

Learning Effective Binary Representation with Deep Hashing Technique for Large-Scale Multimedia Similarity Search



Gengshen Wu
School of Computing and Communications
Lancaster University

A thesis submitted for the degree of
Doctor of Philosophy

September 2020

I would like to dedicate this thesis to my family

Declaration

This thesis has not been submitted in support of an application for another degree at this or any other university. It is the result of my own work and includes nothing that is the outcome of work done in collaboration except where specifically indicated. Many of the ideas in this thesis were the product of discussion with my supervisor Dr. Jungong Han.

Parts of this thesis have been published previously in the following publications:

[Chapter 3]

G. Wu, Z. Lin, G. Ding, Q. Ni and J. Han. “On Aggregation of Unsupervised Deep Binary Descriptor with Weak Bits,” IEEE Transactions on Image Processing, 2020. (Accepted)

[Chapter 4]

G. Wu, L. Liu, Y. Guo, G. Ding, J. Han, J. Shen, and L. Shao. “Unsupervised deep video hashing with balanced rotation,” in International Joint Conference on Artificial Intelligence, 2017, pp. 3076-3082.

G. Wu, J. Han, Y. Guo, L. Liu, G. Ding, Q. Ni, and L. Shao. “Unsupervised Deep Video Hashing via Balanced Code for Large-Scale Video Retrieval,” IEEE Transactions on Image Processing, vol. 28, no. 4, pp. 1993-2007, April 2019.

[Chapter 5]

G. Wu, Z. Lin, J. Han, L. Liu, G. Ding, B. Zhang, and J. Shen. “Unsupervised Deep Hashing via Binary Latent Factor Models for Large-scale Cross-modal Retrieval,” in International Joint Conference on Artificial Intelligence, 2018, pp. 2854-2860.

G. Wu, J. Han, Z. Lin, G. Ding, B. Zhang and Q. Ni. “Joint Image-Text Hashing for Fast Large-Scale Cross-Media Retrieval Using Self-Supervised Deep Learning,” IEEE Transactions on Industrial Electronics, vol. 66, no. 12, pp. 9868-9877, Dec. 2019.

Gengshen Wu
September, 2020

Acknowledgements

First of all, I would like to thank my supervisor Dr. Jungong Han, who allowed me to complete my Ph.D. study at Lancaster University. Under his supervision, we enjoy unprecedented freedom of research in the academic field. He has been providing me with continuous support, insightful guidance, inspiring ideas and shaping me to be a qualified researcher.

I would like to extend the sincerest thanks to the following people, who have all helped in the completion of this thesis.

I would like to thank Prof. Qiang Ni, Dr. Matthew Broadbent, Dr. Zheng Wang and Dr. Hossein Rahmani in our school for their kind support and suggestions.

I would like to thank the previous and current visiting scholars in our lab, including Prof. Heng Liu, Prof. Hai Li, and Dr. Yi Liu, Dr. Yuanjun Huang and Dr. Shanfeng Wang for their helpful discussions and research ideas.

I would also like to thank my co-authors, Prof. Ling Shao, Prof. Guiguang Ding, Dr. Jialie Shen, Dr. Baochang Zhang, Dr. Yuchen Guo, Dr. Zijia Lin and Dr. Li Liu, who have improved my research outcomes from many aspects.

I would like to thank Dr. Yi Zhou, who introduced me to Dr. Jungong Han as a Ph.D. candidate and let the story began.

I would also like to thank all previous members from Northumbria University, Lancaster University, Warwick University, and East Anglia University, including Dr. Yuming Shen, Dr. Bingzhang Hu, Dr. Yijun Shen, Dr. Jingtian Zhang, Dr. Shanfeng Hu, Dr. Tianhao Guo, Dr. Zheming Zuo, Dr. Xiaowei Gu, Dr. Zhaoxu Yang, Dr. Haiyang Liu, Dr. Binta He, Dr. Ning Gao, Dr. Wenda Tang, Shoujiang Xu, Jiaojiao Zhao, Shuo Zhang, Yao Zhang, Matthew Gingfung Yeung, Binbin Su, Peng Cheng, Yunqi

Miao, Mingqi Gao, and Zhuang Shao. Thank them for their friendly help during my Ph.D. study.

I would like to thank Northumbria University and Lancaster University, they provided the financial support for me to complete my Ph.D. research.

Many thanks to my family for their unconditional support during my Ph.D. study.

Abstract

The explosive growth of multimedia data in modern times inspires the research of performing an efficient large-scale multimedia similarity search in the existing information retrieval systems. In the past decades, the hashing-based nearest neighbor search methods draw extensive attention in this research field. By representing the original data with compact hash code, it enables the efficient similarity retrieval by only conducting bitwise operation when computing the Hamming distance. Moreover, less memory space is required to process and store the massive amounts of features for the search engines owing to the nature of compact binary code. These advantages make hashing a competitive option in large-scale visual-related retrieval tasks. Motivated by the previous dedicated works, this thesis focuses on learning compact binary representation via hashing techniques for the large-scale multimedia similarity search tasks. Particularly, several novel frameworks are proposed for popular hashing-based applications like a local binary descriptor for patch-level matching (Chapter 3), video-to-video retrieval (Chapter 4) and cross-modality retrieval (Chapter 5).

This thesis starts by addressing the problem of learning local binary descriptor for better patch/image matching performance. To this end, we propose a novel local descriptor termed Unsupervised Deep Binary Descriptor (UDBD) for the patch-level matching tasks, which learns the transformation invariant binary descriptor via embedding the original visual data and their transformed sets into a common Hamming space. By imposing a $\ell_{2,1}$ -norm regularizer on the objective function, the learned binary descriptor gains robustness against noises. Moreover, a weak bit scheme is applied to address the ambiguous matching in the local binary descriptor, where the best match is determined for each query by comparing a series of weak bits between the query instance and the candidates, thus improving the matching performance.

Furthermore, Unsupervised Deep Video Hashing (UDVH) is proposed to facilitate large-scale video-to-video retrieval. To tackle the imbalanced distribution issue in the video feature, balanced rotation is developed to identify a proper projection matrix such that the information of each dimension can be balanced in the fixed-bit quantization, thus improving the retrieval performance dramatically with better code quality. To provide comprehensive insights on the proposed rotation, two different video feature learning structures: stacked LSTM units (UDVH-LSTM) and Temporal Segment Network (UDVH-TSN) are presented in Chapter 4.

Lastly, we extend the research topic from single-modality to cross-modality retrieval, where Self-Supervised Deep Multimodal Hashing (SSDMH) based on matrix factorization is proposed to learn unified binary code for different modalities directly without the need for relaxation. By minimizing graph regularization loss, it is prone to produce discriminative hash code via preserving the original data structure. Moreover, Binary Gradient Descent (BGD) accelerates the discrete optimization against the bit-by-bit fashion. Besides, an unsupervised version termed Unsupervised Deep Cross-Modal Hashing (UDCMH) is proposed to tackle the large-scale cross-modality retrieval when prior knowledge is unavailable.

List of Publications

Conference Papers

1. **G. Wu**, L. Liu, Y. Guo, G. Ding, J. Han, J. Shen, and L. Shao. “Unsupervised deep video hashing with balanced rotation,” in International Joint Conference on Artificial Intelligence, 2017, pp. 3076-3082. (CORE A* conference)
2. **G. Wu**, Z. Lin, J. Han, L. Liu, G. Ding, B. Zhang, and J. Shen. “Unsupervised Deep Hashing via Binary Latent Factor Models for Large-scale Cross-modal Retrieval,” in International Joint Conference on Artificial Intelligence, 2018, pp. 2854-2860. (CORE A* conference)

Journal Papers

1. **G. Wu**, J. Han, Y. Guo, L. Liu, G. Ding, Q. Ni, and L. Shao. “Unsupervised Deep Video Hashing via Balanced Code for Large-Scale Video Retrieval,” IEEE Transactions on Image Processing, vol. 28, no. 4, pp. 1993-2007, April 2019. (IF: 6.79)
2. **G. Wu**, J. Han, Z. Lin, G. Ding, B. Zhang and Q. Ni. “Joint Image-Text Hashing for Fast Large-Scale Cross-Media Retrieval Using Self-Supervised Deep Learning,” IEEE Transactions on Industrial Electronics, vol. 66, no. 12, pp. 9868-9877, Dec. 2019. (IF: 7.503)
3. **G. Wu**, Z. Lin, G. Ding, Q. Ni and J. Han. “On Aggregation of Unsupervised Deep Binary Descriptor with Weak Bits,” IEEE Transactions on Image Processing, 2020. (IF: 6.79; Accepted)

Co-Authored Papers

1. Y. Qi, J. Gu, Y. Zhang, **G. Wu**, and F. Wang. “Supervised Deep Semantics-Preserving Hashing for Real-Time Pulmonary Nodule Image Retrieval,” *Journal of Real-Time Image Processing*, 2020.
2. S. Zhang, **G. Wu**, and J. Han. “Pruning Filter with Attention Mechanism for Deep Networks Compression on Remote Sensing Image,” *Remote Sensing*, 2020.

Contents

Declaration	ii
Acknowledgements	iii
Abstract	v
List of Publications	vii
List of Tables	xiv
List of Figures	xvi
List of Acronyms	xx
List of Algorithms	xxv
1 Introduction and Background Theory	1
1.1 Research Background	1
1.2 Fundamental Theory of Hashing Technique	4
1.2.1 Hash Function	4
1.2.2 Objective Function Construction	6
1.2.3 Optimization Strategy	7
1.2.3.1 Continuous Relaxation	7
1.2.3.2 Alternative Optimization	8
1.2.3.3 Coordinate Descent	8
1.2.4 Fast Similarity Search with Hash Code	9
1.2.4.1 Hash Code Ranking	9
1.2.4.2 Hash Table Lookup	10
1.2.5 Evaluation Metrics	11
1.2.5.1 Precision@K	11

1.2.5.2	Mean Average Precision	11
1.2.5.3	Precision-Recall Curve	12
1.2.5.4	Receiver Operating Characteristic Curve	12
1.3	Research Problems and Challenges	13
1.3.1	Local Binary Descriptor	13
1.3.2	Video Hashing	15
1.3.3	Cross-Modality Hashing	16
1.4	Overview of Contributions	19
1.5	Thesis Outline	21
2	Literature Review on Hashing-Based Similarity Search	23
2.1	Single-Modality Similarity Search	23
2.1.1	Image Hashing	24
2.1.2	Local Feature Descriptor	27
2.1.2.1	Handcrafted Feature Descriptors	27
2.1.2.2	Learning-Based Feature Descriptors	28
2.1.3	Video Hashing	30
2.1.3.1	Early Video Hashing	30
2.1.3.2	Deep Learning Based Video Hashing	31
2.2	Cross-Modality Similarity Search	33
2.2.1	Supervised Cross-Modal Hashing	33
2.2.2	Unsupervised Cross-Modal Hashing	36
2.3	Chapter Summary	37
3	Unsupervised Deep Binary Descriptor	39
3.1	Introduction	39
3.2	Methodology	43
3.2.1	Framework Overview	43
3.2.2	Learning Unified Binary Descriptor	44
3.2.2.1	Collective Binary Embedding	44
3.2.2.2	Unsupervised Graph Learning	45
3.2.3	Optimization Algorithm	46
3.2.3.1	\mathbf{W}_v Step	47
3.2.3.2	\mathbf{B} Step	47
3.2.3.3	α_v Step	48
3.2.4	Generating Out-of-Sample Binary Descriptor	49
3.2.5	Refined Matching via Weak Bit Selection	49

3.3	Experiment	51
3.3.1	Dataset Descriptions	51
3.3.1.1	Brown	51
3.3.1.2	Cifar-10	51
3.3.1.3	HPatches	52
3.3.2	Implementation Details	52
3.3.3	Comparisons with State-of-The-Arts	52
3.3.3.1	Results on Brown Dataset	52
3.3.3.2	Results on Cifar-10 Dataset	55
3.3.3.3	Results on HPatches Dataset	57
3.3.4	Further Analysis	58
3.3.4.1	Ablation Study	58
3.3.4.2	Transformation Invariance	59
3.3.4.3	Weak Bit Study	60
3.3.4.4	Loss Term	60
3.3.4.5	Parameter Analysis	61
3.4	Chapter Summary	61
4	Unsupervised Deep Video Hashing	63
4.1	Introduction	63
4.2	Proposed Unsupervised Deep Video Hashing	66
4.2.1	Deep Video Feature Learning	67
4.2.1.1	UDVH-LSTM	67
4.2.1.2	UDVH-TSN	69
4.2.2	Feature Embedding with Pseudo Labels	70
4.2.3	Balanced Rotation	71
4.2.4	Objective Function and Optimization	73
4.2.4.1	R Step	75
4.2.4.2	B Step	76
4.2.4.3	Θ Step	76
4.2.5	Complexity Analysis	77
4.2.6	Discussion	78
4.3	Experiments	79
4.3.1	Datasets and Experimental Setting	79
4.3.1.1	FCVID	79
4.3.1.2	YFCC	79

4.3.1.3	ActivityNet	80
4.3.2	Baselines	80
4.3.3	Implementation Details	80
4.3.4	Evaluation Metrics	81
4.3.5	Comparison with State-of-The-Arts	82
4.3.5.1	Results from UDVH-LSTM	83
4.3.5.2	Results from UDVH-TSN	84
4.3.5.3	Discussion	88
4.3.6	Architecture Investigation	89
4.3.6.1	Parameter Analysis	89
4.3.6.2	Binarization Investigation	90
4.3.6.3	Loss Function	90
4.3.7	Feature Selection	92
4.3.8	Efficiency Analysis	93
4.4	Chapter Summary	94
5	Deep Cross Modal Hashing	97
5.1	Introduction	97
5.2	Proposed Method	101
5.2.1	Problem Definition	101
5.2.2	Deep Architecture	102
5.2.3	Regularized Binary Latent Model	103
5.2.3.1	Binary Reconstruction Loss	103
5.2.3.2	Graph Regularization Loss	104
5.2.4	Deep Hash Function Learning	104
5.2.5	Objective Function and Optimization	105
5.2.5.1	\mathbf{U}_i Step	105
5.2.5.2	\mathbf{B} Step	105
5.2.5.3	α_i Step	107
5.2.5.4	Θ_i Step	107
5.2.6	Computational Complexity	108
5.2.7	Extension to Unsupervised Cross-Modal Hashing	108
5.3	Experiment	109
5.3.1	Dataset Descriptions	110
5.3.1.1	Wiki	110
5.3.1.2	MIRFlickr	110

5.3.1.3	NUS-WIDE	110
5.3.2	Experiment Settings	110
5.3.3	Results and Analysis	112
5.3.3.1	Architecture Investigation	112
5.3.3.2	Overall Comparisons with Baselines	112
5.3.3.3	Top-5 Retrieved Examples for SSDMH	113
5.3.3.4	Effect of Training Data Size	116
5.3.3.5	Parameter Sensitivity Analysis	119
5.3.3.6	Convergence Study	119
5.3.3.7	BGD versus One Entry	119
5.3.3.8	Training Efficiency	121
5.3.4	Quantitative Results for UDCMH	122
5.3.4.1	Comparison With State-of-The-Arts	122
5.3.4.2	Training Data Size	125
5.4	Chapter Summary	125
6	Conclusions and Future Work	126
6.1	Thesis Summary	126
6.1.1	Unsupervised Deep Binary Descriptor	126
6.1.2	Unsupervised Deep Video Hashing	127
6.1.3	Deep Cross-Modal Hashing	127
6.2	Future Research Topics	128
6.2.1	Hashing for Deep Binary Neural Network	128
6.2.2	Online Hashing	128
6.2.3	Fine-Grained Retrieval with Weighted Hamming Distance	129
6.2.4	Fast Person Re-Identification	130
	Bibliography	131

List of Tables

3.1	Mathematical symbols and their short descriptions.	44
3.2	Comparison of the proposed UDBD to the state-of-the-art binary descriptors in terms of FPR@95% on Brown dataset. Dim, SP and USP denote dimension, supervised and unsupervised, respectively. \dagger and \ddagger indicate the train and testing subsets. The results from SIFT and supervised methods are provided as references. Bold values are the best results in unsupervised binary descriptors.	53
3.3	mAP of Top 1,000 (%) returned images at different code length from various unsupervised descriptors on Cifar-10 dataset. Bold values are the best results.	56
3.4	Precision at Top 1 on Cifar-10 dataset when using DeepBit, BinGAN, GraphBit and UDBD at different bit sizes.	56
3.5	Comparison of the proposed UDBD to the state-of-the-art descriptors in terms of mAP (%) on HPatches dataset. Dim, SP and USP denote dimension, supervised and unsupervised, respectively. The real-valued descriptor (SIFT) and the supervised methods are provided as references. Bold values are the best results in unsupervised binary descriptors.	58
3.6	Ablation study on Brown (FPR@95%): <i>Liberty</i> \rightarrow <i>Notre Dame</i> and <i>Yosemite</i> \rightarrow <i>Liberty</i> , HPatches: <i>matching</i> (mAP) and Cifar-10 at 64 bits (mAP@1000) when $\gamma = 0$ (i.e., UDBD $_{\gamma=0}$) and $\beta = 0$ (i.e., UDBD $_{\beta=0}$).	59
3.7	mAP variations on HPatches with/without using weak bit scheme (UDBD ‡ /UDBD †). Bold values show the best results.	60
3.8	Performance variations on Brown (FPR@95%): <i>Notre Dame</i> \rightarrow <i>Liberty</i> and <i>Liberty</i> \rightarrow <i>Notre Dame</i> , HPatches: <i>matching</i> (mAP) at 256 bits, and Cifar-10 at 32 bits (mAP@1,000) when using $\ell_{2,1}$ -norm and $\ell_{2,2}$ -norm loss terms.	61
4.1	Mathematical symbols and their short descriptions.	67

4.2	The network configurations UDVH-LSTM and UDVH-TSN. Other layers like pooling and activation are omitted for concise descriptions. . .	70
4.3	mAP@20 of different cluster numbers at 128 bits on three datasets under UDVH-LSTM and UDVH-TSN.	89
4.4	mAP@K of 128 bits when using various banarization schemes separately in the code learning of UDVH-LSTM and UDVH-TSN.	91
4.5	mAP@20 at 128 bits when applying various loss functions accordingly during the hash function learning of UDVH-LSTM and UDVH-TSN.	92
4.6	The training time at various code lengths on FCVID when using SSTH, UDVH-LSTM and UDVH-TSN. The time unit for network training is hour (h) and the rest processes are reported in second (s).	94
5.1	The Network Configurations for Two Modalities. Other layers like pooling and activation are omitted for concise descriptions.	101
5.2	Mathematical symbols and their short descriptions.	102
5.3	mAP results at the code length of 128 when involving various loss terms deployed in the proposed method: SSDMH _{brl} and SSDMH _{brl+grl}	112
5.4	The variations on $\alpha_i (i = 1, 2)$ during the optimization iteration at 128 bits on three datasets. α_i are initialized as 0.5.	113
5.5	mAP results for Image→Text and Text→Image tasks on three datasets at various code lengths (bits) when using different methods. The best performance is shown in boldface.	114
5.6	Effect of training data size on MIRFlickr and NUS-WIDE at the code length of 64.	119
5.7	Time costs (in seconds) in the training processes of SSDMH on three datasets at 128 bits for one loop (T).	121
5.8	mAP results for Image→Text and Text→Image tasks on three datasets at various code lengths (bits) when using different unsupervised methods and UDCMH. The Best Performance is shown in boldface.	123
5.9	Effect of training data size on NUS-WIDE at 64 bits. k indicates 1,000.	125

List of Figures

1.1	The general procedure of CBIR.	2
1.2	An example of linear and nonlinear models. Given two different data distributions (triangle and parallelogram), the nonlinear model can better fit the complicated data distributions than the linear one. . . .	5
1.3	The search procedure of hash code ranking. The ranked results are obtained based on the Hamming distances (D_h) in an ascending order.	9
1.4	(a) 4-bit hash code is used as an example. The returned items would be x_1, x_4, x_5, x_2, x_3 and x_6 if the Hamming radius is 1; (b) $M(M > 1)$ hash tables are constructed in the multiple table lookup and the neighbor search is conducted across these tables afterward.	10
1.5	The work pipeline of traditional image matching using local feature descriptor, which consists of three major steps: interest point detection, feature descriptor learning and feature matching. The feature learning and matching processes are conducted at the patch level. The input images might be photographed from different views or even processed with various affine transformations.	13
1.6	The general pipeline of video retrieval.	15
1.7	The general workflow of cross-modality retrieval. Two cross-modal retrieval tasks: image→text and text→image, are used as examples.	17
3.1	(a) An example of the Hamming distance distribution on Cifar-10 dataset at 64 bits, where 3 candidates are returned from the database with the same minimum Hamming distance of 16 to the query; (b) Noise effects on Brown dataset (train: <i>Yosemite</i> and test: <i>Liberty</i>) at 256 bits under $\ell_{2,1}$ -norm and $\ell_{2,2}$ -norm losses, where a sharper performance decline from $\ell_{2,2}$ -norm against $\ell_{2,1}$ -norm loss is observed at certain noise level.	40

3.2	The proposed binary descriptor learning framework is made up of deep feature extraction, unified binary code learning and deep embedding function learning. The descriptor size is set to 3 as an example. Best viewed in color.	41
3.3	An example on the weak bit selection process. The red circles denote the marked weak bits with the values between $(-th, th)$	50
3.4	The ROC curves under different settings on Brown dataset when using various unsupervised binary descriptors. Best viewed in color.	54
3.5	The matching results of <i>Notre Dame</i> , <i>Yosemite</i> and <i>Liberty</i> of UDBD on Brown dataset, which includes three matched pairs and two mismatched pairs for each subset.	55
3.6	Precision-Recall curves of the proposed method and the baselines on Cifar-10 dataset at 16, 32 and 64 bits.	56
3.7	The top-10 retrieved images on Cifar-10 dataset by UDBD at the code length of 64. The query images are selected from four categories: cat, car, airplane and bird. Red rectangle indicates the wrong results. . .	57
3.8	Performances variations under different rotation angles on test instances from GraphBit, DeepBit and UDBD.	59
3.9	Performance variations at varying code lengths with/without using weak bit scheme. (a) FPR@95% on Brown: <i>Notre Dame</i> (<i>ND</i>) \rightarrow <i>Liberty</i> (<i>Lib</i>) and <i>Liberty</i> \rightarrow <i>Notre Dame</i> ; (b) Precision@Top 1 on Cifar-10. . .	60
3.10	Parameter sensitivity analysis of γ and β at various bit sizes on Cifar-10 dataset.	61
4.1	(a) Suppose that two data samples (red and green) from a benchmark are projected into a two-dimensional feature space with the coordinates of (x_1, y_1) and (x_2, y_2) and encoded by two bits subsequently, where $ x_1 > y_1 $, $ x_2 > y_2 $ and $ x_1 - x_2 > y_1 - y_2 $. After the proposed balanced rotation, the coordinates of two data points change to (x_1^r, y_1^r) and (x_2^r, y_2^r) accordingly, where $ x_1^r = y_1^r $ and $ x_2^r = y_2^r $. Obviously, compared to the original features, the variances contained in the X and Y axis can be balanced with such rotation strategy applied. That indicates two dimensions of the rotated data points will have the same impact on the calculation of the Hamming distances when encoding them with the fixed number of bits; (b) The variance within each dimension of image and video feature.	65

