan Song. Binary generative adversarial networks for image retrieval. In *AAAI*, 2017.

[85] Shupeng Su, Chao Zhang, Kai Han, and Yonghong Tian. Greedy hash: Towards fast optimization for accurate hash coding in cnn. In *NeurIPS*, pages 798–807, 2018.

[86] Erkun Yang, Tongliang Liu, Cheng Deng, Wei Liu, and Dacheng Tao. Distillhash: Unsupervised deep Hashing by distilling data pairs. In *CVPR*, pages 2946–2955, 2019.

[87] Li Yuan, Tao Wang, Xiaopeng Zhang, Francis EH Tay, Zequn Jie, Wei Liu, and Jiashi Feng. Central similarity quantization for efficient image and video retrieval. In *CVPR*, pages 3083–3092, 2020.

[88] Ting Zhang, Chao Du, and Jingdong Wang. Composite quantization for approximate nearest neighbor search. In *ICML*, volume 2, page 3, 2014.

[89] Ting Zhang, Guo-Jun Qi, Jinhui Tang, and Jingdong Wang. Sparse composite quantization. In *CVPR*, pages 4548–4556, 2015.

[90] Maciej Zieba, Piotr Semberecki, Tarek El-Gaaly, and Tomasz Trzcinski. Bingan: Learning compact binary descriptors with a regularized gan. In *NeurIPS*, pages 3608–3618, 2018.

[91] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, pages 380–388, 2002.

[92] Yun, Sukmin and Park, Jongjin and Lee, Kimin and Shin, Jinwoo. Regularizing class-wise predictions via self-knowledge distillation. In *CVPR*, pages 13876–13885, 2020.

[93] Touvron, Hugo and Cord, Matthieu and Douze, Matthijs and Massa, Francisco and Sablayrolles, Alexandre and Jégou, Hervé. Training data-efficient image transformers & distillation through attention. In *ICML*, pages 10347–10357, 2021.

[94] Cao, Yue and Long, Mingsheng and Liu, Bin and Wang, Jianmin. Deep cauchy hashing for hamming space retrieval. In *CVPR*, pages 1229–1237, 2018.

[95] Fan, Lixin and Ng, KamWoh and Ju, Ce and Zhang, Tianyu and Chan, Chee Seng. Deep Polarized Network for Supervised Learning of Accurate Binary Hashing Codes. In *IJCAI*, pages 825–831, 2020.

[96] Dosovitskiy, Alexey and Beyer, Lucas and Kolesnikov, Alexander and Weissenborn, Dirk and Zhai, Xiaohua and Unterthiner, Thomas and Dehghani, Mostafa and Minderer, Matthias and Heigold, Georg and Gelly, Sylvain and others. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*, 2020.

[97] Liu, Ze and Lin, Yutong and Cao, Yue and Hu, Han and Wei, Yixuan and Zhang, Zheng and Lin, Stephen and Guo, Baining. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.

[98] Gu, Geonmo and Ko, Byungsoo and Kim, Han-Gyu. Proxy synthesis: Learning with synthetic classes for deep metric learning. In *AAAI*, 2021.

초 록

방대한 데이터베이스에서 질의에 대한 관련 이미지를 찾는 콘텐츠 기반 이미지 검색은 컴퓨터 비전 분야의 근본적인 작업 중 하나이다. 특히 빠르고 정확한 검색을 수행하기 위해 해싱 (Hashing) 및 곱 양자화 (Product Quantization, PQ) 로 대표되는 근사최근접 이웃 (Approximate Nearest Neighbor, ANN) 검색 방식이 이미지 검색 커뮤니티에서 주목받고 있다. 신경망 기반 딥 러닝 (CNN-based deep learning) 이 많은 컴퓨터 비전 작업에서 우수한 성능을 보여준 이후로, 해싱 및 곱 양자화 기반 이미지 검색 시스템 모두 개선을 위해 딥 러닝을 채택하고 있다. 본 학위 논문에서는 적절한 검색 시스템을 제안하기 위해 다양한 딥 러닝 학습 환경아래에서 이미지 검색 방법을 제안한다. 구체적으로, 이미지 검색의 목적을 고려하여 의미적으로 유사한 이미지를 검색하는 딥 러닝 해싱 시스템을 개발하기 위한 지도 학습 방법을 제안하고, 의미적, 시각적으로 모두 유사한 이미지를 검색하는 딥 러닝 곱 양자화 기반의 시스템을 구축하기 위한 준지도, 비지도 학습 방법을 제안한다. 또한, 이미지 검색 데이터베이스의 특성을 고려하여, 분류해야할 클래스 (class category) 가 많은 얼굴 이미지 데이터 세트와 하나 이상의 레이블 (label) 이 지정된 일반 이미지 세트를 분리하여 따로 검색 시스템을 구축한다.