4.2	The basic framework of UD VH-LSTM. The whole process consists of three subsections: video feature extraction, unsupervised code learning and deep hash function learning, which are performed iteratively to obtain the solution.	68
4.3	The basic framework of UD VH-TSN. The only difference between UD VH-TSN and UD VH-LSTM is that they adopt LSTM and TSN separately to evaluate the video representation.	70
4.4	The graph illustration of Algorithm 2.	77
4.5	The mAP@K curves at different bit sizes under UD VH-LSTM.	82
4.6	The Precision-Recall curves at 128 bits under UD VH-LSTM.	83
4.7	The Precision@100 curves at various bit sizes under UD VH-LSTM.	83
4.8	Top-5 retrieval results when using SSTH and UD VH-LSTM at the code length of 128 bits.	85
4.9	The mAP@K curves at different bit sizes under UD VH-TSN.	86
4.10	The Precision-Recall curves at 128 bits under UD VH-TSN.	87
4.11	The Precision@100 curves at various bit sizes under UD VH-TSN.	87
4.12	Top-5 retrieval results when using SSTH and UD VH-TSN at the code length of 128 bits, where examples are randomly selected from the video datasets. The left column shows the query videos, the middle blocks and right blocks show the top-5 returned videos by SSTH and UD VH-TSN, respectively. Red rectangles indicate mistakes.	88
4.13	The variance distribution from FCVID at 128 bits under UD VH-TSN.	92
4.14	The mAP@5 variations when using CNN and TSN features separately in the image-based hashing methods, where the solid and dot lines represent the results on TSN and CNN features, respectively.	93
4.15	The mAP@5 variations when using CNN and TSN features separately in the video-based hashing methods, where the solid and dot lines represent the results on TSN and CNN features, respectively.	93
4.16	An example of cross-modality similarity search: text query image, in Google Images.	95

5.1	The overview of our Self-Supervised Deep Multimodal Hashing. There are three subsections in the training process: deep feature learning (left), deep hash function learning (middle) and regularized binary latent representation learning (right). Specifically, the regularized binary latent model consists of two loss terms: binary reconstruction loss and graph regularization loss. The yellow arrows indicate the deep feature learning. The blue arrows show the iterative directions when learning deep hash functions with the guidance of the unified binary code \mathbf{B} . Better viewed in color.	99
5.2	The Precision-Recall curves at 128 bits on three datasets.	115
5.3	The top-5 retrieved results at 128 bits on Wiki dataset.	116
5.4	The top-5 retrieved results at 128 bits on MIRFlickr dataset.	117
5.5	The top-5 retrieved results at 128 bits on NUS-WIDE dataset.	118
5.6	mAP versus γ , β and λ at 64 bits on three datasets.	120
5.7	(a) Objective function values after each iteration (t) when solving the unified binary code at 128 bits; (b) Euclidean losses after every iteration (T) when learning the deep hash functions at 128 bits.	121
5.8	Time costs (in seconds) in optimizing one row of the unified binary code at 128 bits on three datasets when using BGD and One Entry [153] separately.	122
5.9	The precision@N curves at 128 bits on all datasets.	124

List of Acronyms

AGH	Anchor Graph Hashing
ANN	Approximate Nearest Neighbor
AVH	Attention-based Video Hashing
BLSTM	Binary Long Short Term Memory
BOLD	Binary Online Learned Descriptor
BGD	Binary Gradient Descend
BRE	Binary Reconstructive Embedding
BRIEF	Binary Robust Independent Elementary Features
BRISK	Binary Robust Invariant Scalable Keypoints
BQP	Binary Quadratic Programming
CAH	Correlation Autoencoder Hashing
CA-LBFL	Context-Aware Local Binary Feature Learning
C-CBFD	Coupled Compact Binary Face Descriptor
CBIR	Content-Based Image Retrieval
CCA	Canonical Correlation Analysis
CDbin	Compact Discriminative binary descriptors
CDQ	Collective Deep Quantization
CMFH	Collective Matrix Factorization Hashing
CMLA	Cross-Modal correlation Learning with Adversarial samples
CMSSH	Cross-Modal Similarity Sensitive Hashing
CNN	Convolutional Neural Network

CNNH	Convolutional Neural Network Hashing
CT	Computed Tomographic
CVH	Cross-View Hashing
CYC-DGH	Cycle-Consistent Deep Generative Hashing
DBD-MQ	Deep Binary Descriptor with Multi-Quantization
DBRC	Deep Binary Reconstruction
D-BRIEF	Discriminative BRIEF
DCC	Discrete Cyclic Coordinate Descent
DCMH	Deep Cross-Modal Hashing
DDBC	Discriminant Direction Binary Code
DLFH	Discrete Latent Factor model based cross-modal Hashing
DGH	Discrete Graph Hashing
DH	Deep Hashing
DHH	Deep Heterogeneous Hashing
DisCMH	Discriminant Cross-Modal Hashing
DJSRH	Deep Joint-Semantics Reconstructing Hashing
DNNH	Deep Neural Network Hashing
DOAP	Descriptors Optimized for Average Precision
DSADH	Dual-Supervised Attention Network for Deep Hashing
DSH	Deep Sketch Hashing
DVB	Deep Variational Binaries
DVH	Deep Video Hashing
DVSH	Deep Visual-Semantic Hashing
EGDH	Equally-Guided Discriminative Hashing
FCVID	Fudan-Columbia Video Dataset
FPR@95%	False Positive Rate at 95% true positive recall rate
FREAK	Fast Retina Keypoint

GAN	Generative Adversarial Network
GCH	Graph Convolutional Hashing
GCN	Graph Convolutional Network
GPU	Graphics Processing Unit
HamH	Harmonious Hashing
HER	Hashing across Euclidean space and Riemannian manifold
HPatches	Homography Patches
IMH	Inter-Media Hashing
IMVH	Iterative Multi-View Hashing
IsoH	Isotropic Hashing
ITQ	Iterative Quantization
ITQ+	Iterative Quantization Plus
JMVH	Joint Multi-View Hashing
KAEs	K-AutoEncoders
KD-tree	K-Dimensional tree
KLSH	Kernelized Locality Sensitive Hashing
KMH	K-Means hashing
KNN	K Nearest Neighbour
KSH	Kernel Supervised Hashing
LBP	Local Binary Pattern
LCMH	Linear Cross-Modal Hashing
LDAHash	Linear Discriminant Analysis Hashing
LDLP	Local Diagonal Laplacian Pattern
LSH	Locality Sensitive Hashing
LSSH	Latent Semantic Sparse Hashing
LSTM	Long Short-Term Memory
MAP	Mean Average Precision

MFH	Multiple Feature Hashing
MGAH	Multi-pathway Generative Adversarial Hashing
MLP	Multi-Layer Perceptrons
MSAE	Multi-modal Stacked Auto-Encoder
NDH	Nonlinear Discrete Hashing
NN	Nearest Neighbor
NP	Non-deterministic Polynomial
NPH	Neighborhood Preserving Hashing
NSH	Nonlinear Structural Hashing
ORB	Oriented fast and Rotated BRIEF
OKH	Online Kernel-based Hashing
OSH	Online Sketching Hashing
PCA	Principal Component Analysis
PR curve	Precision-Recall curve
Precision@K	Precision at top-K retrieved candidates
QCH	Quantized Correlation Hashing
RFDH	Robust and Flexible Discrete Hashing
RNN	Recurrent Neural Network
RPN	Region Proposal Network
RFDH	Robust and Flexible Discrete Hashing
RI-LBD	Rotation-Invariant Local Binary Descriptor
SAE	Semantic Auto-Encoder
SBIR	Sketch-Based Image Retrieval
SCM	Semantic Correlation Maximization
SCMFH	Supervised Collective Matrix Factorization Hashing
SDCH	Semantic Deep Cross-modal Hashing
SDH	Supervised Discrete Hashing

SePH	Semantics-Preserving Hashing
SGD	Stochastic Gradient Descent
SIFT	Scale-Invariant Feature Transform
SLBFLE	Simultaneous Local Binary Feature Learning and Encoding
SMVH	Stochastic Multi-view Hashing
SpH	Spherical Hashing
SP	SPECTral hashing
SPDTH	Similarity-Preserving Deep Temporal Hashing
SRBD	Superpixel Region Binary Descriptor
SSDMH	Self-Supervised Deep Multimodal Hashing
SSTH	Self-Supervised Temporal Hashing
SSE	Semantic Similarity Embedding
SSVH	Self-Supervised Video Hashing
SUBIC	SUPERvised structured BINARY Code
Submod	Submodular video hashing
SURF	Speeded Up Robust Features
SVM	Support Vector Machine
TDH	Triplet-based Deep Hashing
TSN	Temporal Segment Network
TVDB	Textual-Visual Deep Binaries
UDBD	Unsupervised Deep Binary Descriptor
UDCH-VLR	Unsupervised Deep Cross-modal Hashing with Virtual Label Regression
UDCMH	Unsupervised Deep Cross-Modal Hashing
UDVH	Unsupervised Deep Video Hashing
YFCC	Yahoo Flickr Creative Common

List of Algorithms

1	Unsupervised Deep Binary Descriptor	50
2	Unsupervised Deep Video Hashing	76
3	Self-Supervised Deep Multimodal Hashing	108

Chapter 1

Introduction and Background Theory

1.1 Research Background

Living in the Internet era of big data, the explosive amount of data, which varies in diverse forms like image, text, audio, and video, has become extremely challenging more than ever for the existing multimedia search engines and recommendation systems. According to recent public statistics from two popular social media websites, the average number of photos being shared every day on *Flickr*¹ is more than 1 million. In comparison, over 500 hours of videos are uploaded per minute by the active users on *Youtube*². To tackle such overwhelming data sources, how to perform accurate and efficient content-based similarity retrieval, which is a core technology of the current multimedia systems, has aroused extensive attention from both industry and academia.

The similarity retrieval problem, also known as Nearest Neighbor (NN) search, can be described as a search process of finding the most relevant semantic (similar) candidates from a large gallery set for a given query sample [149, 165, 193]. Particularly, given a query feature vector $x_q \in \mathbb{R}^d$, a gallery set consists of n feature vectors $\mathbf{X} = [x_i]_{i=1}^n \in \mathbb{R}^{d \times n}$, d is the dimensionality, the nearest neighbor search problem can be formulated as:

$$NN(x_q) = \arg \min_{x \in \mathbf{X}} dist(x_q, x), \quad (1.1)$$

¹<https://expandedramblings.com/index.php/flickr-stats/>

²<https://www.tubefilter.com/2019/05/07/number-hours-video-uploaded-to-youtube-per-minute/>

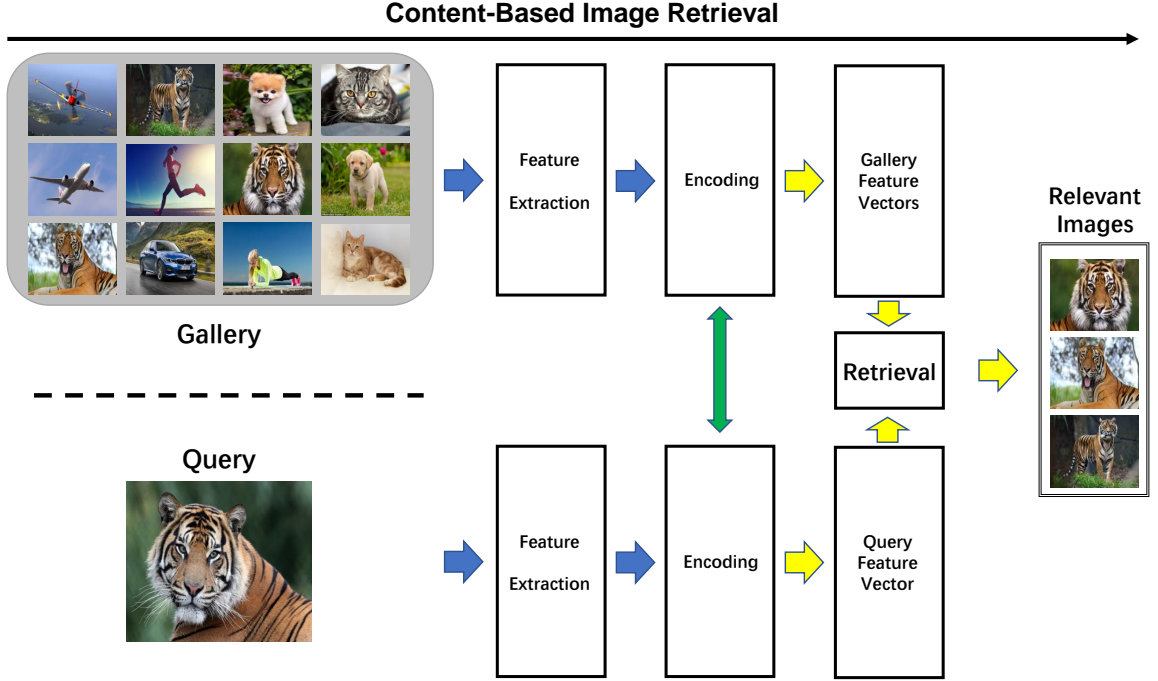


Figure 1.1: The general procedure of CBIR.

where $dist(.)$ represents a specific distance metric (e.g., Euclidean distance) that determines the closest candidates to x_q in the feature space. To further clarify the similarity retrieval process, a simple flowchart of the general content-based image retrieval (CBIR) [165] is presented in Fig. 1.1, which could be easily extended to other related tasks involving different data types. In CBIR, images are firstly represented with various feature vectors and then encoded into alternative representations following certain patterns (i.e., encoding function). Here, the encoding function, which is also the learning objective in most search frameworks, should be carefully designed to obtain better performance in the upcoming retrieval tasks. Then, the gallery and query data are pre-computed by the learned encoding function and their encoded feature vectors are measured under a distance metric. By sorting those distances in an ascending order, the candidates from the gallery with the smallest distances are returned as the relevant (i.e., similar) neighbors to one specific query. Generally speaking, the search efficiency depends on the computational complexity of the retrieval phase. At the same time, the search accuracy is usually determined by the proper design of the encoding mechanism when using fixed feature extractor [7, 193].

The earliest works in the research of similarity retrieval perform the exhaustive/exact Nearest Neighbor (NN) search in the retrieval process. However, such a search strategy (i.e., linear scan) shows weakness in tackling large-scale datasets

with numerous samples in practical applications. Later on, some tree-based search schemes are proposed to subdivide the feature space for data samples via employing various tree structures for fast search. Two representative methods are K-Dimensional tree (KD-tree) and Randomized KD-tree [46, 132], which indexes the data for quick query responses. However, these methods cannot handle the high-dimensional cases, known as the curse of dimensionality, where the computational costs grow exponentially along with the increasing dimension, thus making it less favorable in large-scale retrieval applications [149, 193].

Consequently, Approximate Nearest Neighbor (ANN) search has been developed rapidly, where the hashing-based approaches draw considerable attention in this research field to overcome the limitations via conducting efficient retrieval in low-dimensional (i.e., compact) Hamming space [78]. The core idea of hashing is to represent the high-dimensional real-valued original data with a series of compact binary codes while preserving the semantics as much as possible during the code learning, thus accelerating the retrieval process without compromising the accuracy [193, 194]. Two advantages of hashing algorithms³ are presented as follows:

- By representing data with compact binary code, it enables efficient similarity retrieval by only conducting bitwise operation when computing the Hamming distance between data samples;
- It reduces the memory space to the maximum extent in storing the massive features and conducting the large-scale retrieval due to the nature of compact binary code [51, 165, 177, 193, 197].

These unique characteristics make hashing extremely competitive in conducting large-scale visual-related similarity search tasks. In this thesis, we will focus on learning compact binary representation via a hashing algorithm for the large-scale multimedia similarity search tasks. Particularly, we propose several new hashing frameworks and apply them in three major applications like local binary descriptors for patch-level⁴ matching and retrieval (Chapter 3), video-to-video retrieval (Chapter 4) and cross-modality retrieval (Chapter 5). The proposed methods cover various forms of the similarity search tasks, from single-modality to cross-modality, and also enable to tackle different forms of multimedia data, e.g., image, text, and video [50, 186].

³Hashing algorithm and hashing technique are used interchangeably in this thesis.

⁴Patch (i.e., small image) with small size (e.g., 64×64 , 32×32) is generated as the composition of a detected interest point and its surrounding pixels from a global image.

In the following subsections, we provide further insights on the fundamental theory of the hashing algorithm first and briefly present the motivations when proposing those novel methods. Then we summarize the contributions for each chapter. Finally, the organization of this thesis is given.

1.2 Fundamental Theory of Hashing Technique

Given a data sample represented by a feature vector $x \in \mathbb{R}^d$, the ultimate goal of hashing technique (i.e., learn to hash) is to design an optimal hash function $h(\cdot)$ that projects x from the original high-dimensional space into compact binary space $b \in \{-1, 1\}^p$ ($\mathbb{R}^d \rightarrow \mathbb{R}^p$ and $d \gg p$), while keeping its true nearest neighbors as closer as possible in the Hamming space [193, 194, 197]. In other words, the similar data samples in the original feature space should be represented with similar (low Hamming distance) binary codes, thus improving the retrieval efficiency dramatically with decent accuracy.

To achieve that learning goal, several crucial problems should be considered when designing a robust hashing framework. In this thesis, we address them concisely from five different aspects: hash function, objective function construction, optimization strategy, fast similarity search with hash code, and evaluation metrics.

1.2.1 Hash Function

As discussed above, hash function plays a crucial role in determining the retrieval accuracy in the applications of hashing technique and the code generalization efficiency [194]. Notably, we can formulate the general hashing process as follow:

$$h(x) \rightarrow b, \quad (1.2)$$

where the hash function $h(\cdot)$ can be summarized into two main types: linear and nonlinear functions. An example of the generalized linear hash function is given as below:

$$\text{sign}(\mathbf{W}x + y) \rightarrow b, \quad (1.3)$$

where $\mathbf{W} \in \mathbb{R}^{p \times d}$ and $y \in \mathbb{R}^p$ represent the linear projection matrix and bias vector, respectively. Hard thresholding function $\text{sign}(x)$ equals 1 if $x \geq 0$, otherwise -1 . However, in real applications, linear hash function usually suffers from less discriminative power though simplicity itself [111].

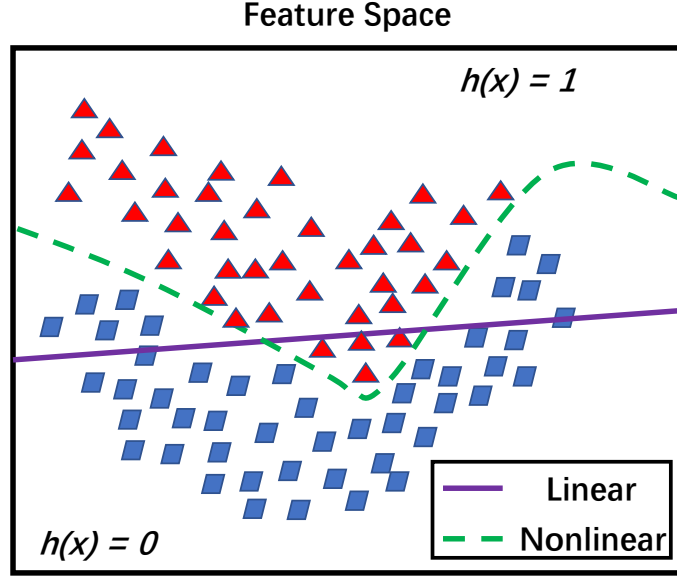


Figure 1.2: An example of linear and nonlinear models. Given two different data distributions (triangle and parallelogram), the nonlinear model can better fit the complicated data distributions than the linear one.

To overcome the discussed limitation of the linear hash function, nonlinear hash functions have been widely developed in recent studies, where kernel, spherical function, or boosting model can be applied to the original feature before the binarization. Compared to the linear hash function, such nonlinear operation enables to enhance the feature expressive ability, which is more compatible with the real data collected from complex real-world scenarios [23]. Fig. 1.2 gives a simple example in such a case. Without loss of generality, the widely-used kernel-based hash function is presented as below:

$$\text{sign}\left(\sum_i \mathbf{W}_i \phi(c_i, x) + y\right) \rightarrow b, \quad (1.4)$$

where c_i denotes the randomly-sampled data point or cluster center from the dataset. $\phi(\cdot)$ is the kernel function [91] and \mathbf{W}_i represents the weight matrix.

With the recent development of deep learning technologies, deep neural networks have been widely incorporated into the hashing frameworks, which can also be treated as an advanced form of nonlinear hash functions with various activation functions [23, 43, 94, 219]. In this thesis, deep networks are also adopted as hash functions to improve the performance of our methods.

1.2.2 Objective Function Construction

Constructing a proper objective (i.e., loss) function is the most crucial factor in obtaining better retrieval performance for a hashing framework. Generally, most existing methods build their loss functions with two essential elements: learning the hash function while keeping the similarity consistency between feature spaces at the same time [194]. For the hash function learning part, one of the most commonly adopted solutions is to minimize the loss between hash code b and hash function $h(x)$, which can be generalized as a loss term below:

$$\min \|b - h(x)\|_F^2, \quad (1.5)$$

where Frobenius norm (i.e., $\|\cdot\|_F$) is used here as an example, and the distance metrics might vary in different methods. The hash function $h(x)$ can be in the forms of linear or nonlinear, as discussed in the previous section. The underlying meaning of Eq. (1.5) is to bridge the original feature space and compact Hamming space via learning the hash function, which is also described as a projection (i.e., mapping) procedure in most related works [51, 193].

Regarding the similarity-preserving criterion, the core concept focuses on keeping the consistency between the original and binary spaces for the similar data samples during the learning process such that they are more likely to be represented by similar hash codes. In most existing hashing works, the similarity-preserving criterion can be achieved by including pairwise-, multiwise- or quantization-based solutions in the objective function [194, 197], where a simple example is provided for each solution to clarify the problem. Particularly, in the pairwise similarity-preserving criterion, the distances or similarities of a pair of data samples are aligned in the loss term. One of the most representative works is SPectral hashing (SP) [211], where the similarity-preserving term is formulated as:

$$\min \sum_{i,j} s_{ij} d_{ij} = \sum_{i,j} s_{ij} \|b_i - b_j\|^2. \quad (1.6)$$

Here, the similarity s_{ij} is calculated as $e^{-\frac{\|x_i - x_j\|^2}{\sigma^2}}$ for a pair of data points x_i and x_j . Eq. (1.6) measures the average Hamming distance between similar neighbors [211]. While in the multiwise similarity-preserving criterion, the distances or similarities of more than one pair of data points are considered, where the triplet loss is widely employed in previous methods like [137]. Given a triplet set of data points x, x^+, x^-

and their binary codes as b, b^+, b^-, x^+ is more similar to x than x^- , the triplet loss term is written as follow:

$$\max(1 - \|b - b^-\|_1 + \|b - b^+\|_1, 0). \quad (1.7)$$

According to Eq. (1.7), the loss term would be penalized when b^- is more (or equally) similar to b than b^+ , thus preserving the similarity structure during the optimization. In the quantization-based methods, the reconstruction loss function is generally employed to find the optimal approximations of original data points in the compact Hamming space. A good example of this case is Iterative Quantization (ITQ) [51] and its objective function can be formulated as below:

$$\min_{\mathbf{B}, \mathbf{R}} \|\mathbf{B} - \mathbf{R}^T \mathbf{V}\|_F^2 = \|\mathbf{B} - \mathbf{R}^T \mathbf{P}^T \mathbf{X}\|_F^2, \quad (1.8)$$

where $\mathbf{V} = \mathbf{P}^T \mathbf{X}$ denotes the projected feature after Principal Component Analysis (PCA) [213] on the centralized input \mathbf{X} . To be specific, ITQ aims at exploiting an orthogonal rotation matrix \mathbf{R} via iterative optimization such that similar data points could be aligned properly in the binary space [51]. The performance from these quantization-based methods significantly relies on the feature distribution (e.g., zero-centered) and projection quality.