먼저 이미지에 부여된 의미론적 레이블을 사용하는 지도 학습을 도입하여 해싱 기반 검색 시스템을 구축한다. 클래스 간 유사성 (다른 사람 사이의 유사한 외모) 과 클래스 내 변화(같은 사람의 다른 포즈, 표정, 조명) 와 같은 얼굴 이미지 구별의 어려움을 해결하기 위해 각 이미지의 클래스 레이블을 사용한다. 얼굴 이미지 검색 품질을 더욱 향상시키기 위해 SGH (Similarity Guided Hashing) 방식을 제안하며,

여기서 다중 데이터 증강 결과를 사용한 자기 유사성 학습이 훈련 중에 사용된다. 그리고 해싱 기반의 일반 이미지 검색 시스템을 구성하기 위해 DHD(Deep Hash Distillation) 방식을 제안한다. DHD에서는 지도 신호를 활용하기 위해 클래스별 대표성을 나타내는 훈련 가능한 해시 프록시 (proxy) 를 도입한다. 또한, 해싱에 적합한 자체 증류 기법을 제안하여 증강 데이터의 잠재력을 일반적인 이미지 검색 성능 향상에 적용한다.

둘째로, 레이블이 지정된 이미지 데이터와 레이블이 지정되지 않은 이미지 데이터를 모두 활용하는 준지도 학습을 조사하여 곱 양자화 기반 검색 시스템을 구축한다. 지도 학습 딥 러닝 기반의 이미지 검색 방법들은 우수한 성능을 보이려면 값비싼 레이블 정보가 충분해야 한다는 단점이 있다. 게다가, 레이블이 지정되지 않은 수많은 이미지 데이터는 훈련에서 제외된다는 한계가 있다. 이 문제를 해결하기 위해 벡터 양자화 기반 반지도 영상 검색 방식인 GPQ (Generalized Product Quantization) 네트워크를 제안한다. 레이블이 지정된 데이터 간의 의미론적 유사성을 유지하는 새로운 메트릭 학습 (Metric learning) 전략과 레이블이 지정되지 않은 데이터의 고유한 잠재력을 최대한 활용하는 엔트로피 정규화 방법을 사용하여 검색 시스템을 개선한다. 이 솔루션은 양자화 네트워크의 일반화 용량을 증가시켜 이전의 한계를 극복할 수 있게한다.

마지막으로, 딥 러닝 모델이 사람의 지도 없이 시각적으로 유사한 이미지 검색을 수행할 수 있도록 하기 위해 비지도 학습 알고리즘을 탐색한다. 비록 레이블 주석을 활용한 심층 지도 기반의 방법들이 기존 방법들에 대비 우수한 검색 성능을 보일지라도, 방대한 양의 훈련 데이터에 대해 정확하게 레이블을 지정하는 것은 힘들고 주석에서 오류가 발생하기 쉽다는 한계가 있다. 이 문제를 해결하기 위해 레이블 없이 자체 지도 방식으로 훈련하는 SPQ (Self-supervised Product Quantization) 네트워크라는 심층 비지도 이미지 검색 방법을 제안한다. 새롭게 설계된 교차 양자화 대조 학습 방식으로 서로 다르게 변환된 이미지를 비교하여 곱 양자화의 코드워드와 심층 시각적 표현을 동시에 학습한다. 이 방식을 통해 이미지에 내제된 내용을 별도의 사람 지도 없이 네트워크가 스스로 이해하게 되고, 시각적으로 정확한 검색을 수행할 수 있는 설명 기능을 추출할 수 있게 된다.

벤치마크 데이터 세트에 대한 광범위한 이미지 검색 실험을 수행하여 제안된 방법이 다양한 평가 프로토콜에서 뛰어난 결과를 산출함을 확인했다. 지도 학습 기반의 얼굴 영상 검색의 경우 SGH는 저해상도 및 고해상도 얼굴 영상 모두에서 최고의 검색 성능을 달성하였고, DHD는 최고의 검색 정확도로 일반 영상 검색 실험에서 효율성을 입증한다. 준지도 일반 이미지 검색의 경우 GPQ는 레이블이 있는 이미지 데이터와 레이블이 없는 이미지 데이터를 모두 사용하는 프로토콜에 대한 최상의 검색 결과를 보여준다. 마지막으로, 비지도 학습 이미지 검색의 경우 지도 방식으로 미리 학습된 초기 값 없이도 SPQ를 사용하여 최상의 검색 점수를 얻었으며 시각적으로 유사한 이미지가 검색 결과로 성공적으로 검색되는 것을 관찰할 수 있다.

주요어: 이미지 검색, 얼굴 이미지 검색, 컨볼루션 신경망, 딥 러닝, 근사최근접이웃 검색, 해싱, 곱 양자화, 메트릭 학습, 반지도 학습, 비지도 학습

학번: 2016-20967