In the above discussions, we have presented two essential elements of constructing a hashing objective function. It worth noting that the solutions above could be expanded as different forms but not limited to merely Eq. (1.5), (1.6), (1.7) and (1.8) [194]. Many extra problem-oriented components (e.g., regularization terms or restrict conditions) could be added to the loss function for better similarity search quality.

1.2.3 Optimization Strategy

Due to the discrete constraints introduced by the *sign* function, it is exceptionally challenging to optimize the non-convex objective functions (i.e., mixed-binary-integer optimization problem) directly in most hashing methods [194, 197]. Subsequently, various discrete optimization strategies have been proposed in previous works, and they can be roughly categorized into three mainstream types as follows.

1.2.3.1 Continuous Relaxation

During the optimization, the binary constraints are relaxed to continuous variables (i.e., approximated hash codes) via using different relaxation functions (e.g., *sigmoid* :

$x \rightarrow \frac{1}{1+e^{-x}} \in (0, 1)$ and $\tanh : x \rightarrow \frac{e^x - e^{-x}}{e^x + e^{-x}} \in (-1, 1)$ [117, 166] or even discarding the *sign* function (i.e., $\text{sign}(x) \approx x$) [118]. The purpose of performing continuous relaxation is to make the objective function differentiable such that standard gradient-based optimization schemes can be applied. However, the direct optimization over the relaxed variable cannot guarantee the optimal solution because of the gaps between approximated and real hash codes.

1.2.3.2 Alternative Optimization

In this learning paradigm, the discrete optimization procedure is decomposed into two distinct stages, namely, learn hash codes first and build hash functions afterward based on the learned codes (i.e., two-stage optimization) [105, 238]. In other words, there is no need to update the hash function during each iteration of code learning, which reduces the computational costs to some extent. For example, in [238], the binary codes are generated via exploiting the eigenvalues in a relaxed version of the Laplacian graph and then train the Support Vector Machine (SVM) classifiers as hash functions by using the learned binary codes as class labels. However, such a learning strategy usually leads to locally optimal solutions because of the separate optimization steps.

1.2.3.3 Coordinate Descent

Instead of applying relaxation strategy, the binary constraints remain unchanged during the optimization process, and the objective function is optimized iteratively via flipping each entry within hash code sequentially, which indicates that a single bit is learned based on the rest fixed bits in one binary vector [153]. However, the objective function is usually required to follow specific patterns (e.g., Binary Quadratic Programming problem [228]) or adopt a proper initialization strategy to make the coordinate-descent optimization tractable. More worriedly, the bit-by-bit optimization scheme is extremely low-efficient, which is also addressed in Chapter 5.

Generally speaking, the presented optimization strategies are all double-edged swords. Nevertheless, they provide flexible solutions when tackling such discrete optimization problems in hashing applications. Some of those strategies are also adopted in our works, which will be detailed in the following chapters.

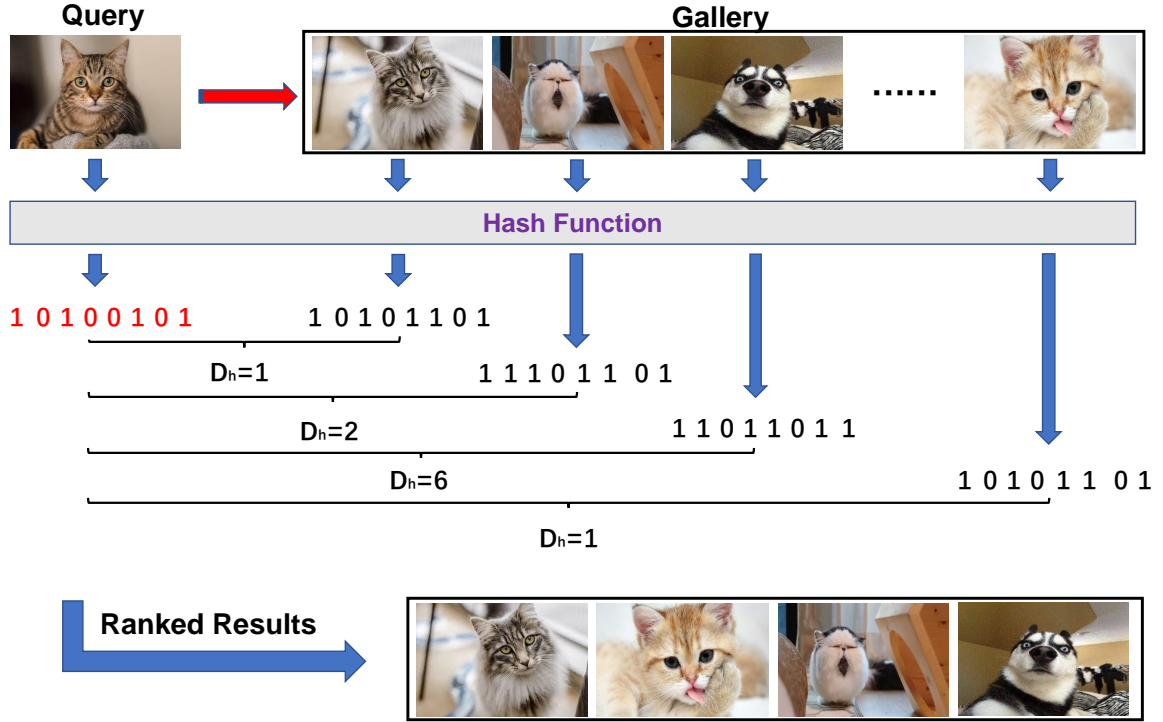


Figure 1.3: The search procedure of hash code ranking. The ranked results are obtained based on the Hamming distances (D_h) in an ascending order.

1.2.4 Fast Similarity Search with Hash Code

As long as we can obtain binary codes for the data from the proposed hash function, efficient nearest neighbor search can be conducted by using one of the following search strategies: hash code ranking and hash table lookup [194, 197].

1.2.4.1 Hash Code Ranking

As the most straightforward hashing-based search strategy, in the hash code ranking, the Hamming distances between the binary codes of query and gallery data are exhaustively calculated via bitwise XOR operation and then sorted in ascending order, where the candidates with smallest Hamming distances are returned as the nearest neighbors to that query. This search strategy exhibits one main advantage of using hash codes rather than real-valued representations, where the Hamming distance between hash codes can be computed efficiently with much lower computational costs, comparing to that under other distance metrics (e.g., Euclidean distance) in the original feature space [194]. Besides, hash code ranking adopts an exhaustive search strategy to provide coarse-level retrieved candidates for query data and more re-ranking techniques can be further applied for the fine-grained retrieval results. A

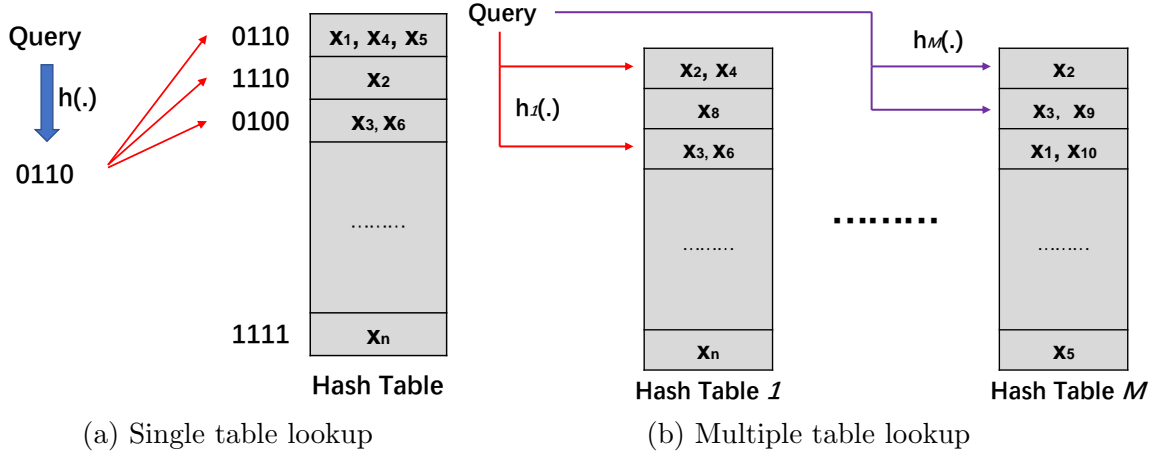


Figure 1.4: (a) 4-bit hash code is used as an example. The returned items would be x_1, x_4, x_5, x_2, x_3 and x_6 if the Hamming radius is 1; (b) $M(M > 1)$ hash tables are constructed in the multiple table lookup and the neighbor search is conducted across these tables afterward.

simple example of the hash code ranking is presented in Fig. 1.3 and it has been widely adopted to evaluate the system performance of our methods and the state-of-the-arts.

1.2.4.2 Hash Table Lookup

Theoretically, the search process of the hash table (i.e., hash map) lookup is more complicated than the hash code ranking. Precisely, the hash table consists of multiple buckets, where each bucket is indexed by unique hash code and assigned with at least one item (i.e., data sample) from the gallery under the perfect condition. One fundamental principle in constructing such a hash table is maximizing the probability of similar items being stored in the same bucket and vice versa. It speeds up the search process via reducing the frequency of distance computations in the inverse lookup when given a query data [22, 194].

There are two different types: single table and multiple tables, in the hash table lookup. In single table lookup, all gallery items (e.g., x_1, x_2, \dots, x_n in Fig. 1.4(a)) are placed in one table and the query needs to visit more buckets to guarantee the retrieval accuracy. In multiple tables lookup, a bunch of tables (e.g., M in Fig. 1.4(b)) are constructed to store multiple copies of those gallery items. By placing the item copies in multiple hash tables, it obtains satisfactory performance by improving the possible hit number of each relevant item though relatively high space costs. Fig. 1.4 briefly presents the hash table indexing procedures. The hash table lookup is

widely employed to evaluate the retrieval performance of those LSH-based methods and more details on hash table lookup can be found in [160, 194, 197, 211].

1.2.5 Evaluation Metrics

Several popular performance evaluation metrics used to measure the search quality are elaborated in this section, including Precision@K, Mean Average Precision, Precision-Recall Curve and Receiver Operating Characteristic Curve.

1.2.5.1 Precision@K

Precision at top-K retrieved candidates (Precision@K) is one of the most fundamental performance evaluation metrics in large-scale information retrieval. In this thesis, Precision@K can be computed as:

$$Precision@K = \frac{|\{Relevant\ Data\} \cap \{Top\ K\ Retrieved\ Data\}|}{K}, \quad (1.9)$$

where $|\cdot|$ denotes the size of the common subset. This metric has been used in the experiments of Chapter 3, Chapter 4 and Chapter 5.

1.2.5.2 Mean Average Precision

In the information retrieval, mean Average Precision (mAP)⁵ is a crucial retrieval performance metric. To calculate the mAP score, the average precision of each query should be computed first during the retrieval. To be specific, the average precision on the retrieved results for one query is calculated as:

$$Average\ Precision = \frac{\sum_{k=1}^{N_c} (Precision@k \times Rel(k))}{|\{relevant\ data\}|}. \quad (1.10)$$

where N_c denotes the retrieved candidates number. $Rel(k)$ refers to an indicator function, which equals 1 if the relevant item to the query is retrieved at rank k and otherwise 0 [185]. Therefore, the mAP value for all the queries during the retrieval can be calculated as below:

$$mAP = \frac{\sum_{i=1}^{N_q} Average\ Precision(q_i)}{N_q}, \quad (1.11)$$

where N_q is the queries number. This metric has been used in the experiments of Chapter 3, Chapter 4 and Chapter 5.

⁵Sometimes it could be Mean Average Precision (MAP).

1.2.5.3 Precision-Recall Curve

In information retrieval, precision is a measure of result relevancy, while recall is a measure of how many genuinely relevant results are returned. The Precision-Recall (PR) curve shows the tradeoff between precision and recall for different thresholds. A high area under the curve represents both high recall and high precision, where high precision relates to a low false-positive rate, and high recall refers to a flat false-negative rate. High scores of both metrics show that the classifier is returning accurate results (high precision), as well as returning a majority of all positive results (high recall).

Particularly, precision is defined as the number of true positives (T_p) over the number of true positives plus the number of false positives (F_p), which is formulated as:

$$Precision = \frac{T_p}{T_p + F_p}. \quad (1.12)$$

Recall is defined as the number of true positives (T_p) over the number of true positives plus the number of false negatives (F_n), which is computed as:

$$Recall = \frac{T_p}{T_p + F_n}. \quad (1.13)$$

This metric has been used in the experiments of Chapter 3, Chapter 4 and Chapter 5.

1.2.5.4 Receiver Operating Characteristic Curve

Receiver Operating Characteristic (ROC) curve is determined by True Positive Rate (TPR) and False Positive Rate (FPR). To be specific, TPR is defined as the number of true positives (T_p) over the number of true positives (T_p) plus the number of false negatives (F_n), which is formulated as:

$$TPR = \frac{T_p}{T_p + F_n}. \quad (1.14)$$

FPR is defined as the number of false positives (F_p) over the number of true negatives (T_n) plus the number of false positives (F_p), which is calculated as:

$$FPR = \frac{F_p}{T_n + F_p}. \quad (1.15)$$

This metric has been used in the experiments of Chapter 3.

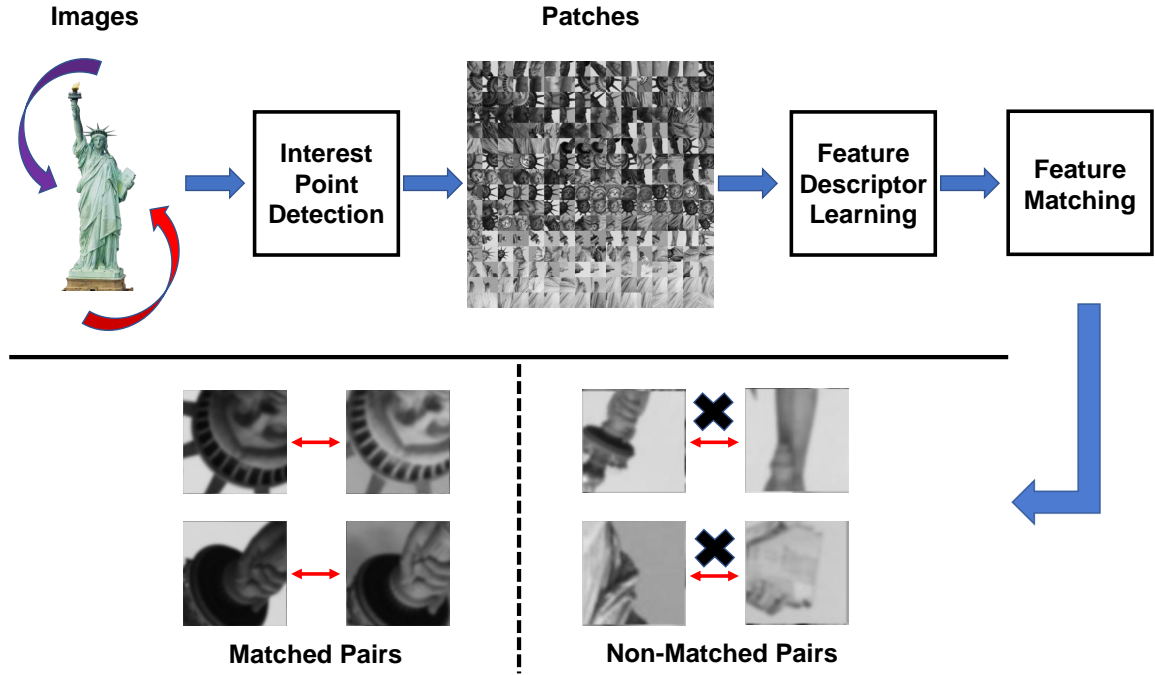


Figure 1.5: The work pipeline of traditional image matching using local feature descriptor, which consists of three major steps: interest point detection, feature descriptor learning and feature matching. The feature learning and matching processes are conducted at the patch level. The input images might be photographed from different views or even processed with various affine transformations.

In the next section, we will present the concrete research problems and challenges in designing the hashing algorithms for various applications in this thesis.

1.3 Research Problems and Challenges

Although considerable success has been achieved by the modern hashing methods in a wide range of various similarity search tasks, there are still several limitations that need to be addressed for better system performance. In this thesis, we mainly focus on three popular hashing applications: image matching/retrieval with the local binary descriptor, video hashing, and cross-modality hashing, where the existing challenges that motivate our works in these applications are elaborated.

1.3.1 Local Binary Descriptor

Fig. 1.5 shows the traditional image matching process by using a local feature descriptor. To be specific, it first generates local patch sets for each image by the interest point detectors, such as Scale Invariant Transform Invariant (SIFT) [121] and

Hessian-Affine [129], and learns the corresponding local feature descriptor for each patch. Then the feature matching can be conducted by measuring the distance (e.g., Euclidean distance) between those descriptors. Especially, matching can be treated as a particular case of retrieval, where only one exact candidate is returned for the query in the matching process. The fundamental principle of designing local feature descriptors is that the descriptor should be robust against various geometric transformations like rotation, translation, and scaling, etc. [97, 121]. In the earlier works, they mainly focus on learning the real-valued feature descriptors for the matching process. Hundreds of interest points per image would be selected and the high-dimensional feature vectors for each patch are usually required, which leads to high computation complexity and memory costs in the upcoming matching process. Consequently, the binary descriptor has drawn extensive attention to accelerate the matching process while consuming low computing resources. In this thesis, we discuss several limitations of the existing binary descriptors below.

- Earlier binary descriptors (e.g., Binary Robust Independent Elementary Features (BRIEF) [17], Binary Robust Invariant Scalable Keypoints (BRISK) [97]) generally adopt shallow handcrafted sampling patterns and perform pairwise intensity comparisons to generate the feature descriptors [231]. For example, in BRIEF, a bunch of specific location pairs are selected from the smoothened image patch and pixel-level intensity comparison is conducted between each pair [17]. However, such handcrafted descriptors are extremely vulnerable to the distortions/transformations, which yields unstable performance when tackling large-scale visual recognition tasks [182, 183, 231].
- The learning-based binary descriptor is becoming a popular research topic for its desirable performance against those handcrafted ones. Most previous methods adopt hashing ideas and pay intense attention to novel discrete optimization strategies, however, the basic principle in designing local feature descriptors, anti-geometric transformation, is not considered during the optimization [97, 183]. Moreover, most learning paradigms fail to preserve the manifold structure during the discrete optimization, which makes binary descriptor less effective in the large-scale nearest neighbor search tasks [56, 107, 153].
- Traditional binary descriptors measure the similarity between database and query via performing exhaustive Hamming distance calculations in the testing phase, which is more likely to return many candidates with equal Hamming

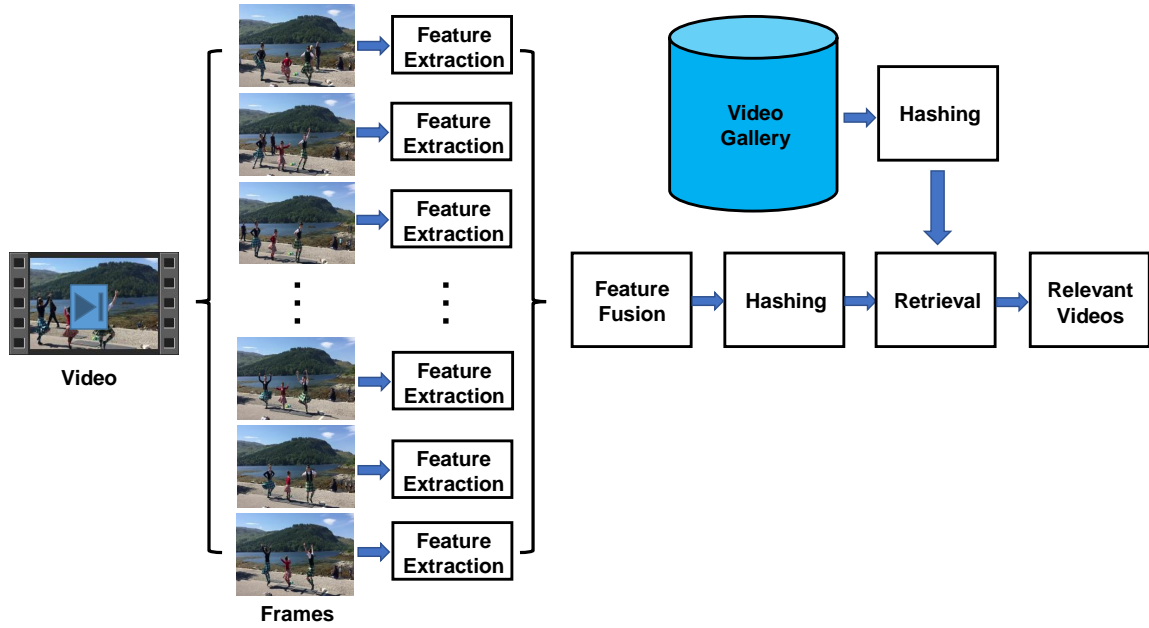


Figure 1.6: The general pipeline of video retrieval.

distances to one specific query compared to using the real-valued feature descriptors [120]. However, the ultimate goal of the feature matching is to find one exact candidate (with the lowest Hamming distance) to a specific query rather than returning a bunch of ambiguous options (with equal minimum Hamming distance) in the general retrieval process.

Those defects give rise to the less favorable performance from the previous binary descriptors and should be carefully addressed in the descriptor learning to improve the matching accuracy.

1.3.2 Video Hashing

Fig. 1.6 presents the general pipeline of video retrieval based on the hashing algorithm. Compared with images, videos, which can be treated as more complicated 3D signals, are more difficult to be hashed because of variable lengths and tremendous redundancies. To tackle those issues, video is firstly represented as a certain number of consecutive keyframes prior to employing various hashing algorithms, as shown in Fig. 1.6 [243]. However, two major limitations in previous works impede their performance in large-scale video retrieval, which are presented as follows:

- In earlier video hashing methods [28,38,172], they generally wrap the keyframe-level features up as a single video feature first via employing various feature

fusion strategies, and then apply the binarization tactics from image hashing [51, 211] on the concentrated feature directly. Some of them even perform image hashing on each frame independently and generate video hash code by thresholding the average value of all frame-level binary codes [24]. However, such learning paradigms degrade the quality of video hash code significantly because of the improper hashing mechanisms and the ignorance of the video's temporal nature [109, 197, 200, 225].

- As revealed in recent studies on video representation learning, it is more likely to obtain better video features via incorporating the frame-level spatial information with the temporal information of a video sequence, thus improving the system performance in various vision tasks [175, 226, 235]. Currently, two mainstream ways to obtain such video feature are presented as follows: 1) feeding the sparsely-sampled frame-level image features from a video clip into Recurrent Neural Network (RNN) or Long-Short Term Memory (LSTM) [73] in order to explore the temporal nature and then aggregating into the global video-level feature [109, 197, 217]; 2) fusing the spatial and temporal features (e.g., optical flow) from the two-stream networks to generate the unified video representation via various pooling schemes [200]. However, those operations usually make the synthetic video feature with more scattered (less correlated) distribution and unbalanced dimensions compared to the pure image feature [86, 175]. When projecting these video features into the low-dimensional compact space before the binarization step, the data variances on projected dimensions tend to be large (i.e., unbalanced) [35, 57, 58]. Especially for the fixed bit quantization, this imbalanced projection will degrade the performance of the generated hash codes because each dimension will be treated equivalently and allocated with the same number of bits (e.g., 1 bit in most existing works) in the quantization step afterward. It is unable to achieve effective hash codes for video retrieval due to such an unfair bit assignment scheme.

1.3.3 Cross-Modality Hashing

Fig. 1.7 depicts the general process of cross-modality hashing, where the pairs of image and text are utilized in the training phase and the cross-modal retrieval tasks: image \rightarrow text and text \rightarrow image⁶, are performed afterward. Compared with the single-

⁶Image \rightleftharpoons text is used as an example here. Other data types (e.g., audio and video) can also be applied.

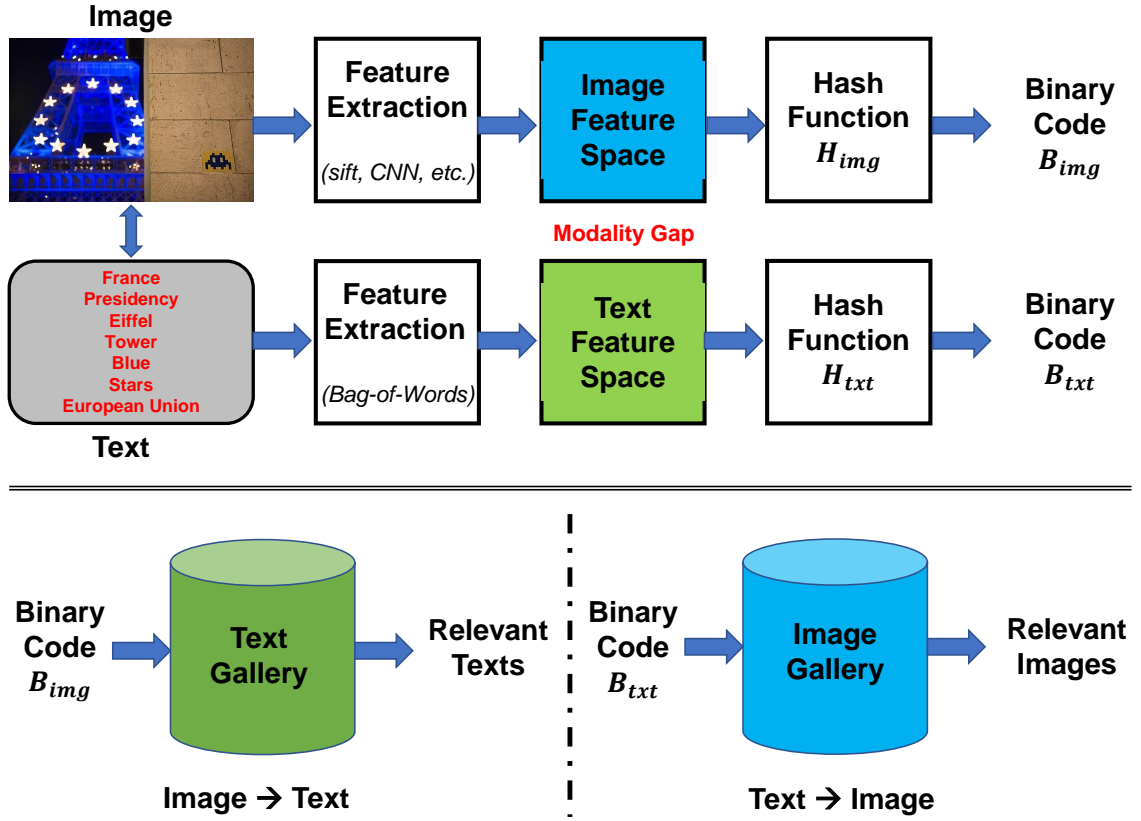


Figure 1.7: The general workflow of cross-modality retrieval. Two cross-modal retrieval tasks: image \rightarrow text and text \rightarrow image, are used as examples.

modality retrieval (e.g., video-to-video retrieval discussed in the previous section), the data instances are not in the same feature space when conducting the cross-modality similarity search. One of the main challenges in this research field is how to tackle the semantic gaps within different modalities (i.e., modality gap) properly. Many types of research have been devoted to handling this problem, however, some defects from these methods become the hard barriers of retrieval performance gains.

- Most existing methods, both in unsupervised [35, 170, 190] and supervised [10, 108, 180, 223] manners, concentrate on exploring a common latent space for the data from various modalities during the training process such that the heterogeneity among modalities can be minimized [35, 108]. For instance, as a classical work in the common latent space learning, matrix factorization is adopted in [34] to model the relations among different modalities. Specifically, a two-stage learning strategy is applied in learning unified binary code, where the original features of data points from different modalities are first projected into a real-valued common latent space, and then the latent space is quantized

to obtain the binary code. Recently, deep learning technology has been widely incorporated in the cross-media hashing, which improves retrieval performance with better feature representation and nonlinearity modeling ability [210]. However, most of those deep-based methods still employ a two-step scheme in the code learning following [34], where large quantization errors exist in the binarization step [189]. That is to say, the two-stage learning paradigm yields large quantization errors, and such errors will be further magnified after the iterative code learning, thus leading to suboptimal results because of the bad code quality.

- Moreover, most previous works focus on learning the unified representation for the multimodal inputs. However, neighborhood structures from the original feature space are not well preserved, thus compromising the retrieval performance significantly because of less discriminative code. In another word, only instances within data pairs (e.g., a pair of image and its description) are represented with the same binary code in the simple common latent space learning. While the positions between data pairs cannot be guaranteed in the binary space after the projection. Compared to unsupervised methods, supervised cross-modal hashing addresses this issue with the aid of dedicated prior knowledge (e.g., semantic labels, similarity matrix) [199]. Nevertheless, such similarity preservation is usually performed on approximated (i.e., real-valued) hash codes without any restrictions on binary codes in the training process, where the gap between real-valued and binary spaces makes that operation less effective [108, 126, 202, 204, 223].

Based on the discussions above, the retrieval performance would be profoundly affected by those drawbacks, thus preventing the existing methods from massive deployment in real-world cross-media similarity search applications.

In this section, we have discussed the significant research challenges concisely in various hashing-based applications. To handle those problems above, several novel solutions are proposed for those applications separately, which will be further detailed in the main chapters. In the next subsections, we briefly summarize the contributions of each work and provide the chapter outlines correspondingly.

1.4 Overview of Contributions

In this thesis, the content covers the research challenges of the compact binary code learning based on hashing algorithm in three popular applications: ranging from local binary descriptor to video hashing and finally to cross-modal hashing. The contributions of this thesis can be presented as proposing novel solutions to tackle the discussed research challenges while achieving superior similarity search performance in the above applications. Without loss of generality, the contributions of each work are briefly summarized as follows:

- **Chapter 3:** To address the challenges above in Section 1.3.1, a novel method termed Unsupervised Deep Binary Descriptor (UDBD) is proposed to learn the transformation invariant binary descriptor, where the original visual data and their transformed sets are being embedded into a common Hamming space in an unsupervised manner. Moreover, a graph constraint that preserves the manifold structure from the original feature space is employed in the unified binary representation learning, thus improving the code quality. Since patch mainly contains noise-sensitive local features, $\ell_{2,1}$ -norm loss is proposed to regularize the binary embedding. On one hand, ℓ_1 -norm distance at the patch level provides the robustness against outlier samples. On the other hand, ℓ_2 -norm measures the distance along space dimension, which spreads out the errors over each bit uniformly to lower the possibility that certain bits are mistakenly flipped after getting significant errors. To this end, an alternating discrete optimization strategy is proposed to optimize the $\ell_{2,1}$ -norm constrained objective function, where the binary code can be solved directly with no need for relaxation. Additionally, a weak bit scheme, which considers the reliability of each bit in a descriptor, is applied along with the proposed binary descriptor to find the best match when there are multiple candidates with the same distance to the query after comparing the Hamming distance of descriptors.
- **Chapter 4:** To boost the large-scale video retrieval performance, a novel unsupervised deep hash framework termed Unsupervised Deep Video Hashing (UDVH) is proposed to organize hash code learning in a self-taught manner. Instead of minimizing feature reconstruction distortion, our structure minimizes the quantization error of projecting video features to a binary hypercube, thus allowing the feature extraction and hash function learning to engage with each

other. Involving the feature clustering in the code learning enables the neighborhood structure to be preserved. Specifically, for the unbalanced dimensions of video features as discussed above, a balanced rotation scheme is proposed to identify a proper projection matrix such that the variance of each projected dimension can be balanced. By doing so, the information in each dimension of video features can be equalized. This operation would greatly benefit the quantization step, in which each dimension is allocated with the same number of bits. This binarization scheme taking imbalanced properties of video feature into account improves the video retrieval performance dramatically. To provide comprehensive insights on the proposed framework, two different learning structures: stacked LSTM units (UDVH-LSTM) and Temporal Segment Networks (UDVH-TSN) are discussed and analyzed.

- **Chapter 5:** To tackle the problems in Section 1.3.3, a matrix factorization based supervised cross-modal hashing method termed Self-Supervised Deep Multimodal Hashing (SSDMH) is proposed, which incorporates deep feature learning, binarization and deep hash function learning seamlessly into a unified learning framework. Notably, a novel regularized *binary* latent model is proposed during the code learning, where the unified binary code is learned via projecting the original features from various modalities into a common binary space. Moreover, the discrete unified binary codes can be solved without relaxation and the weights of different modalities are optimized dynamically. To make the most advantage of supervision knowledge, we propose to minimize the graph regularization loss, which explicitly preserves the neighborhood structures of the original data and is prone to produce the discriminative hash codes. An alternating optimization strategy is adopted in solving the discrete-constrained objective function, where deep parameters and unified binary codes are optimized jointly. Particularly, a novel discrete optimization method, termed as *Binary Gradient Descent*, is proposed to accelerate the optimization speed dramatically, in contrast to the traditional bit-by-bit fashions. Besides, the proposed algorithm is further extended to an unsupervised version termed Unsupervised Deep Cross-Modal Hashing (UDCMH), which is suitable to be applied in the large-scale training process with no prior knowledge is available.

1.5 Thesis Outline

The rest of this thesis consists of a comprehensive literature review on the existing works and the proposed state-of-the-art hashing frameworks for three popular applications of similarity search. Firstly, we propose to learn the robust local binary descriptor against arbitrary geometric transformations via using matrix factorization for effective image matching and retrieval tasks. Continuing the topic in the single-modality domain, we present an unsupervised deep video hashing model for general video-to-video retrieval. Then we extend the research topic to the cross-modality (i.e., multi-modality) domain and propose a novel deep cross-modal hashing network. Finally, we conclude this thesis and discuss potential research directions. The remaining chapters are summarized as follows:

Chapter 2: Literature Review on Hashing-Based Similarity Search. A comprehensive overview of the state-of-the-art hashing-based retrieval algorithms is given in this chapter, which covers the research works from various domains in different similarity search applications.

Chapter 3: Unsupervised Deep Binary Descriptor. In this chapter, we propose a novel learning-based feature descriptor, namely Unsupervised Deep Binary Descriptor (UDBD), which learns transformation invariant binary descriptors without relaxation via projecting the original data and their transformed sets into common binary space directly. The proposed method can also be applied in the similarity search tasks like image matching and retrieval. Extensive experimental results on public datasets: Brown [11], Cifar-10 [89], and HPatches [3] show the superiority of UDBD in terms of matching and retrieval accuracy over the state-of-the-arts.

Chapter 4: Unsupervised Deep Video Hashing. In this chapter, we propose a novel video hashing framework named Unsupervised Deep Video Hashing (UDVH) for large-scale video similarity search tasks to learn compact yet effective binary codes. Two different video feature learning structures: stacked LSTM units (UDVH-LSTM) and Temporal Segment Networks (UDVH-TSN) are integrated into the proposed video hashing framework. Particularly, a smart rotation applied to the video-specific features that are widely spread in the low-dimensional space such that the variance of dimensions can be balanced, thus facilitating the subsequent quantization step. Extensive experiments on three popular video datasets: FCVID [140], YFCC [181], and ActivityNet [14], show that UDVH is overwhelmingly better than the state-of-the-arts in terms of various evaluation metrics.

Chapter 5: Deep Cross-Modal Hashing. In this chapter, we propose a novel hashing method termed Self-Supervised Deep Multimodal Hashing (SSDMH), for large-scale cross-media search. Notably, the hashing system based on the *binary* latent factor models can generate unified binary codes by solving a discrete-constrained objective function directly with no need for relaxation. Moreover, we propose a new discrete optimization solution, termed as *Binary Gradient Descent*, which aims at improving the optimization efficiency towards the real-time operation. Besides, an unsupervised version of the proposed algorithm, termed Unsupervised Deep Cross-Modal Hashing (UDCMH), is introduced to tackle the cross-modal retrieval applications without supervision information. Extensive experiments on three benchmark datasets: Wiki [146], MIRFlickr [77], and NUS-WIDE [25], demonstrate the superiority of the proposed methods over state-of-the-art cross-modal hashing approaches.

Chapter 6: Conclusion and Future Work. Finally, we summarize the contributions of this thesis in this chapter and present future research interests.

Chapter 2

Literature Review on Hashing-Based Similarity Search

In this chapter, a comprehensive review of previous hashing-based similarity retrieval methods from both single-modality and cross-modality is presented. Notably, in this thesis, local binary descriptor, image, and video hashing are concluded as three sub-topics of the single-modality similarity search in Section 2.1. Without loss of generality, the literature review starts from the traditional image hashing methods, which are the fundamentals in the research field. The state-of-the-art local binary descriptors, where most of them adopt the hashing idea in descriptor learning, are categorized and discussed as a single-modality retrieval problem in Section 2.1.2. Then the review topic is changed to the video hashing applications, which can be treated as an extension of image hashing. Finally, state-of-the-art cross-modal hashing methods are reviewed in Section 2.2, which motivate our work to address the problems as mentioned above.

2.1 Single-Modality Similarity Search

In single-modality hashing, the hash function is learned based on one specific type of data (e.g., image, text, video), and the retrieval is then conducted between the data samples under the same type. Generally, it covers the methods from the traditional image and video hashing, where they can be either supervised or unsupervised for each of them. The difference between supervised and unsupervised hashing is that whether the supervision knowledge is used or not [193]. Particularly, in supervised hashing [55, 91, 113, 114, 117, 136, 196], the label or pairwise similarity are adopted in the training process, which usually yields better retrieval performance than the unsupervised methods under the same settings. However, the labor-extensive labeling

process is required in supervised hashing, which makes it less favorable in conducting large-scale similarity retrieval. While unsupervised hashing [51, 69, 88, 115, 152, 154, 211] builds the hash function via exploring the data structure (e.g., clustering) with no need for supervision information, which is also the focus of this thesis. In this section, we first provide a quick review of image hashing and then discuss the related works in the local binary descriptor and video hashing to echo the contents in Chapter 3 and Chapter 4.

2.1.1 Image Hashing

Traditional image hashing methods can be generally classified into two domains: data-independent and data-dependent. A representative data-independent method termed Locality Sensitive Hashing (LSH) [22] produces the compact binary codes by using random projections on original high-dimensional features along with thresholding schemes. Then an extended version of LSH named Kernelized Locality Sensitive Hashing (KLSH) [92] is proposed, which builds the hash functions with the kernelized pairwise representations. However, these methods that are adopting a random projection paradigm usually yield unappreciated performance when using short codes, which makes them unsuitable for hashing applications that require compact binary codes.

Consequently, data-dependent methods are widely developed, where advanced statistical learning strategies are utilized in building the hashing functions to produce compact yet effective binary codes. As mentioned above, those hashing methods can be further divided into supervised and unsupervised ones. In supervised hashing, dedicated prior knowledge is involved in the code learning. For example, Binary Reconstructive Embedding (BRE) [91] builds the kernel hash function via minimizing the squared reconstruction errors between the original and the Hamming distances of the to-be-learned binary embeddings. However, the proposed coordinate descent optimization is not efficient and the retrieval performance is not satisfactory. Kernel Supervised Hashing (KSH) [117] approximates the Hamming distances between data samples by utilizing the pairwise similarity calculating from category information. The drawback of KSH is that the intra-class distance is not optimized in the training process, which lowers the binary code quality dramatically when dealing with various categories within one dataset. In [177], Linear Discriminant Analysis hashing (LDA-Hash) is proposed to alleviate this issue by minimizing the intra-class and maximize the inter-class variations of binary codes simultaneously. Shen et al. [153] propose Supervised Discrete Hashing (SDH) that solves the binary codes via applying Discrete

Cyclic Coordinate descent (DCC) method, where the label information is used to approximate the binary code via linear classifiers. However, Support Vector Machine (SVM) [27] is trained during each iteration to generate auxiliary variables, which is not efficient when tackling large-scale datasets.

Recently, breakthrough performance has been achieved by the combination of deep learning techniques and hashing in the large-scale image retrieval tasks [43, 94, 219], where deep Convolutional Neural Network (CNN) [7, 90] is widely deployed in those works. In [219], Convolution Neural Network Hashing (CNNH) is proposed to decompose the hash function learning into two stages: hash code learning and deep network fine-tuning, where the similarity between data pairs is preserved by minimizing the loss between the inner product of binary code and the similarity matrix. While Deep Neural Network Hashing (DNNH) in [94], which can be treated as an updated version of CNNH, outperforms the former framework by jointly conducting in-depth feature and hash code learning within the deep structure. Despite many supervised deep hashing frameworks [101, 111, 192, 198, 248] have been proposed for better retrieval performance, they generally incorporate either labels or pairwise/triplet similarity in the training process and obtaining such supervision information is usually label-extensive and expensive, which are not appreciated in the real-world applications.

In unsupervised hashing, the hash function is formulated and built on the training of unlabeled data. Some representative methods are briefly introduced here. For instance, SPectral hashing (SP) [211] is proposed to build the hash function via solving the eigenfunctions, where the smallest eigenvalues are thresholded to zero to obtain the binary codes. However, SP works under the assumption that the data structure follows a uniform distribution, which impedes it from the complex applications that involve uncertain distributions. In [71], Spherical Hashing (SpH) learns a series of spherical functions and the binary codes are generated by quantizing the distances between the original feature representations and their corresponding centers. The limitation of SpH is that the optimization is not performed in the binary space, which yields low code quality. Moreover, the iterative center learning is computationally expensive and time-consuming. Iterative Quantization (ITQ) [51] is proposed to learn the binary codes via minimizing the quantization errors between the target codes and the product of their original representations and an orthogonal rotation matrix. Principal Component Analysis (PCA) [213] or Canonical Correlation Analysis (CCA) [67] can be applied to reduce the high dimensionality of the original feature before the discrete optimization. While an updated version of ITQ is proposed in [57] termed ITQ+ with both robustness and generalization capability enhanced by using

a ℓ_p -norm distance in the discrete optimization. Some related works of ITQ are also proposed: Isotropic Hashing (IsoH) [88] and Harmonious Hashing (HamH) [221]. To be specific, IsoH utilizes a rotation to balance the variance by minimizing the reconstruction error of the covariance matrix and a diagonal matrix, while HamH minimizes the distance between the rotated data and a perfectly balanced matrix. However, the reconstruction on a small covariance matrix in IsoH is unstable in large-scale and high-dimensional experiments. The strict requirements of HamH and its non-iterative optimization algorithm may fail to find a good solution. Consequently, Liu et al. [118] propose Anchor Graph Hashing (AGH) for the large-scale image retrieval, where a small number of anchors are learned to represent the similarities between data pairs in the training set, thus preserving the neighborhood structure inherently in an efficient way. Discrete Graph Hashing (DGH) [116] is further developed to improve retrieval performance, where the hash codes are directly optimized in such graph-based hashing. The idea of using anchor points is also introduced in K-Means hashing (KMH) [69], which generates effective hash codes via minimizing the Hamming distances between anchors after binarization. However, the hash code quality from both the aforementioned methods heavily relies on the anchor point selection, where the selection process is sometimes arbitrary and computationally expensive in most cases.

Consistent with supervised hashing, deep neural networks have been further involved in the recent algorithms of unsupervised hashing, which enables to handle the data distributions with the nonlinearity nature in the real-world application scenarios [193]. In [43], Deep Hashing (DH) is developed to learn the hash function with multiple hierarchical nonlinear transformations, where independent bits in binary codes with even distribution can be achieved. Subsequently, an unsupervised learning-based deep hashing framework is proposed in [122] that trains the deep neural network by minimizing the compact real-valued codes and their binary codes. They further extend this work to supervised deep hashing and multi-label supervised deep hashing, which aims at generating more discriminative binary codes with the aid of supervision information. In [23], Nonlinear Discrete Hashing (NDH) is proposed for scalable image search, where the binary codes are optimized with discrete quantization and the reconstruction errors are minimized between the learned binary codes and the original data, respectively. The above deep-based methods generally adopt shallow deep networks as the backbone in the hash code learning, which weakens the feature representation ability to some extent. Consequently, more advanced deep networks like Generative Adversarial Network (GAN) [52] have been incorporated in the

hashing framework, such as HashGAN [49], which improves the retrieval performance significantly because of the more powerful deep model.

2.1.2 Local Feature Descriptor

Local feature descriptor plays a vital role in many computer vision tasks such as object recognition, image matching, 3D reconstruction and image retrieval [12, 68, 121, 179]. In fact, the appearance of original data (i.e., patch) is easily affected by many factors like variations of lighting, scaling, as well as geometric transformations, the local feature descriptor is supposed to represent the data accurately despite those external factors [97, 121].

2.1.2.1 Handcrafted Feature Descriptors

Most handcrafted local descriptors are real-valued in the early research stage. Two classical feature descriptors: Scale Invariant Fourier Transform (SIFT) [121] and Speeded Up Robust Features (SURF) [6] are widely used in the visual recognition tasks like image retrieval and feature matching. Particularly, the local gradient histograms are applied in SIFT to generate the scale-invariant descriptors. The computation process of SIFT is accelerated dramatically by SURF, which takes advantage of the integral images in the calculations. However, the excellent performance from both of these real-valued feature descriptors heavily relies on the high dimensionality (i.e., long descriptor length), which leads to high storage requirement and computational complexities for the similarity search tasks when using those descriptors [1, 17].

Consequently, many efforts have been devoted to developing local binary descriptor to address those problems above, such as Local Binary Pattern (LBP) [138], Binary Robust Independent Elementary Features (BRIEF) [17], Oriented fast and Rotated BRIEF (ORB) [150], Binary Robust Invariant Scalable Keypoints (BRISK) [97], and Fast Retina Keypoint (FREAK) [1]. These descriptors generally perform a set of pairwise intensity comparisons to generate compact binary codes. While the efficiency of these binary descriptors for the similarity search tasks has been improved significantly because of the XOR operations in Hamming space, their robustness is relatively worse than that of the real-valued local descriptors. The reason is that these binary descriptors are mainly built according to some manually predefined sampling modes and shallow pixel-level intensity comparisons, which are very sensitive to the affine transformations and quality variations on the original image/patch, thus compromising their performance when dealing with complex visual tasks [182, 183, 231].

Recently, a novel binary RGB-D descriptor termed GEOBIT is presented in [133] for the textured depth map tracking, where the binary descriptor is claimed to be invariant to the non-rigid transformation by integrating the appearance and the geometric information from RGB-D images in the code learning. Superpixel Region Binary Descriptor (SRBD) [227] proposes a new kernel-distance-based clustering method to select the stable superpixels from the templates and encodes the dominant gradient orientation of each superpixel as its rotation-invariant binary descriptor. The local binary descriptor also has been applied in separating the Computed Tomographic (CT) medical images, where Local Diagonal Laplacian Pattern (LDLP) [232] applies the second-order derivatives to model the relationship between the center pixel and its diagonal elements in generating the binary descriptor. However, they still adopt handcrafted patterns like BRIEF [17] and ORB [150], which indicate their weak generalization ability.

2.1.2.2 Learning-Based Feature Descriptors

More recently, the learning-based feature descriptors, which involve a dedicated training process of encoding function on massive training data, are widely developed to boost the descriptor performance and gain better robustness.

Earlier learning-based works learn the shallow projections to obtain the local descriptors. For example, Linear Discriminant Analysis Hashing (LDAHash) [177] is proposed that uses linear projections combining linear discriminant analysis to generate binary descriptors. Discriminative BRIEF (D-BRIEF) [184] produces the descriptors by projecting the training data into a latent subspace. To deal with the nonlinear data structure, BinBoost [183] learn a set of nonlinear classifiers in encoding the data, which makes the learned binary codes more discriminative with applying the boosting algorithm jointly. Online learning is adopted in Binary Online Learned Descriptor (BOLD) [4], which aims at selecting binary intensity tests to produce low intra-class and high inter-class distances in the code learning. However, these methods generally adopt simple binary intensity tests and some critical cues of a patch cannot be captured in the to-be-learned descriptor. Subsequently, Coupled Compact Binary Face Descriptor (C-CBFD) [124] is proposed to generate binary codes under three complementary learning objectives: high variance for information preservation, low quantization errors and even-distribution at each bit. A one-stage learning strategy is utilized in Simultaneous Local Binary Feature Learning and Encoding (SLBFLE) [123], where the binary codes and the encoding codebook are jointly optimized for local face patches. Consequently, they extend these works as Context-Aware

Local Binary Feature Learning (CA-LBFL) [40] and Rotation-Invariant Local Binary Descriptor (RI-LBD) [39], which learns the robust local binary descriptor further to improve the efficiency and accuracy in face recognition.

With the development of deep learning techniques, more recent works apply CNN network and deep features in learning the local feature descriptor. For example, Dosovitskiy et al. [37] train a CNN network by optimizing the classification loss, where the output vectors before the classification layer are used as the patch descriptors. Particularly, data augmentation is applied to the training data to avoid overfitting for the upcoming classification process, where the augmented patches are generated by adding some random variations/noises. Instead of merely optimizing the classification loss, Siamese loss is introduced in the network training of DeepDesc [162], where the patch pairs as the network inputs are selected by applying an aggressive searching strategy. A central-surround two-stream network structure is utilized in [237] to improve the matching performance of the learned feature descriptor, where the center of a patch is used as input and the similarity between patch pairs is computed through a Siamese network. HardNet [130] proposes a triplet loss function that explores the hard examples by an effective mining strategy to mimic the matching procedure in a batch fashion, where at least one positive pair is guaranteed in building the triplet input. Descriptors Optimized for Average Precision (DOAP) [70] is proposed to train the deep network via optimizing a new loss function termed Average Precision (AP) directly, which improves the ranking-based retrieval performance. Wei et al. [209] introduce a novel pooling method termed Subspace Pooling in the code learning, which is claimed to obtain the robustness against a range of geometric deformations for the learned feature descriptor.

More works have been done recently to learn the binary descriptor from deep-based frameworks. For example, Deep Hashing (DH) [43] optimizes the binary descriptor with independence and even distribution, while Deep Supervised Hashing (DSH) [111] optimizes distance loss and Siamese loss jointly to improve the binary descriptor quality. Subsequently, L2-NET [182] trains a Siamese network for pairwise patches and produces binary codes by directly quantizing the real-valued outputs, where different regularization terms are applied to the intermediate layer outputs to improve the code quality. More than just pairwise inputs, the triplet loss is incorporated in the objective function of [233] to further guarantee the code discriminativeness. Deep Binary Descriptor with Multi-Quantization (DBD-MQ) [41] adopts a multi-quantization strategy that reduces the quantization errors within the K-AutoEncoders (KAEs) networks. GraphBit [42] integrates the reinforcement learning with binary

code learning, where the uncertainty of binary codes is minimized by maximizing the mutual information between the real-valued inputs and the corresponding bits. With the mighty Generative Adversarial Network (GAN) [52], BinGAN [252] learns the compact binary descriptors from patches via optimizing two additional losses from distance matching and entropy regularizers. GAN has also been employed in [169] to facilitate image retrieval and compression. More recently, Compact Discriminative binary descriptor (CDBin) [231] is proposed to generate the binary descriptors via jointly optimizing four complementary loss functions in an end-to-end manner. In such cases, dedicated prior knowledge (e.g., labels) is required, which is usually impractical in real application scenarios. Despite the great success achieved by those descriptors, the transformation-invariant nature of the local feature descriptor is not considered in the training process. Consequently, DeepBit [106] is proposed to learn compact binary descriptors via optimizing several loss functions in network training, one of which minimizes the Hamming distances of the binary codes from the original patch and their transformed versions in a pairwise manner. Although it encodes the transformation invariance to some extent, the learned binary codes of original data and their transformed sets via minimizing the Euclidean distances between them in the binary space are not identical.

Recently, such learning-based binary descriptors have been widely developed in many other applications like palmprint and object recognition [45, 145]. For example, Discriminant Direction Binary Code (DDBC) [45] learns a simple mapping function to project the convolution difference vectors to the neighboring directions of the templates. While in [145], they propose a stacked convolutional autoencoder structure to generate the compact binary code for the accurate object detection. In these applications, the learning-based binary descriptors act as the leading contributing roles in improving the performance of the specific tasks.

2.1.3 Video Hashing

2.1.3.1 Early Video Hashing

As the video is becoming a crucial information carrier in modern times, however, a minimal amount of effort has been made to push forward the development of video hashing. Early research on video hashing mainly focused on learning proper video representation by fusing the frame-level features such that the existing image hashing techniques can be applied directly to generate video hash codes. Many methods discussed in the image hashing part have been widely used as baselines in recent video

hashing works [65, 109]. In this section, we mainly focus on the original video hashing methods. For example, Douze et al. [38] conduct the video copy detection by using binary codes generating from the uniform-sampled individual video frames. The spatial-temporal consistency between those frames is considered in the frame matching process. Multiple Feature Hashing (MFH) [171] obtains effective binary codes by utilizing multiple types of hand-crafted features and different local frame-level structures. A multi-view video hashing method termed Joint Multi-View Hashing (JMVH) [134] is proposed by preserving the global and local structures of multiple features jointly when learning the hash function. Submodular (Submod) video hashing [18] learns the hash function based on some relevant keyframes from videos, which continues the idea of image hashing in binary code learning. Generally, such methods suffer from two drawbacks: 1) the handcrafted frame-level features have tremendous limitations in the comprehensive video representation; 2) the direct deployment of image hashing in video hash code learning compromises the binary code quality dramatically, thus leading to suboptimal retrieval performance [38, 90, 94].

Later on, the temporal nature over successive frames is becoming more attractive in expressive video representation, including motion trajectory [191] and temporal consistency [230]. The experimental results reveal that such dedicated temporal information, rather than those with spatial features adopted only, indeed enhances the video retrieval performance. For example, a supervised structural video hashing method is proposed in [230], which exploits the temporal consistency between successive frames to learn the linear hashing functions. However, the linear projection may not be able to capture the nonlinear nature of video data truly. A video hashing model termed Hashing across Euclidean space and Riemannian manifold (HER) [104] learns hashing functions based on the kernel max-margin framework for the face video retrieval. Their work represents videos with a single feature representation in the form of a covariance matrix, which cannot fully exploit the spatial-temporal information in videos. Subsequently, a low-rank tensor approximation method is introduced in [99] to model the video clips both in 2D and temporal evolution in the third dimension, thus producing robust video hash codes.

2.1.3.2 Deep Learning Based Video Hashing

Inspired by the recent boosting performance of deep image hashing, deep architectures have been incorporated in the recent video hashing frameworks to improve the retrieval accuracy further. One of the most typical examples is a supervised CNN-based hashing framework [109], namely Deep Video Hashing (DVH), which can

generate similar binary codes for videos belonging to the same category by exploiting the discriminative temporal nature of the video. However, pairwise information is required to compute hash codes in DVH, which might not be easily obtained when dealing with large-scale retrieval tasks. Meanwhile, inspired by the advance of internal video structure in the content modeling, Nonlinear Structural Hashing (NSH) [24] is developed to exploit the nonlinear relationship between videos and structural information between frames via subspace clustering. However, temporal information is completely ignored in NSH. The same problem also occurs in [66], where the proposed Stochastic Multi-View Hashing (SMVH) converts multiple types of keyframe features into binary codes by exploring the relationship between the original feature and its approximated hash code. In [65], they substantially upgrade the working mechanism of SMVH via adopting Student t-distribution and deep neural networks in the similarity preservation and hash function learning separately.

Subsequently, Self-Supervised Temporal Hashing (SSTH) is proposed in [240], where Binary Long Short Term Memory (BLSTM) unit is incorporated into a deep encoder-decoder structure and it directly encodes the video features into compact binary codes. To be specific, the reconstruction losses between the real-valued features and their binary codes are minimized during the batch-wise training, where the hash function learning and the temporal feature learning are uniformly engaged. However, SSTH suffers from serious efficiency issues because of the time-consuming de-binarization and de-LSTM processes in the decoder network. Moreover, directly minimizing the reconstruction error cannot guarantee the preservation of the original neighborhood structure in the code learning, which is crucial for the accurate nearest neighbor search. Despite the potential drawbacks analyzed above, SSTH is viewed as a pioneer work in unsupervised deep video hashing with temporal sequence modeling and a strong competitor in many recent video hashing paradigms [173, 217]. Extension work of SSTH has been released recently in [173] termed Self-Supervised Video Hashing (SSVH), which incorporates a hierarchical binary auto-encoder and neighborhood structure in the code learning. One of the main drawbacks of SSVH is that the similarity matrix is only constructed in the initialization stage without considering the temporal information, which limits the hash code quality improvement. Instead of treating each LSTM step equally as previous works, Attention-based Video Hashing (AVH) [208] adopts an attention mechanism to learn different weights for the LSTM time steps, which enables to capture the temporal information in the video, thus improving the retrieval performance significantly. In [100], Neighborhood Preserving Hashing (NPH) integrates a neighborhood attention mechanism into an

RNN-based autoencoder structure to generate the effective binary codes via capturing the consistent spatial-temporal information in a video sequence.

Recently, the triplet loss is adopted in Similarity-Preserving Deep Temporal Hashing (SPDTH) [157] for video hashing. Accurately, a new loss function termed $\ell_2 All_loss$ is optimized in the end-to-end network training, which aims at preserving the intra-class and inter-class similarity in a mini-batch optimization. Despite the great performance achieved by SPDTH, constructing such triplet input pairs is extremely computationally expensive, thus making it difficult to be deployed in large-scale video retrieval. In Deep Heterogeneous Hashing (DHH) [144] for the face video retrieval, it adopts Riemannian kernel mapping to project data (i.e., covariance matrices of frames) from Euclidean space into the manifold tangent space and optimize the triplet losses between videos afterward, thus preserving the intra-space discriminability and the inter-space compatibility at the same time. However, lacking temporal information in the video modeling of DHH compromises its similarity search performance.

2.2 Cross-Modality Similarity Search

Compared to the single-modality hashing in previous sections, cross-modal hashing aims at tackling the similarity retrieval problems between different modalities. In this section, a typical case of cross-modal hashing termed image and text retrieval is used here as an example. As discussed in Chapter 1, the current research challenge in the cross-modal hashing is how to tackle the modality gap between different modalities properly. A popular solution in this research field is to map the data samples from different modalities (e.g., image, text) into a shared latent feature space such that those data can be measured directly in this space. Similar to the single-modal hashing, existing cross-modal methods can also be classified into supervised and unsupervised ones, which are presented separately in the following sections [199].

2.2.1 Supervised Cross-Modal Hashing

Supervised cross-modal hashing involves dedicated supervision information (e.g., semantic labels, affinity matrix) in the training process, which usually obtains considerable performance gain over unsupervised methods [199]. Several state-of-the-arts are discussed briefly. One of the earliest works is proposed in [10] termed Cross-Modal Similarity Sensitive Hashing (CMSSH). Particularly, the label information is used to assist the preservation of the inter-modality correlation, and two different projection matrices are learned independently for each modality. A cross-modal semantic affinity

matrix is constructed in Semantic Correlation Maximization (SCM) [239] based on the data labels and then is approximated from the to-be-learned binary codes. However, such a matrix reconstruction is usually storage expensive when dealing with large amounts of training data.

Consequently, the semantic data affinity is utilized as a probability distribution model in Semantics-Preserving Hashing (SePH) [108] to alleviate the limitations. The approximated affinity is then obtained by computing the distance in Hamming space via minimizing the Kullback-Leibler divergence (KL divergence), which is relatively low efficiency in the code learning process. Quantized Correlation Hashing (QCH) [214] introduces the quantization loss across the modalities, where the inter-modality correlation is optimized and the quantization error is minimized at the same time. However, the intra-modality correlation is not taken into account during the optimization, which lowers the code quality and leads to unsatisfactory retrieval results. While Discriminant Cross-Modal Hashing (DisCMH) [223] improves the quality of hash codes by means of the label information in the shallow linear classifier. However, all the above methods employ the two-step schemes in code learning, which inevitably yields suboptimal results because of large quantization errors. More importantly, they generally use handcrafted features, such as SIFT [121] for images and Bag-of-Words (BoW) for texts, in their methods, which make the learned code less discriminative [75, 215, 234].

Recently, deep learning technology has been widely incorporated in cross-media hashing [210]. Several representative works are discussed briefly in this section. For instance, Jiang and Li [84] adopt a negative log-likelihood criterion in a deep end-to-end framework named Deep Cross-Modal Hashing (DCMH), where the similarity structure between the real-valued representations is retained. However, such similarity preservation is only performed on the approximated hash codes without restrictions on the true binary codes in the training process. In [32], a Triplet-based Deep Hashing (TDH) method is proposed, which combines the triplet labels and a graph regularization term on the binary codes to ensure the retrieval accuracy. While Deep Visual-Semantic Hashing (DVSH) [20] employs a metric-based approach to train the visual semantic fusion network with cosine hinge loss. However, the label information is not fully exploited and the performance compromises because of the noisy annotations. Subsequently, Textual-Visual Deep Binaries (TVDB) [158] is proposed to exploits the region proposals from the image network and preserves the semantic similarity by using the affinity matrices constructed from the image and text domains

separately. Despite the great performance achieved by TVDB, the similarity preservations are performed in different domains independently, which indicates that the modality gap is not reduced in this case and the performance could be improved by using a unified semantic relationship between modalities.

To address this issue, in [141], Dual-Supervised Attention Network for Deep Hashing (DSADH) is proposed, which involves better feature learning with attention mechanisms and a regression term that links the binary codes to the labels. However, the code quality heavily relies on the label quality in the direct mapping, which may be noisy in the large-scale datasets because of the manual labeling process. To overcome this issue, Semantic Deep Cross-modal Hashing (SDCH) [224] proposes a deep learning framework that explores the cross-modal correlation in both between and within modalities by using the correlation similarity matrix in the end-to-end learning. Consequently, Discrete Latent Factor model based cross-modal Hashing (DLFH) [83] is proposed to model the supervised information in a discrete latent factor model and a corresponding discrete optimization algorithm is presented. Nevertheless, the proposed optimization suffers from the high computational cost in calculating the gradients on the discrete latent variables. Alternatively, Equally-Guided Discriminative Hashing (EGDH) [159] presents a new angle-based connection between the semantic structures across modalities, where the semantic matrix is encoded to build a common semantic classifier in the hash function learning. One limitation is that it only works under an assumption that the norms of the binary codes and the classifier weights equal to the square root of the code length. Cross-Modal correlation Learning with Adversarial samples (CMLA) [98] proposes a novel way to integrate the inter- and intra- modality similarity regularizations by using the adversarial samples from the cross-modal data. As a matter of fact, adversarial training is required in the code learning process.

Recently, more complex networks like GAN and Graph Convolutional Network (GCN) [13] are applied in the cross-modal hashing frameworks to improve the multi-modal retrieval performance further. For example, Cycle-Consistent Deep Generative Hashing (CYC-DGH) [218] adopts an adversarial training scheme via optimizing a cycle consistency loss. Particularly, it enable us to maximize the correlation between the input-output correspondence and minimize the information loss simultaneously. Graph Convolutional Hashing (GCH) [222] utilizes GCN in exploring the inherent similarity structure among the data from multiple modalities. GAN and GCN are the main contributing factors to the great performance boost in their methods.

2.2.2 Unsupervised Cross-Modal Hashing

Regarding unsupervised cross-modal hashing, Cross-View Hashing (CVH) [93], Inter-Media Hashing (IMH) [170] and Linear Cross-Modal Hashing (LCMH) [251] are three representatives in this research field. The first two methods can be regarded as an extension of SH [211] in the cross-modal application. Similar to SH, CVH still assumes the training data follows a uniform distribution, which is unsuitable for the cross-modal data. While IMH performs the relaxed optimization in Euclidean distance instead of Hamming space, thus compromising the code quality. In LCMH, some cluster centroids are picked up to represent the original data, and the intra-modality similarity is preserved by keeping code distances close to the clustering centers. However, the eigenvalue decomposition in LCMH compromises the hash code quality because of arbitrary mapping and the time complexity is highly related to the dataset size [188].

Consequently, Collective Matrix Factorization Hashing (CMFH) [34] adopts matrix factorization to model the relations among different modalities, where the unified binary codes are being learned via projecting the multimodal data into a common latent space. Accordingly, Latent Semantic Sparse Hashing (LSSH) [249] integrates the sparse coding into matrix factorization to learn binary codes for different modalities. However, in both CMFH and LSSH, various relaxation and rounding schemes are utilized in generating hash codes, namely a two-step learning paradigm, which usually leads to large quantization errors in the binarization. Subsequently, Robust and Flexible Discrete Hashing (RFDH) is proposed in [189] to directly optimize and generate the unified binary codes for various views in an unsupervised manner via matrix factorization such that large quantization errors caused by relaxation can be relieved to some extent. Despite the claimed contributions from RFDH, the neighborhood structures of inter-modal and intra-modal existed in the original data are not explored.

Similar to the supervised methods, recent works tend to combine the deep learning techniques in unsupervised cross-modal hashing, where most of them utilize the auto-encoder structure in their methods. For example, Multi-modal Stacked Auto-Encoders (MSAE) [206] adopts several auto-encoders to formulate the hash functions for heterogeneous data, where two-phase training procedures: pre-training and fine-tuning, are required. However, this two-phase optimization doubles the training time and the intra-modality data similarities are no longer preserved after fine-tuning. Different from MSAE, a stacked auto-encoder architecture termed Correlation Autoencoder Hashing (CAH) is proposed in [21], where the feature and semantic correlation

across modalities are jointly maximized. In [102], another auto-encoder framework for unsupervised cross-modal hashing termed Deep Binary Reconstruction (DBRC) is proposed, which reconstructs the original features from the joint binary representation without considering the similarity relations. As discussed in RFDH, such ignorance on the neighbor relationships across different domains inevitably lowers the discriminativeness of the hash codes, thus compromising the retrieval performance. In light of this issue, Unsupervised Deep Cross-modal Hashing with Virtual Label Regression (UDCH-VLR) [205] establishes a link between the virtual labels and the to-be-learned binary codes for different modalities. However, the code quality could be affected dramatically by the initialization ways of virtual labels, which is sometimes intractable when optimizing the objective function.

More recently, Deep Joint-Semantics Reconstructing Hashing (DJSRH) [178] is proposed to learn the binary codes while preserving the original manifold structure via integrating a joint-semantics affinity matrix in the code learning. In the training process, the binary codes from two modalities (image and text) are not identical. While in [242], they propose a multi-pathway GAN-based hashing framework termed Multi-pathway Generative Adversarial Hashing (MGAH), where the GAN is trained along with a correlation graph-based approach that captures the manifold structures across different modalities for better retrieval performance. However, the training process is somewhat intractable due to the complicated GAN structure.

2.3 Chapter Summary

In this chapter, we have reviewed most but not limited to related hashing-based works on three popular application scenarios: single-modality similarity search, such as image hashing, local feature descriptor, and video hashing, and cross-modality hashing. More hashing methods can be found in [193, 194, 197]. Our works in this thesis are greatly inspired by the related works discussed in previous sections. In the main chapters, some of these works are further analyzed and picked up as the state-of-the-art baselines in our experiments.

In the following chapters, we focus on detailing our proposed methods regarding different applications and testing the system performance on the public datasets. Particularly, Chapter 3 presents a new deep binary descriptor for the large-scale matching/retrieval tasks. This work is inspired by the idea of unsupervised multi-view embedding via matrix factorization and the matching performance is further improved by imposing a weak bit scheme on ambiguous matching. Then Chapter 4 addresses the

large-scale video-to-video retrieval via jointly engaging the deep video feature learning with the proposed balanced rotation. Chapter 5 presents the proposed cross-modal hashing frameworks that integrate the binary code with the hash function learning to improve the cross-media retrieval performance.

Chapter 3

Unsupervised Deep Binary Descriptor

3.1 Introduction

Recently, the local binary descriptor has attracted wide attention in many visual applications, such as patch matching, image retrieval, object recognition and 3D reconstruction [12, 68, 121, 179]. Benefiting from the characteristics of high compactness and efficient bitwise calculation, binary descriptor is a more favorable option in conducting matching and retrieval in *large-scale* database over the traditional floating-point descriptors (e.g., SIFT [121], FAST [148] and SURF [6]) [2, 183]. This paper focuses on applying binary descriptor in both patch matching and image retrieval, where patches can be obtained from full image via keypoint detection technology in the former applications [11].

Consistent with traditional feature descriptors, binary descriptor is supposed to represent data (image/patch) accurately in despite of geometric transformations (e.g., rotation, translation and scaling) [97, 121]. Earlier binary descriptors (e.g., BRIEF [17], BRISK [97], ORB [150] and FREAK [1]) are generally data-independent, which adopt various hand-crafted sampling patterns and perform a series of pairwise intensity comparisons afterwards [231]. However, such predefined sampling modes and intensity comparisons are extremely vulnerable to the distortions/transformations, thus yielding unstable performance when tackling large-scale visual recognition tasks [182, 183, 231]. Consequently, many efforts have been devoted to developing learning-based binary descriptors. Existing methods draw on the soul idea from hashing techniques (e.g., LSH [2], ITQ [51], CMFH [35]), where the data points are projected from their original feature space into the compact binary space and the similar points could be represented by the similar binary descriptors (low Hamming distance) [41, 177, 231].

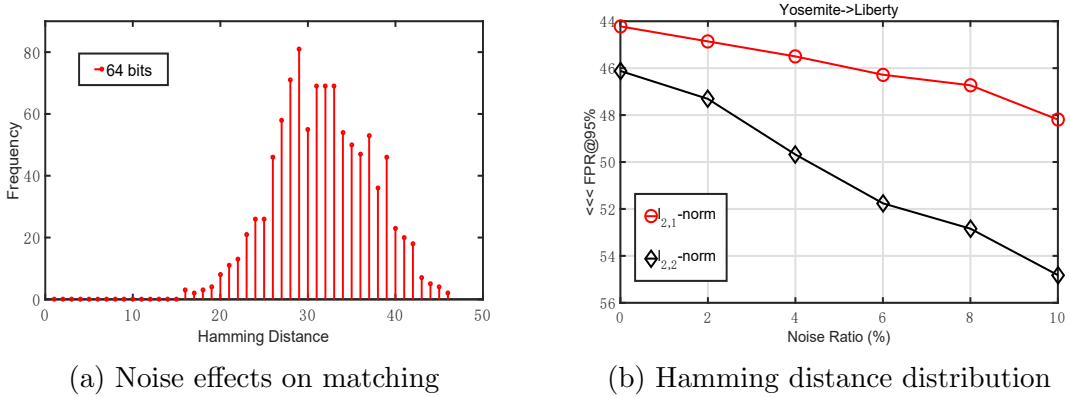


Figure 3.1: (a) An example of the Hamming distance distribution on Cifar-10 dataset at 64 bits, where 3 candidates are returned from the database with the same minimum Hamming distance of 16 to the query; (b) Noise effects on Brown dataset (train: *Yosemite* and test: *Liberty*) at 256 bits under $\ell_{2,1}$ -norm and $\ell_{2,2}$ -norm losses, where a sharper performance decline from $\ell_{2,2}$ -norm against $\ell_{2,1}$ -norm loss is observed at certain noise level.

Although the learning-based binary descriptors obtain great performance gains over the handcrafted ones, some drawbacks become bottlenecks that impede their further development in large-scale application scenarios.

Firstly, they pay intensive attention to novel discrete optimization strategies, while the nature of local feature descriptor, anti-geometric transformation, cannot be fully guaranteed [97, 183]. That is crucial to the success of binary descriptor in large-scale visual recognition tasks. More worriedly, most learning paradigms fail to preserve the manifold structure during the discrete optimization, which makes binary descriptor less effective in large-scale neighbor search tasks [56, 107, 153]. Supervised methods address this issue by using prior knowledge (e.g., pair-wised labels). However, they are not preferred in real-world applications because of intensive labeling work.

Furthermore, traditional binary descriptors measure the similarity between database and query via exhaustive Hamming distance calculations in the testing phase. In practice, however, it is more likely to return many candidates with equal Hamming distances to one specific query [120]. To clarify the problem, we plot the Hamming distance distribution of a query to the database (with randomly-selected 1,000 candidates) on Cifar-10 dataset in Fig. 3.1(a). For instance, 3 candidates are returned from the database with the same minimum Hamming distance of 16 to the query at the code length of 64. That reduces the discriminative power of binary descriptor dramatically. It is especially harmful to the matching performance, where *each query is expected to be matched with one exact candidate* (with the lowest Hamming

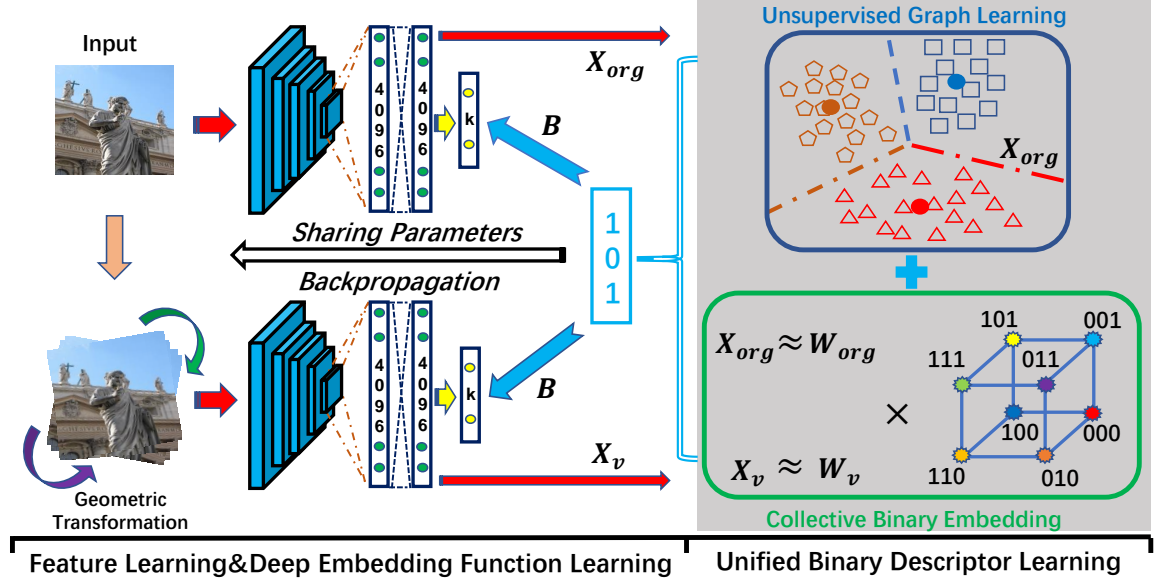


Figure 3.2: The proposed binary descriptor learning framework is made up of deep feature extraction, unified binary code learning and deep embedding function learning. The descriptor size is set to 3 as an example. Best viewed in color.

distance) rather than a bunch of ambiguous options (with equal minimum Hamming distance).

In this chapter, we propose a novel learning-based framework, termed **Unsupervised Deep Binary Descriptor (UDBD)**, to overcome the above limitations in compact binary descriptor learning. Fig. 3.2 shows the flowchart of UDBD. Particularly, the original visual data and their transformed counterparts are projected into *common* Hamming subspace directly during the binary code learning. By doing so, transformation invariance could be conserved along with the binary embedding process, which is theoretically more advanced than the primitive approach [106] that simply minimizes the differences between the binary codes of original data and those transformed ones.

In the meantime, $\ell_{2,1}$ -norm loss is employed together with the proposed binary embedding to improve the robustness of our binary descriptor against data noises/outliers for the patch-level recognition tasks [56, 85]. To make it clear, we plot the matching performance variations with increasing noise ratios using ITQ+ [56] on Brown dataset [11] in Fig. 3.1(b). As can be seen, there is a sharper performance decline from $\ell_{2,2}$ -norm against $\ell_{2,1}$ -norm loss function at certain noise level. The main cause is that patches mainly contain low-level texture features, which are more prone to the noises/outliers compared to general global images [179]. Without noticing it, previous methods directly adopt the squared ℓ_p -norm regularization to build

their loss functions [56], which may exaggerates the adverse effects caused by severe noises/distortions, thus leading to worse results [56, 112]. That implies $\ell_{2,1}$ -norm loss is more suitable for the patch-level recognition.

Then an unsupervised graph constraint is formed and added into the loss function so as to preserve the original manifold structure of training data in the Hamming space [153, 211]. With an alternating optimization scheme, the binary code can be solved directly without relaxation, which avoids the accumulated quantization errors from the two-step learning strategy [19, 51, 180]. By training an unified deep network with the guidance of the learned binary descriptors, the deep embedding function is able to generate robust binary codes for various visual tasks.

In the feature matching procedure, a weak bit scheme, where the Hamming distance is recalculated based on the reliability of each bit, is further applied to find the best match among the returned candidates with the same initial Hamming distance [59, 161]. In summary, our work differs the previous algorithms in the following three aspects:

- To the best of our knowledge, this is the first work that learns the transformation invariant binary descriptor via *embedding the original visual data and their transformed sets into a common Hamming space in an unsupervised manner*. Moreover, a graph constraint that preserves the manifold structure from the original feature space is employed in the unified binary representation learning, thus improving the code quality.
- Since patch mainly contains noise-sensitive local features, $\ell_{2,1}$ -norm loss is proposed to regularize the binary embedding. On one hand, ℓ_1 -norm distance at the patch level provides the robustness against outlier samples. On the other hand, ℓ_2 -norm measures the distance along space dimension, which spreads out the errors over each bit uniformly to lower the possibility that certain bits are mistakenly flipped after getting large errors. To this end, an alternating discrete optimization strategy is proposed to optimize the $\ell_{2,1}$ -norm constrained objective function, where the binary code can be solved directly with no need for relaxation.
- As a means of distance re-measure, a weak bit scheme, which considers the reliability of each bit in a descriptor, is applied along with the proposed binary descriptor. It helps to find the best match if there are multiple candidates with the same distance to the query when comparing the Hamming distance of descriptors.

The rest of this work is organized as follows. In Section 3.2, the proposed method is elaborated along with the comprehensive analysis. Extensive experimental results are provided and analyzed in Section 5.3. Finally, the conclusion is given in Section 3.4.

3.2 Methodology

3.2.1 Framework Overview

Some mathematical symbols are defined to ease the following explanations on the framework. Assuming that the training set consists of n data samples (images/patches) and each one has m different transformation sets, where the transformed versions of each sample could be obtained by rotation, scaling and translation [106]. We denote the training set as $\mathcal{O} = \{o_i\}_{i=1}^n$, $o_i = \{x_v^i\}_{v=1}^m$, where v denotes the index of the transformation set, $x_v^i \in \mathbb{R}^{p_v}$ is a feature vector and p_v represents the dimensionality of x_v^i in the set. For each transformation set, we denote the feature matrix as $\mathbf{X}_v = [x_v^1, x_v^2, \dots, x_v^n] \in \mathbb{R}^{p_v \times n}$. Given the code length k , the goal of the proposed method is to learn the *unified* binary descriptor $\mathbf{B} \in \{-1, +1\}^{k \times n}$ for the training samples within all transformation sets. Particularly, each sample and its transformed versions should be encoded as the same binary code because the semantics of those samples keeps unchanged even after certain transformations.

To achieve this learning goal, the deep features of different input sources are first extracted from the fully-connected (*fc*) layers of a pre-trained VGG-16 network [164]. Then the features are fed into binary code learning that generates the uniformed binary descriptor for various transformation sets via exploring their common binary space. With sharing network parameters Θ , an unified deep embedding function \mathcal{H} is built via projecting all transformation sets into the learned binary code to generate new descriptors for the query. In the online stage, a weak bit scheme that excludes the contributions of unreliable bits in distance calculation is adopted to further improve the matching performance. The major mathematical symbols used in this chapter are summarized in Table 3.1 for the ease of explanation. Other symbols like \mathbf{G}_v and $\tilde{\mathbf{X}}_v$ are applied as the auxiliary parameters in the equation deduction, which are omitted in this table.

Table 3.1: Mathematical symbols and their short descriptions.

Symbol	Description	Symbol	Description
\mathcal{O}	training set	n	number of training samples
x_v	feature vector	m	transformation set number
\mathbf{X}_v	feature matrix	\mathbf{S}	affinity matrix
v	transformation set index	\mathbf{L}	Laplacian matrix
\mathbf{B}	unified binary descriptor	k	code length
\mathbf{W}_v	latent embedding matrix	α_v	weight factors
β, γ	balance parameters	z	weak bit mark
$\mathcal{H}(\cdot)$	deep embedding function	Θ	network parameter
f	real-valued feature vector	th	threshold

3.2.2 Learning Unified Binary Descriptor

In this section, we analyze two involved sub modules within the unified binary descriptor learning: collective binary embedding and unsupervised graph learning.

3.2.2.1 Collective Binary Embedding

The ideas of this module can be explained from two aspects: 1) the original image patch and its transformed versions should be encoded with the same binary descriptor, which can be achieved via embedding all the those sets into a *common* Hamming space; 2) the unified binary code is learned from different transformation sets, which encodes the nature of transformation invariance to the maximum. Particularly, we formulate the objective function of this part as below:

$$\begin{aligned}
& \min_{\mathbf{B}, \mathbf{W}_v, \alpha_v} \sum_{v=1}^m (\alpha_v)^\gamma \|\mathbf{X}_v - \mathbf{W}_v \mathbf{B}\|_{2,1}, \\
& \text{s.t. } \mathbf{B} \in \{-1, +1\}^{k \times n}, \sum_{v=1}^m \alpha_v = 1, \alpha_v > 0.
\end{aligned} \tag{3.1}$$

where $\mathbf{B} \in \{-1, +1\}^{k \times n}$, $\sum_{v=1}^m \alpha_v = 1$ and $\alpha_v > 0$. Here, \mathbf{X}_v are the deep features extracted from the *fc7* layer of the pretrained VGG-16 model, $\mathbf{W}_v \in \mathbb{R}^{p_v \times k}$ are the latent embedding matrices that connect the unified binary descriptor with the deep features. α_v are the weight factors that measure the contributions of different transformation sets in learning the binary descriptor. γ is the balance parameter. $\ell_{2,1}$ -norm is defined as $\|Y\|_{2,1} = \sum_{i=1}^n \|y_i\|_2$ for a matrix $Y = [y_1, y_2, \dots, y_n] \in \mathbb{R}^{p \times n}$ [112].

Generally speaking, this module is proposed to encode the transformation invariance maximally in the to-be-learned binary descriptor via applying affine-transformation and performing matrix factorization on every single patch. It is worth noting that this module differs data augmentation in traditional classification tasks. From the functionality perspective, data augmentation involves the process of creating new data points by manipulating the original data to increase the training data diversity, thus avoiding overfitting. However, the overfitting issue is not our concern here and the transformed data is provided merely for the proposed invariance encoding. From the technical perspective, being identified as the same category label is the only optimization goal for the original image and its augmented ones in the classification. The same category label does not necessarily guarantee the same feature descriptor. In our method, Eq. (3.1) regularizes all transformation sets of each patch to be represented by a unified binary descriptor (i.e., feature). Therefore, our learning objective is more stringent and optimizing such complicated loss functions is much more challenging.

More importantly, the proposed embedding function has been upgraded to make it more compatible with the local binary descriptor learning [35, 180]. Firstly, $\ell_{2,1}$ -norm is introduced into the discrete optimization model to reduce the negative effects caused by severe noises/distortions. In contrast to the widely-used squared ℓ_2 -norm that is extremely prone to noises/outliers, $\ell_{2,1}$ -norm is a more rational choice in the patch-level transformation invariant descriptor learning. On one hand, ℓ_1 -norm distance at the patch level provides the robustness against outlier samples after random transformations in this case (see Fig. 3.1(b)). On the other hand, ℓ_2 -norm distance enables the error allocations to each bit uniformly across the space dimension. Doing so lowers the possibility that certain bits are mistakenly flipped after getting large errors [56, 85, 112, 143, 229]. Those flipped bits may dramatically disturb the subsequent Hamming distance measure. Moreover, we solve the unified binary representation \mathbf{B} directly under the restrictions of $\ell_{2,1}$ -norm in the proposed model, where the accumulated quantization errors from the two-step learning paradigms [35, 56, 88, 153, 180] are avoided and the robustness of the learnt binary code is further enhanced.

3.2.2.2 Unsupervised Graph Learning

As discussed above, the essence of the binary descriptor learning can be described as a process of projecting the high-dimensional original features into the compact binary space properly [2, 51, 56]. During the projection, the neighbourhood relationship preservation plays an important role in *generating the similar binary descriptors for those data (images/patches) that belong to the same category*. In this work, a

unsupervised Laplacian constraint is derived from the *original* data set and imposed on all the transformation sets during the optimization, which shares the similar idea in common dictionary learning [19, 56, 247]. The reason is that the relative positions of data points in the feature space will be inevitably shifted after geometric transformations and provide unreliable neighborhood structures within the transformation sets [68, 85, 131]. That will mislead the unified binary descriptor learning and thus adversely affect the code quality. The basic functionality of using such a Laplacian term is to keep the consistency between the original and binary feature spaces during the code learning [116, 153, 211]. Let $\mathbf{B}_{*,j}$ and $\mathbf{B}_{*,l}$ denote the j -th and l -th columns of \mathbf{B} , the affinity matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$ from the original patch set, the graph problem can be formulated as follow:

$$\begin{aligned} \min_{\mathbf{B}} \quad & \frac{1}{2} \sum_{j=1}^n \sum_{l=1}^n \|\mathbf{B}_{*,j} - \mathbf{B}_{*,l}\|_F^2 \mathbf{S}_{j,l} = \text{tr}(\mathbf{B}\mathbf{L}\mathbf{B}^T), \\ \text{s.t. } \quad & \mathbf{B} \in \{-1, +1\}^{k \times n}, \mathbf{L} \in \mathbb{R}^{n \times n}, \mathbf{S} \in \mathbb{R}^{n \times n}, \end{aligned} \quad (3.2)$$

where $\mathbf{B} \in \{-1, +1\}^{k \times n}$. $\mathbf{L} \in \mathbb{R}^{n \times n}$ is the Laplacian constraint and computed as $\mathbf{L} = \text{diag}(\mathbf{S}\mathbf{1}) - \mathbf{S}$. $\text{diag}(\mathbf{S}\mathbf{1})$ represents the diagonal matrices with each diagonal element being calculated as the sum of values in the corresponding row of \mathbf{S} , where \mathbf{S} is constructed via k-Nearest-Neighbour (kNN). Particularly, the anchor graph scheme [118] can be adopted to reduce the computational complexity following the previous works [116, 155].

Based on the discussions, the unified binary code for the training data can be learned via jointly optimizing the above learning objectives. By incorporating Eq. (5.19) into Eq. (3.1), the overall objective function of unified binary descriptor learning can be formulated as:

$$\min_{\mathbf{B}, \mathbf{W}_v, \alpha_v} \sum_{v=1}^m (\alpha_v)^\gamma \left(\|\mathbf{X}_v - \mathbf{W}_v \mathbf{B}\|_{2,1} + \beta \text{tr}(\mathbf{B}\mathbf{L}\mathbf{B}^T) \right), \quad (3.3)$$

where $\sum_{v=1}^m \alpha_v = 1$, $\alpha_v > 0$. $\mathbf{W}_v \in \mathbb{R}^{p_v \times k}$, $\mathbf{L} \in \mathbb{R}^{n \times n}$ and $\mathbf{B} \in \{-1, +1\}^{k \times n}$. β is the balance parameter.

3.2.3 Optimization Algorithm

It is intractable to solve the objective function Eq. (3.3) directly because of the discrete-constrained conditions and the non-convex $\ell_{2,1}$ -norm term, which refers to an NP-hard problem [56, 72, 190]. Consequently, an alternating optimization strategy is employed to tackle this issue, which is presented as the following steps.

3.2.3.1 \mathbf{W}_v Step

For \mathbf{W}_v with other parameters fixed, the objective function in Eq. (3.3) can be simplified as follow:

$$\psi_v = \min_{\mathbf{W}_v} \|\mathbf{X}_v - \mathbf{W}_v \mathbf{B}\|_{2,1} = \min_{\mathbf{W}_v} \sum_{i=1}^n \|\mathbf{X}_v^i - \mathbf{W}_v \mathbf{B}^i\|_2, \quad (3.4)$$

where $\mathbf{W}_v \in \mathbb{R}^{p_v \times k}$, \mathbf{X}_v^i and \mathbf{B}^i are the i -th columns of \mathbf{X}_v and \mathbf{B} , respectively. Then we can calculate the gradient of ψ_v with respect to \mathbf{W}_v as:

$$\begin{aligned} \frac{\partial \psi_v}{\partial \mathbf{W}_v} &= \sum_{i=1}^n \frac{\mathbf{W}_v \mathbf{B}^i (\mathbf{B}^i)^T - \mathbf{X}_v^i (\mathbf{B}^i)^T}{\|\mathbf{X}_v^i - \mathbf{W}_v \mathbf{B}^i\|_2} \\ &= (\mathbf{W}_v \mathbf{B} - \mathbf{X}_v) \mathbf{D}_v \mathbf{B}^T. \end{aligned} \quad (3.5)$$

Here, the diagonal matrix \mathbf{D}_v are led into the problem and its i -th diagonal element is obtained as $(\mathbf{D}_v)_{i,i} = \frac{1}{\|\mathbf{X}_v^i - \mathbf{W}_v \mathbf{B}^i\|_2}$. Although there is no closed-form solution for \mathbf{W}_v in the above equation, the calculation of $(\mathbf{X}_v - \mathbf{W}_v \mathbf{B})$ can be leveraged to compute \mathbf{D}_v and $\frac{\partial \psi_v}{\partial \mathbf{W}_v}$ directly with the minimal efforts. Then a gradient descent strategy can be employed to optimize the objective function [16].

3.2.3.2 \mathbf{B} Step

For \mathbf{B} with other parameters fixed, the objective function (3.3) can be further rewritten as follow:

$$\min_{\mathbf{B}} \sum_{v=1}^m (\alpha_v)^\gamma \left(\|\mathbf{X}_v - \mathbf{W}_v \mathbf{B}\|_{2,1} + \beta \text{tr}(\mathbf{B} \mathbf{L} \mathbf{B}^T) \right), \quad (3.6)$$

where $\mathbf{B} \in \{-1, 1\}^{k \times n}$. Inspired by recent coordinate descent based methods [153], the objective loss can be minimized via optimizing all the bits in \mathbf{B} sequentially. Here, we denote $b^T \in \{-1, 1\}^{1 \times n}$ as the i -th row of \mathbf{B} , and \mathbf{B}' the matrix of \mathbf{B} excluding b^T . Let $w_v \in \mathbb{R}^{p_v}$ be the i -th column of \mathbf{W}_v , \mathbf{W}'_v be the matrix of \mathbf{W}_v excluding w_v . Considering $\mathbf{W}_v \mathbf{B} = \mathbf{W}'_v \mathbf{B}' + w_v b^T$, $\text{tr}(\mathbf{B} \mathbf{L} \mathbf{B}^T) = \text{tr}(\mathbf{B}' \mathbf{L} \mathbf{B}'^T) + b^T \mathbf{L} b$ and $\text{tr}(\mathbf{B}' \mathbf{L} \mathbf{B}'^T)$ is *const*, Eq. (3.6) with respect to $b \in \{-1, 1\}^n$ can be formulated as:

$$\min_b \sum_{v=1}^m (\alpha_v)^\gamma \left(\|\mathbf{X}_v - \mathbf{W}'_v \mathbf{B}' - w_v b^T\|_{2,1} + \beta b^T \mathbf{L} b \right). \quad (3.7)$$

Let $\tilde{\mathbf{X}}_v = \mathbf{X}_v - \mathbf{W}'_v \mathbf{B}'$, Eq. (3.7) is further simplified as:

$$\min_b \sum_{v=1}^m (\alpha_v)^\gamma \left(\|\tilde{\mathbf{X}}_v - w_v b^T\|_{2,1} + \beta b^T \mathbf{L} b \right). \quad (3.8)$$

The above derivations transform the objective function into the similar form like Binary Quadratic Problem (BQP), but more complex. The closed-form solution of b cannot be obtained directly from Eq. (3.8). Following the previous works, it is still feasible to optimize the objective function via flipping each bit sequentially in b , where the bit would be flipped if the flipping operation decreases the objective function loss [153]. Fortunately, the initial values for b , denoted as b^0 , can be set properly to minimize the first term in Eq. (3.8). Namely, the j -th bit in b^0 is calculated as:

$$b_j^0 = \text{sign} \left(\sum_{v=1}^m (\alpha_v)^\gamma (\|\tilde{\mathbf{X}}_v^j + w_v\|_2 - \|\tilde{\mathbf{X}}_v^j - w_v\|_2) \right), \quad (3.9)$$

where $b_j^0 \in \{-1, 1\}$ and $\tilde{\mathbf{X}}_v^j \in \mathbb{R}^{p_v}$ is the j -th column of $\tilde{\mathbf{X}}_v$. $\text{Sign}(x) = 1$ if $x \geq 0$ and otherwise -1 . After getting b^0 , we can flip each bit sequentially as in [153] to optimize the objective function.

3.2.3.3 α_v Step

For α_v with other parameters fixed and let $\mathbf{G}_v = \|\mathbf{X}_v - \mathbf{W}_v \mathbf{B}\|_{2,1} + \beta \text{tr}(\mathbf{B} \mathbf{L} \mathbf{B}^T)$, we can rewrite Eq. (3.3) as:

$$\min_{\alpha_v} \sum_{v=1}^m (\alpha_v)^\gamma \mathbf{G}_v, \quad \text{s.t.} \quad \sum_{v=1}^m \alpha_v = 1, \alpha_v > 0. \quad (3.10)$$

By introducing the Lagrange multiplier η , the above problem is then transformed to:

$$\min \mathcal{E}(\alpha_v, \eta) = \sum_{v=1}^m (\alpha_v)^\gamma \mathbf{G}_v - \eta \left(\sum_{v=1}^m \alpha_v - 1 \right), \quad (3.11)$$

where the partial derivatives with respect to α_v and η are calculated as:

$$\begin{cases} \frac{\partial \mathcal{E}_v}{\partial \alpha_v} = \gamma (\alpha_v)^{\gamma-1} \mathbf{G}_v - \eta, \\ \frac{\partial \mathcal{E}_v}{\partial \eta} = \sum_{v=1}^m \alpha_v - 1. \end{cases} \quad (3.12)$$

By setting those derivatives as 0, we have the optimal solution of α_v as:

$$\alpha_v = \frac{(\mathbf{G}_v)^{\frac{1}{1-\gamma}}}{\sum_{v=1}^m (\mathbf{G}_v)^{\frac{1}{1-\gamma}}}. \quad (3.13)$$

By repeating the above steps, the objective function converges to local minimum after a few iterations (the iteration number $t \leq 10$ in the experiment), thus obtaining unified binary descriptors for the training data. The major difference against the previous discrete optimization strategies [153, 180, 190] is that only the gradient descent is performed to make the overall objective function keep decreasing in the proposed method. There is no need to find the closed-form solution for each variable during each optimization iteration [36].

3.2.4 Generating Out-of-Sample Binary Descriptor

After learning the binary descriptors for training data, an unified deep embedding function $\mathcal{H}(\mathbf{X}_v; \Theta)$ is trained as the code generator for out-of-sample data. Particularly, the input data \mathbf{X}_v from multiple sets ($v = 1, \dots, m$) are sequentially fed into the deep network and the Euclidean distances between feature vectors from the last output layer and their corresponding binary representations \mathbf{B} are minimized, as shown in Fig. 3.2. By doing so, the geometric transformation invariance could be preserved maximally during the deep embedding function learning. Moreover, the computational complexity can be reduced by updating the sharing weight Θ for the original data and its transformation sets simultaneously, instead of training different deep networks for them separately as in [106]. The objective function of this process is presented as:

$$\min_{\Theta} \sum_{v=1}^m \|\mathcal{H}(\mathbf{X}_v; \Theta) - \mathbf{B}\|_F^2, \text{ s.t. } \mathbf{B} \in \{-1, +1\}^{k \times n}. \quad (3.14)$$

The optimization problem can be solved by fine-tuning the deep network with Stochastic Gradient Descent (SGD), where the sharing weight Θ is iteratively optimized until convergence. Given a query instance \mathbf{x}_q , we can obtain its binary descriptor by simply calculating $\text{sign}(\mathcal{H}(\mathbf{x}_q; \Theta))$. The proposed algorithm is summarized in Algorithm 1.

3.2.5 Refined Matching via Weak Bit Selection

Once we have obtained binary descriptors for both query and gallery data, the matching can be done by simply comparing their Hamming distance. However, as the binary representation reduces the discriminative power of data, it is very often that there are multiple candidates with the same minimum Hamming distance (even 0 in the worst-case scenarios) to a specific query (see Fig. 3.1(b)). It might be acceptable for

Algorithm 1 Unsupervised Deep Binary Descriptor

Input: Deep features \mathbf{X}_v for different transformation sets, code length k , parameters β and γ , Laplacian matrix \mathbf{L} , maximum epoch T . Randomly initialize binary code \mathbf{B} , latent embedding matrices \mathbf{W}_v and deep parameters Θ . Set average weights $\alpha_v, v = \{1, \dots, m\}$.

Output: Deep hash functions $\mathcal{H}(\mathbf{X}_v; \Theta)$;

- 1: Extract the feature matrices \mathbf{X}_v from *fc7* layers;
- 2: **for** $t = 1$ to T **do**
- 3: Update the latent embedding matrices \mathbf{W}_v by Eq. (3.5);
- 4: Update the unified hash code \mathbf{B} by Eq. (3.7)~(3.9);
- 5: Update the weight factors α_v by Eq. (3.13);
- 6: **end for**
- 7: Update the network parameters Θ by Eq. (3.14);
- 8: **return** $\mathcal{H}(\mathbf{X}_v; \Theta)$;

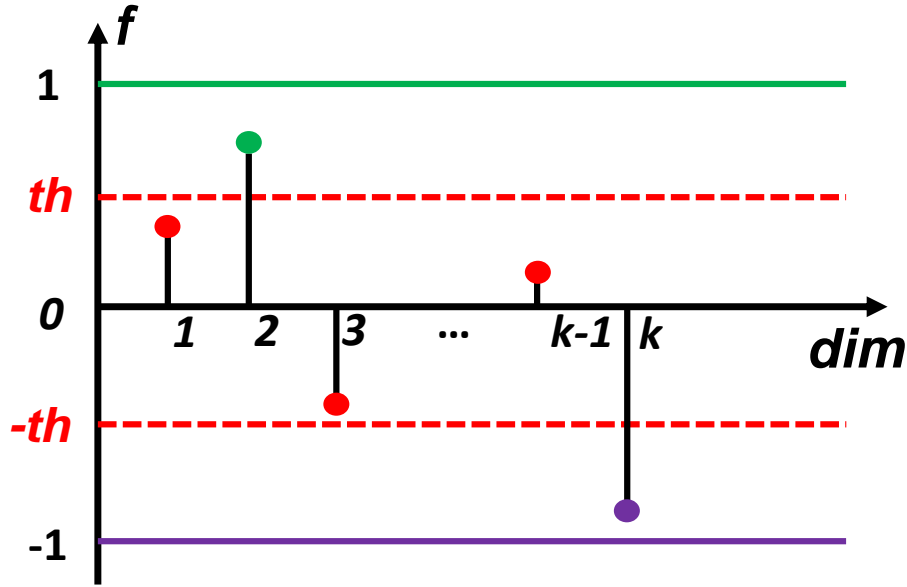


Figure 3.3: An example on the weak bit selection process. The red circles denote the marked weak bits with the values between $(-th, th)$.

applications like retrieval, but is definitely a problem for local feature points matching, where one true match should be provided. In this case, a means to conduct the second distance measurement is required. Inspired by the advocate of weak bit (i.e., unreliable bit) in fingerprinting systems [5, 59, 127, 160], we found that the contribution/reliability of each bit within the binary codes differs. Hence, such information can be useful to refine the initial Hamming distance computation. Concretely, the unreliable bits (with values closed to 0) for each input $x \in \mathbb{R}^p$ are selected based on

its real-valued vector $f \in \mathbb{R}^k$, which is extracted from the last output layer of the deep embedding network. With a certain threshold $th > 0$, the weak bit $z \in \{0, 1\}^k$ in its binary code $b \in \{-1, 1\}^k$ can be defined as:

$$z_k = \begin{cases} 1, & |f_k| < th; \\ 0, & |f_k| \geq th, \end{cases} \quad (3.15)$$

where the bits with values in the range of $(-th, th)$ are marked as weak bits (ie, $z_k = 1$). The weak bit selection process is briefly presented in Fig. 3.3. Here, the intuition is that the closer the real-valued feature gets to 0 the weaker it will be. This does make sense because the value closer to 0 is likely to be mistakenly flipped in the existence of noises, considering the fact that we use a *sign* function to convert a real value to a binary bit. In this second matching procedure, a sequence of binary digits of a query, formed by weakness indications at each bit location, will be compared against the counterpart digits of a candidate. As a result of doing this, the aggregated distance enables to find the best match, thus improving the matching performance.

3.3 Experiment

In this section, we conduct extensive experiments on three public datasets to evaluate the matching and retrieval performance of the proposed binary descriptor.

3.3.1 Dataset Descriptions

3.3.1.1 Brown

Brown¹ [11] is the most popular dataset in the evaluation of local feature descriptors, which contains three subsets: *Notre Dame*, *Yosemite*, and *Liberty* collected from the Photo Tourism reconstructions. In each subset, there are more than 400,000 gray-scale patches with the size of 64×64 . Those patches are split into training and test sets, which contains 200,000 pairs (100,000 matched and non-matched pairs) and 100,000 pairs (50,000 matched and non-matched pairs), respectively.

3.3.1.2 Cifar-10

Cifar-10² [89] consists of 60,000 images with the size of 32×32 from 10 different categories, which are split into training and test sets with 50,000 and 10,000 images

¹<http://matthewalunbrown.com/patchdata/patchdata.html>

²<https://www.cs.toronto.edu/~kriz/cifar.html>

separately. The training set is employed for the code learning, and use the test set as the queries for retrieval evaluation.

3.3.1.3 HPatches

Homography Patches (HPatches)³ [3] consists of about 1 million patches extracted from 116 images using the combination of various interest point detectors, where the patches are collected from the 3D reconstructions of several landmarks in Rome. Each patch is annotated with its ground truth label and then post-processed after extraction with the fixed size of 65×65 . We follow the default settings in [231] and test the performance on the full split within the dataset.

3.3.2 Implementation Details

The experiments are carried out on Linux Ubuntu Server with the configuration of Intel i7-5960X CPU@3.0GHz, 64GB RAM and NVIDIA GTX 1080 Ti GPU. Most source codes of the baselines are publically available online, which can be tuned via open source softwares (e.g., *Caffe* [82], *OpenCV* [9]) according to the papers. Specifically, the geometric transformation of the input patches are implemented by following the data augmentation in [106], where the rotation angles are within the range of $[-10, 10]$. Particularly, 5 different rotation angles: $[-10, -5, 0, 5, 10]$, are imposed on each input patch, which simulates the small viewpoint variations from human perspective [106]. Their deep features are extracted from the *fc7* layer (4096-d) of the pre-trained VGG-16 [164].

In the proposed method, γ and β are set as 5 and 10^{-3} during the discrete optimization, while the discrete optimization usually converges within 10 iterations. In the network training phase, VGG-16 model is used as the backbone with the output size of k and *tanh* as activation function in the last *fc* layer. The basic learning rate as 0.0001, momentum as 0.9 and weight decay as 0.0005. The batch sizes is 32 and the maximum iteration is 30000. The threshold is set to 0.3 via cross-validation in the weak bit selection.

3.3.3 Comparisons with State-of-The-Arts

3.3.3.1 Results on Brown Dataset

On Brown dataset, we conduct extensive comparisons on the patch matching performance between our approach and several state-of-the-art binary descriptors. These

³<https://github.com/hpatches/hpatches-dataset>

Table 3.2: Comparison of the proposed UDBD to the state-of-the-art binary descriptors in terms of FPR@95% on Brown dataset. Dim, SP and USP denote dimension, supervised and unsupervised, respectively. † and ‡ indicate the train and testing subsets. The results from SIFT and supervised methods are provided as references. Bold values are the best results in unsupervised binary descriptors.

Method	Dim	Type	Notre Dame [†]		Liberty [†]		Yosemite [†]		Yosemite [†]		Average FPR@95%
			Liberty [†]	Yosemite [‡]	Notre Dame [‡]	Yosemite [‡]	Notre Dame [‡]	Liberty [‡]			
SIFT [121]	128	USP	36.27	29.15	28.09	29.15	28.09	36.27		31.17	
BinBoost [183]	64	SP	20.49	18.96	16.9	22.88	14.54	21.67		19.24	
L2-Net [182]	128	SP	7.53	7.74	5.92	9.12	5.43	9.25		7.49	
HardNet [130]	128	SP	2.22	2.28	0.57	2.13	0.96	2.35		1.9	
CDbin [231]	128	SP	6.81	3.02	7.92	3.02	4.26	9.0		6.46	
BRIEF [17]	256	USP	59.15	54.96	54.57	54.96	54.57	59.15		56.23	
BRISK [97]	512	USP	79.36	73.21	74.88	73.21	74.88	79.36		75.82	
ORB [150]	256	USP	56.26	54.13	48.03	54.13	48.03	56.26		52.81	
DBD-MQ [41]	256	USP	31.1	57.24	25.78	57.15	27.2	33.11		38.59	
BinGAN [252]	256	USP	25.76	40.8	27.84	47.64	16.88	26.08		30.83	
DeepBit [106]	256	USP	33.83	54.63	20.66	56.69	28.49	34.64		38.15	
GraphBit [42]	256	USP	24.24	50.54	16.75	49.11	21.09	27.23		31.49	
UDBD	256	USP	18.99	52.6	11.76	52.17	14.61	20.79		28.49	

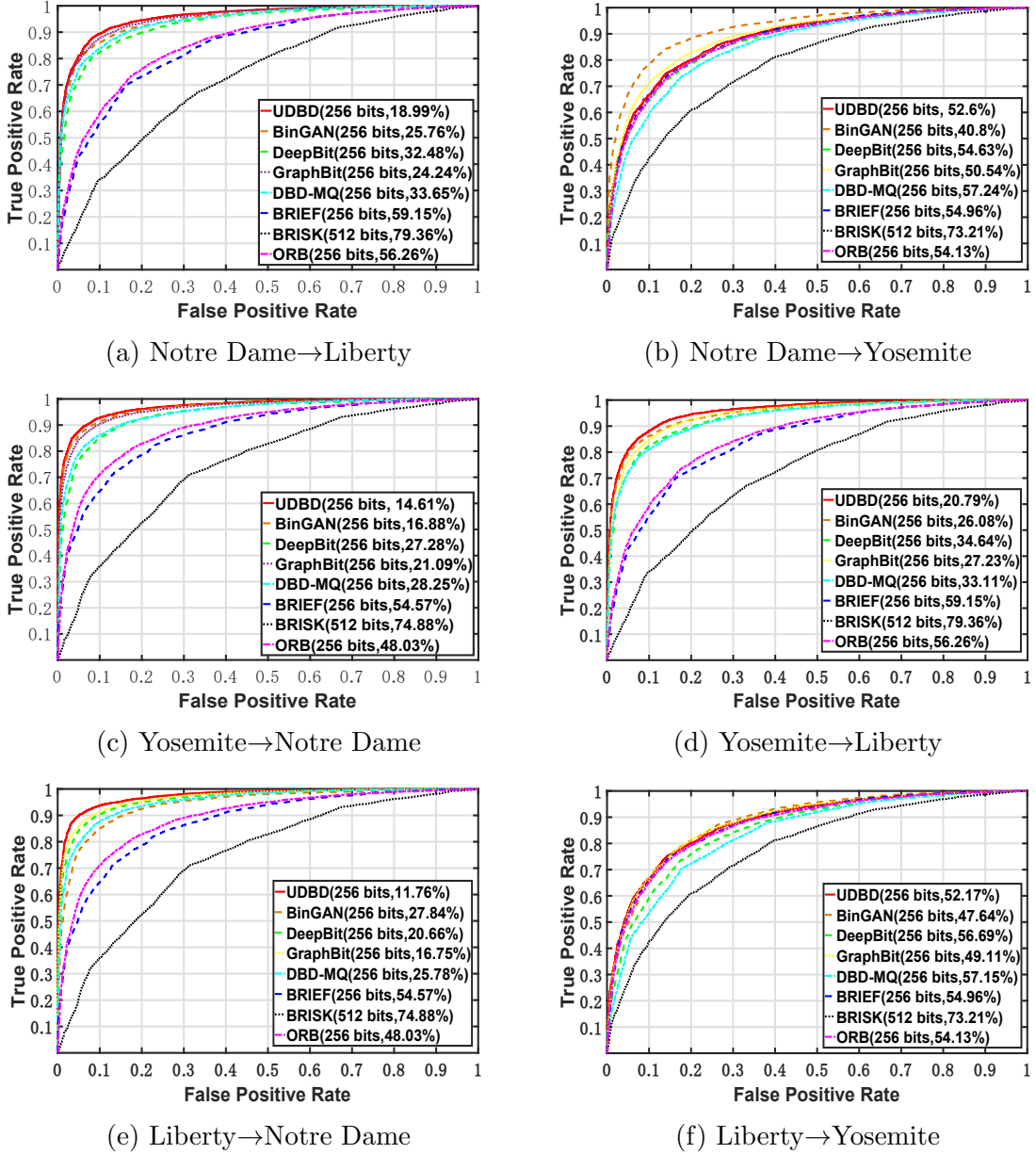


Figure 3.4: The ROC curves under different settings on Brown dataset when using various unsupervised binary descriptors. Best viewed in color.

baselines are categorized into unsupervised (e.g., BRIEF [17], GraphBit [42] and DeepBit [106], etc.) and supervised approaches (e.g., BinBoost [183], L2-Net [182], HardNet [130] and CDbin [231]). The results from floating-pointed (SIFT [121]) and supervised methods are provided as reference. Following [106] and [231], False Positive Rates at 95% (FPR@95%) from the cross-validations on three subsets are provided in Table 3.2. *Lower FPR@95% indicates better performance.* As can be seen, the proposed method outperforms unsupervised approaches significantly on most training

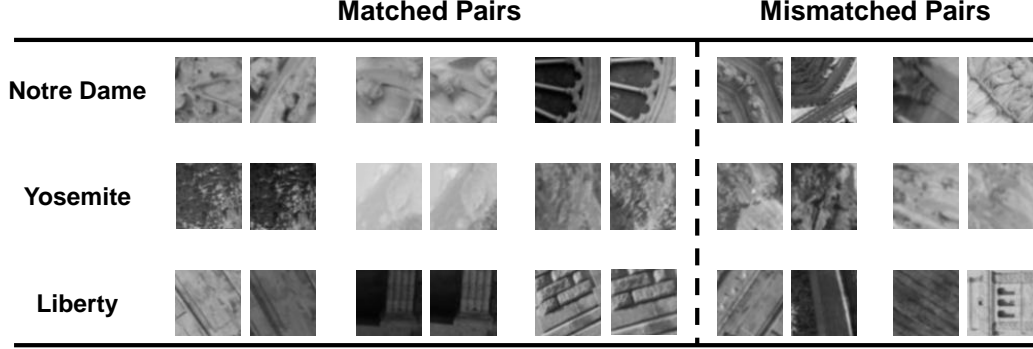


Figure 3.5: The matching results of *Notre Dame*, *Yosemite* and *Liberty* of UDBD on Brown dataset, which includes three matched pairs and two mismatched pairs for each subset.

and test configurations. Particularly, the error rates achieved by UDBD are 18.99%, 52.6%, 11.76%, 52.17%, 14.61% and 20.79% from bottom left to right. However, our method performs less favorable than BinGAN on *Yosemite*. The subset contains too many visually similar patches (e.g., snow and forest), which makes them difficult to be distinguished [106]. Nevertheless, UDBD still achieves the best average FPR@95% (28.49%) among unsupervised binary descriptors. Compared with the supervised methods, UDBD is highly competitive, where our method even has better result (11.76%) against BinBoost [183] (16.9%) on the setting of *Liberty* and *Notre Dame*.

Moreover, the ROC curves of those unsupervised descriptors on different subsets are plotted in Fig. 3.4 to further verify the above discussions. As shown in the figures, the ROC curves from UDBD rank at the top under most settings, which indicates its advantage over those unsupervised binary descriptors.

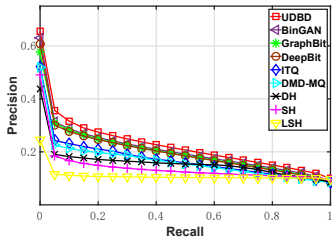
In Fig. 3.5, some matching results including three matched pairs and two mismatched pairs of UDBD under different settings: *Liberty*→*Notre Dame*, *Liberty*→*Yosemite* and *Notre Dame*→*Liberty*, on Brown [11] dataset are presented. As can be observed from Fig. 3.5, the proposed method enables to obtain the matched pairs of those visually-similar patches successfully.

3.3.3.2 Results on Cifar-10 Dataset

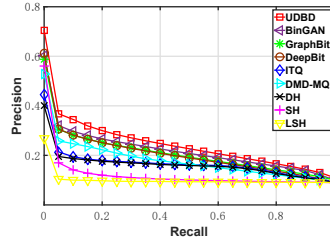
Without loss of generality, on Cifar-10 dataset, we first compare our method with several unsupervised binary descriptors regarding image retrieval performance, including the unsupervised binary descriptors and some classical hashing methods. The

Table 3.3: mAP of Top 1,000 (%) returned images at different code length from various unsupervised descriptors on Cifar-10 dataset. Bold values are the best results.

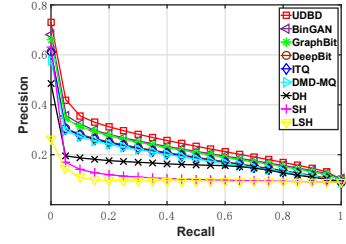
Method	mAP@1000 (%)		
	16 bits	32 bits	64 bits
LSH [2]	10.31	11.39	13.74
ITQ [51]	24.85	27.32	30.84
SH [152]	16.25	19.64	20.91
DH [43]	22.43	23.21	25.84
DBD-MQ [41]	21.53	26.5	31.85
BinGAN [252]	30.05	34.65	36.77
DeepBit [106]	26.36	27.92	34.05
GraphBit [42]	27.79	33.45	37.97
UDBD	32.24	36.17	39.6



(a) Cifar-10 at 16 bits



(b) Cifar-10 at 32 bits



(c) Cifar-10 at 64 bits

Figure 3.6: Precision-Recall curves of the proposed method and the baselines on Cifar-10 dataset at 16, 32 and 64 bits.

Table 3.4: Precision at Top 1 on Cifar-10 dataset when using DeepBit, BinGAN, GraphBit and UDBD at different bit sizes.

Method	Precision@Top 1(%)		
	16 bits	32 bits	64 bits
DeepBit [106]	24.38	32.51	39.74
BinGAN [252]	33.72	41.48	44.31
GraphBit [42]	32.12	41.39	46.79
UDBD	38.46	46.63	52.06

retrieval performance is evaluated under mean average precision (mAP) at top 1,000 returned images, which is detailed in Table 3.3 at the code length of 16, 32 and 64. As observed from Table 3.3, our method improves the mAP@1000 values by 2.19%, 1.52% over BinGAN on 16 and 32 bits, while 1.63% on 64 bits over GraphBit. In Fig. 3.7, the top-10 retrieved results of UDBD at 64 bits on Cifar-10 [89] dataset are presented. In particular, four categories: *cat*, *car*, *airplane* and *bird*, are included. As

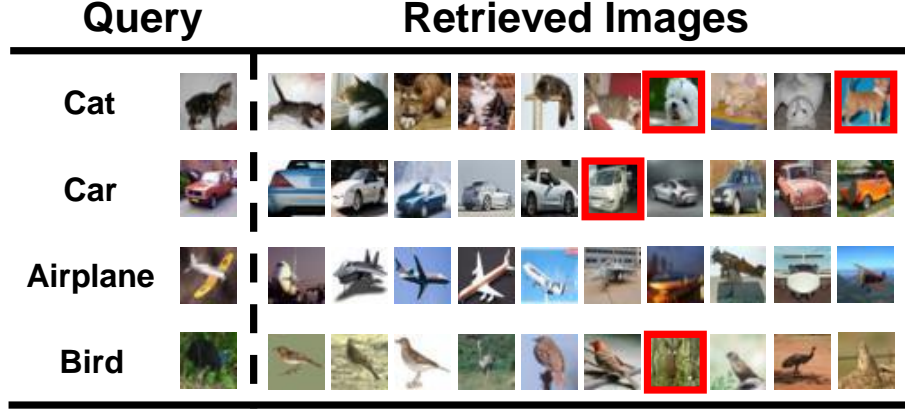


Figure 3.7: The top-10 retrieved images on Cifar-10 dataset by UDBD at the code length of 64. The query images are selected from four categories: cat, car, airplane and bird. Red rectangle indicates the wrong results.

can be seen, UDBD can achieve accurate retrieval results with minor errors in most cases. Note that the small image size (32×32) gives rise to the blurred figure.

Moreover, we provide the Precision-Recall curves on Cifar-10 dataset at different code lengths in Fig. 3.6, where the results are consistent with the above discussions. Additionally, the matching performance measured by Precision@Top 1 returned candidate from the state-of-the-arts are also provided in Table 3.4, where the image is treated as a big patch. As can be seen, the proposed method obtains the highest values in term of Precision at top 1, at least 4.74% higher than the most competitive baseline, which consolidates the contribution on improving the matching accuracy from the proposed method.

3.3.3.3 Results on HPatches Dataset

Finally, we report mAP values from the three visual tasks: matching, retrieval and verification, on HPatches dataset to provide broader insights on the binary descriptor performance. Specifically, the matching is conducted by comparing patch sets between a reference image and a target one. The retrieval aims at finding similar patches for each query and the verification is to classify whether two patches are matched or not [3, 231]. Following the evaluation protocols suggested in [3, 231], the results on the full split are summarized in Table 3.5. We compared UDBD with several unsupervised binary descriptors and provided the results of SIFT [121], BinBoost [183], L2-Net [182] and CDbin [231] for references. Table 3.5 shows that UDBD

Table 3.5: Comparison of the proposed UDBD to the state-of-the-art descriptors in terms of mAP (%) on HPatches dataset. Dim, SP and USP denote dimension, supervised and unsupervised, respectively. The real-valued descriptor (SIFT) and the supervised methods are provided as references. Bold values are the best results in unsupervised binary descriptors.

Method	Dim	Type	Matching	Retrieval	Verification
SIFT [121]	128	USP	25.47	31.98	65.12
BinBoost [183]	64	SP	14.77	22.45	66.67
L2-Net [182]	128	SP	30.89	41.29	70.58
CDbin [231]	128	SP	39.76	46.19	82.68
BRIEF [17]	256	USP	10.5	16.03	58.07
ORB [150]	256	USP	15.32	18.85	60.15
DBD-MQ [41]	256	USP	13.45	23.56	63.43
DeepBit [106]	256	USP	13.05	20.61	61.27
GraphBit [42]	256	USP	14.22	25.19	65.19
UDBD	256	USP	17.27	28.88	69.77

outperforms the most competitive GraphBit by 3.05%, 3.69% and 4.58% on matching, retrieval and verification, respectively, which indicates the superiority of UDBD in generating effective binary descriptors for various visual tasks.

Based on above discussions, the proposed method can achieve superior performance against most existing binary descriptors. Compared to the real-valued descriptor (e.g., SIFT), the results from UDBD are still highly competitive. Moreover, the dominant advantage on high matching speed from binary descriptors, over 100x faster than real-valued descriptors in terms of matching time per pair [106], makes UDBD suitable for large-scale similarity search applications.

3.3.4 Further Analysis

In this section, further insights are provided to address some key features in our proposed method.

3.3.4.1 Ablation Study

Firstly, the comprehensive analysis on the involved components: view weighting scheme and Laplacian constraint, during the code learning is provided in Table 3.6. Particularly, two different settings: $\gamma = 0$ (i.e., $\text{UDBD}_{\gamma=0}$: NO view weighting scheme) and $\beta = 0$ (i.e., $\text{UDBD}_{\beta=0}$: NO graph loss term), are investigated on

Table 3.6: Ablation study on Brown (FPR@95%): *Liberty*→*Notre Dame* and *Yosemite*→*Liberty*, HPatches: *matching* (mAP) and Cifar-10 at 64 bits (mAP@1000) when $\gamma = 0$ (i.e., $\text{UDBD}_{\gamma=0}$) and $\beta = 0$ (i.e., $\text{UDBD}_{\beta=0}$).

Method	Brown [11]		Cifar-10 [89]	HPatches [3]
	Liberty→Notre Dame	Yosemite→Liberty	64 bits	Matching
$\text{UDBD}_{\gamma=0}$	14.18	23.62	35.33	14.24
$\text{UDBD}_{\beta=0}$	12.92	22.36	34.08	14.95
UDBD	11.76	20.79	39.6	17.27

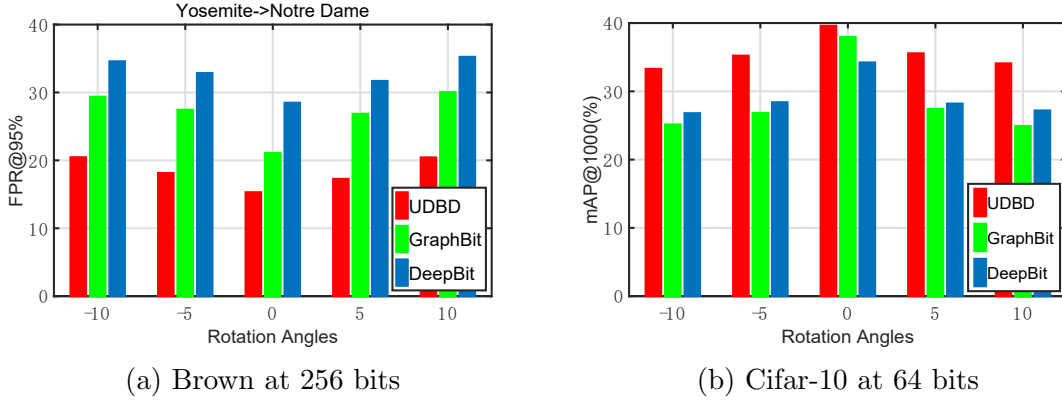


Figure 3.8: Performances variations under different rotation angles on test instances from GraphBit, DeepBit and UDBD.

various datasets. For instance, mAP@1000 result at 64 bits on Cifar-10 when $\beta = 0$ would decrease dramatically to 34.08%, and that with $\gamma = 0$ is 35.33%, which are far below than the original value (39.6%) achieved by UDBD in Table 3.3. That indicates the importance and necessity of the involved graph loss term and view weighting scheme in the proposed framework, respectively.

3.3.4.2 Transformation Invariance

Then we investigate the performance variations: FPR95% on matching and mAP@1000 on retrieval, under certain affine transformation imposing on test images, where rotation is given as an example as plotted in Figure 3.8. The variations are calculated at 64 bits from GraphBit, DeepBit and UDBD. Large rotation angles usually reduce visual similarity on the images, thus yielding worse performance [106]. However, UDBD still outperforms the others significantly at all angle ranges. Particularly, mAP@1000 for UDBD is 34.1% when rotating 10 degrees, which is much higher than those achieved by DeepBit (27.26%) and GraphBit (23.51%). That indicates the proposed binary descriptor is more robust to rotation. More analysis on other transformations (e.g., scaling, translation and occlusion) will be made in the future work.

3.3.4.3 Weak Bit Study

Moreover, the impact of weak bit scheme on the system performance is investigated in Fig. 3.9 and Table 3.7, under three measurements as FPR@95%, Precision@Top 1 and mAP on different datasets. As can be seen, noticeable performance gains have been achieved with the weak bit scheme on Brown and Cifar-10 datasets, especially when using shorter codes. For instance, Precision@Top 1 is 46.63% (with weak bit) and 42.33% (without weak bit) on Cifar-10 using 32 bits. Slight improvements also have been achieved when tackling three tasks (1.44%, 0.13% and 1.3%) at 256 bits on HPatches by applying weak bit scheme. The results show that the proposed weak bit scheme plays vital role in improving the matching performance, which further verifies the claimed contribution.

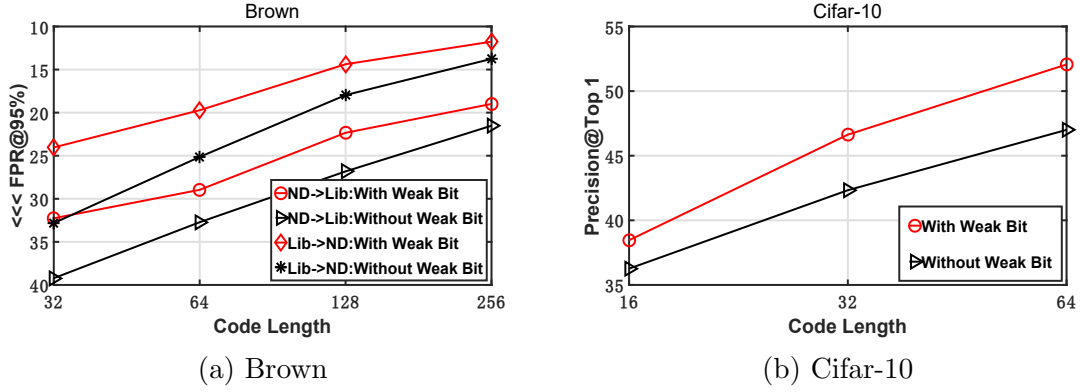


Figure 3.9: Performance variations at varying code lengths with/without using weak bit scheme. (a) FPR@95% on Brown: *Notre Dame* (ND)→*Liberty* (Lib) and *Liberty*→*Notre Dame*; (b) Precision@Top 1 on Cifar-10.

Table 3.7: mAP variations on HPatches with/without using weak bit scheme (UDBD[‡]/UDBD[†]). Bold values show the best results.

Method	Matching	Retrieval	Verification
UDBD [†]	15.83	28.75	68.47
UDBD [‡]	17.27	28.88	69.77

3.3.4.4 Loss Term

Then we report the performance variations when using different loss terms (i.e., $\ell_{2,1}$ -norm *vs* $\ell_{2,2}$ -norm) in the code learning process, as shown in Table 3.8. For example, on *Liberty*→*Notre Dame*, FPR@95% is 15.81% under $\ell_{2,2}$ -norm, which is 4.05% lower than 11.76% when applying $\ell_{2,1}$ -norm. Generally, $\ell_{2,1}$ -norm loss yields better results

Table 3.8: Performance variations on Brown (FPR@95%): *Notre Dame*→*Liberty* and *Liberty*→*Notre Dame*, HPatches: *matching* (mAP) at 256 bits, and Cifar-10 at 32 bits (mAP@1,000) when using $\ell_{2,1}$ -norm and $\ell_{2,2}$ -norm loss terms.

Loss Term	Brown [11]		Cifar-10 [89]	HPatches [3]
	Notre Dame→Liberty	Liberty→Notre Dame	32 bits	Matching
$\ell_{2,2}$ -norm	22.51	15.81	34.24	14.81
$\ell_{2,1}$ -norm	18.99	11.76	36.17	17.27

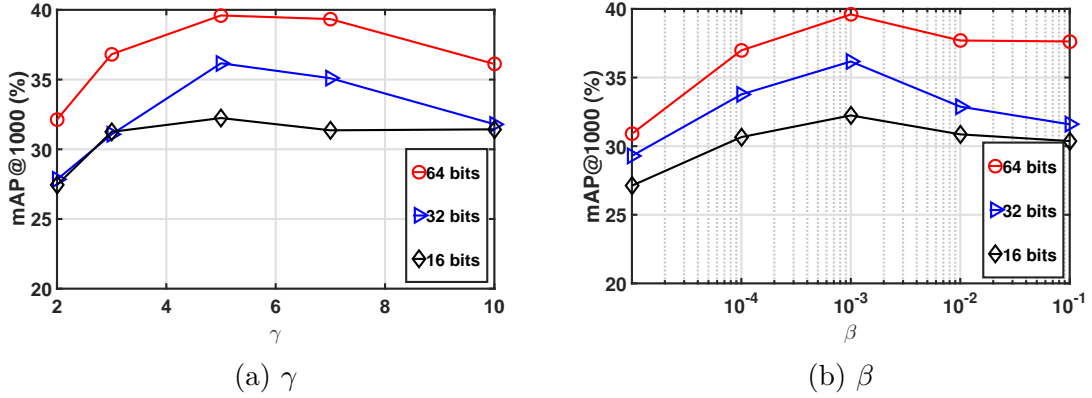


Figure 3.10: Parameter sensitivity analysis of γ and β at various bit sizes on Cifar-10 dataset.

compared to the widely used $\ell_{2,2}$ -norm, which are consistent with the previous discussions on $\ell_{p,q}$ -norm based similarity search and other regularizers even obtain worse performance [56].

3.3.4.5 Parameter Analysis

Finally, more experiments are conducted on Cifar-10 as examples in the retrieval performance analysis with varying hyperparameters (γ and β), as shown in Fig. 3.10. γ and β are varied in wide ranges from $\{2, 3, 5, 7, 10\}$ and $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$, where the best performance is given around the setting of 5 and 10^{-3} . It is worth noting that the performance degrades heavily when small β is being set, which inevitably weakens the impact of the graph constraint learning, thus yielding worse code quality.

3.4 Chapter Summary

In this chapter, a learning-based unsupervised binary descriptor termed UDBD is proposed to facilitate large-scale visual recognition. Particularly, the binary descriptor is learned via exploiting the common binary space between the original and transformed data sets. With $\ell_{2,1}$ -norm loss as regularization term, the learned descriptor is highly

robust to potential outliers. An unsupervised graph constraint is further employed to preserve the original manifold structure in the code learning, thus improving the code quality dramatically. Then the discrete and $\ell_{2,1}$ -norm constrained objective function is solved directly without relaxation following an alternating optimization strategy. Additionally, a weak bit scheme is used to address the ambiguous matching issue and further boost the matching performance of the proposed binary descriptor in the online search stage. Experiments on several public datasets show that UDBD outperforms the state-of-the-arts significantly.

The presented method focuses on feature matching tasks using the local binary descriptor, which is a classical similarity search application in the single-modality domain. In the next chapter, we continue the research topic in this domain but tackle a more challenging retrieval task, namely video-to-video retrieval, as discussed in Section 1.3.2. The video, which consists of a series of consecutive frames, can be viewed as a more advanced and highly redundant 3D signal against a single image. We aim at proposing a novel video hashing framework for efficient large-scale video retrieval, where the details will be revealed in the following chapter.

Chapter 4

Unsupervised Deep Video Hashing

4.1 Introduction

With the fast development of Internet and mobile communication technologies, recent decades have witnessed the explosive growth of massive online information. Consequently, Approximate Nearest Neighbor (ANN) search based on hashing techniques has attracted substantial attentions in the research field of efficient similarity search [51, 78, 107, 197, 221]. At the core of hashing-based visual search is how to generate a compound hash function that is able to project high-dimensional floating-point features into the compact binary Hamming space with vital properties from the original data preserved.

According to the prior arts, generating binary codes directly from the original feature is usually an NP-hard problem [72, 118, 195, 211, 238]. Most existing methods adopt a two-stage learning framework, consisting of projection stage and quantization stage [58]. At the projection stage, certain linear projection functions, which are usually learned from the original data to preserve important properties, such as global Euclidean structure or manifold structure, are built and such functions, in turn, are exploited to project the data from the original feature space to a low-dimensional compact space [35, 57]. At the quantization stage, the real-value feature representation on that space is quantified into binary codes by arbitrary thresholding [51, 87]. With effective projections, such a framework has achieved promising results to some extent. Unfortunately, it has been acknowledged that the performance of those hashing schemes may degrade significantly if the projected dimensions are imbalanced [87, 118, 221], i.e., their variances vary a lot. The major reason is that the equal-length bit allocation adopted in the quantization stage (e.g., 1 bit in most cases, and 2 bits in a Double-bit Quantization [87] ends up with a suboptimal situation that dimensions with lower variances, which contain less information, will have the same

influence on Hamming distance computing as high-variance dimensions. The problem mentioned above is illustrated in Fig. 4.1(a), taking the 2-bit quantization in the two-dimensional feature space as an example. As can be seen, the X axis of the data point obviously contains more information than the Y axis in the original feature space, but both of them are quantized with 1 bit in the Hamming space. This implies that those dimensions containing various amounts of information equally contribute to the calculation of Hamming distance, which is likely to make the quantization intractable.

Recent studies have shown that the spatio-temporal features combining the frame-level spatial information and the temporal information of a video sequence make great contribution in boosting the expressive capability of video representation [175, 226, 235]. Currently, the methods that generate deep video features basically follow two pathways: 1) feeding the sparsely-sampled frame-level image features from video clip into Recurrent Neural Network (RNN) or Long-Short Term Memory (LSTM) [73] in order to explore the temporal nature and then aggregating into the global video-level feature [109, 197, 217]; 2) fusing the spatial and temporal features (e.g., optical flow) from the two-stream networks to generate the unified video representation via various pooling schemes [200]. However, we argue that those methods will yield the imbalanced video representation inevitably due to the following reasons. Firstly, the sparse-sampling strategy is usually adopted when tackling video representation learning to reduce the training costs [95]. The contents within those frames are less correlated, especially for the cases of long videos, implying the features in terms of distributions are quite different [207, 243]. Therefore, the imbalanced variance distributions within dimensions can be accumulated when fusing those frame features directly in the video representation learning. There is no evidence that those temporal encoders (e.g., RNN or LSTM) can alleviate such a problem. Moreover, the imbalanced situation is even getting worse when making the video-level evaluation with the aggregation of spatial and temporal features in most two-stream based works, which leads to huge variance fluctuations within dimensions. A reasonable explanation is that the temporal network is actually designed to capture different visual aspects of videos compared to the spatial network, which indicates that the feature distributions are more diverse between those fields (e.g., optical flow and RGB) [86, 142, 200]. The arbitrary pooling scheme will aggravate the problem of imbalanced video features. Here, we plot the curves of variances within each dimension of image and video features on ActivityNet [14] in Fig. 4.1(b) to evidentiate the above discussions, where image features are extracted via a spatial ConvNet [200] and video features are

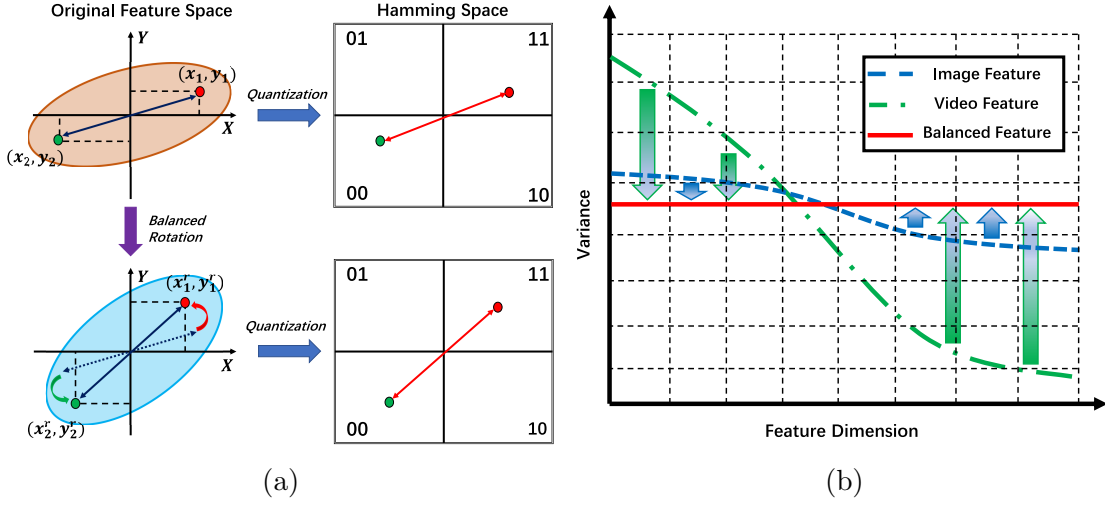


Figure 4.1: (a) Suppose that two data samples (red and green) from a benchmark are projected into a two-dimensional feature space with the coordinates of (x_1, y_1) and (x_2, y_2) and encoded by two bits subsequently, where $|x_1| > |y_1|$, $|x_2| > |y_2|$ and $|x_1 - x_2| > |y_1 - y_2|$. After the proposed balanced rotation, the coordinates of two data points change to (x_1^r, y_1^r) and (x_2^r, y_2^r) accordingly, where $|x_1^r| = |y_1^r|$ and $|x_2^r| = |y_2^r|$. Obviously, compared to the original features, the variances contained in the X and Y axis can be balanced with such rotation strategy applied. That indicates two dimensions of the rotated data points will have the same impact on the calculation of the Hamming distances when encoding them with the fixed number of bits; (b) The variance within each dimension of image and video feature.

generated via fusing the output vectors of two sub ConvNets in temporal segment networks [200]. As can be observed from Fig. 4.1(b), the large variance variations do exist within each dimension of video features. However, most existing video hashing methods usually concentrate on selecting the appropriate fusion strategies that can wrap frame-level features up as a single video feature so that the binarization tactics existed in image hashing frameworks can be applied directly [109, 197, 200, 225], regardless of the imbalanced features. This would considerably degrade the quality of video hash codes.

In view of the above analysis, it is clear that such an imbalanced problem needs to be addressed carefully in designing video hashing method [186], because applying image feature binarizations to video features directly is suboptimal in producing effective video hash codes. To the best of our knowledge, this is the first attempt to tackle the imbalanced problem when binarizing *video* features. Specifically, we propose a novel hashing framework termed **Unsupervised Deep Video Hashing (UDVH)**, which distinguishes from the existing methods in three main aspects:

- An unsupervised deep hash framework termed Unsupervised Deep Video Hashing (UDVH) is proposed to organize the hash code learning in a self-taught manner. Instead of minimizing feature reconstruction distortion [240], our framework minimizes the quantization error of projecting video features to a binary hypercube, thus allowing the feature extraction and hash function learning to engage with each other. Involving the feature clustering in the code learning enables the neighborhood structure to be preserved. To solve the objective function, a novel scheme is proposed, where the rotation matrix, binary code generation, and the deep framework parameters are jointly optimized.
- During the code learning, a balanced rotation designed for video features is proposed to identify a proper projection matrix such that the variance of each projected dimension can be balanced (see Fig. 4.1). By doing so, the information in each dimension of video features can be equalized. This would greatly benefit the quantization step, in which each dimension is allocated with the same number of bits. An in-depth analysis of the balanced rotation and other related works is provided.
- Two different video feature learning structures: stacked LSTM units (UDVH-LSTM) and Temporal Segment Networks (UDVH-TSN), are investigated to validate the effectiveness of the proposed framework. Comprehensive experimental study has been carried out on three publicly available video datasets: FCVID [140], YFCC [181] and ActivityNet [14]. The experimental results demonstrate various advantages of UDVH compared to existing video hashing approaches.

The rest of this chapter is organized as follows: Section 4.2 introduces the proposed UDVH. The experimental results are given and analyzed comprehensively in Section 4.3. The paper is concluded with detail summary and introduction of future work in Section 4.4.

4.2 Proposed Unsupervised Deep Video Hashing

Given a benchmark data set that contains N videos, 20 keyframes are selected averagely for each video. Our goal is to exploit the deep hash function $\mathcal{F}(\cdot)$ that encodes those videos into k -bit binary representation as $\mathbf{B} \in \{-1, +1\}^{N \times k}$. Particularly, the deep networks are treated as to-be-learnt binary encoding functions, defined as

Table 4.1: Mathematical symbols and their short descriptions.

Symbol	Description	Symbol	Description
\mathbf{Z}	input stream	N	training video number
\mathbf{Z}	feature matrix from FC7	t	training loops
\mathbf{Y}	pseudo labels	\mathbf{R}	rotation matrix
\mathbf{B}	binary code	k	code length
\mathbf{P}	projection matrix	C	cluster number
\mathbf{H}	video feature matrix	v	variance
$\mathcal{F}(\cdot)$	deep hash function	s	standard deviation
\mathcal{M}	feasible set	Θ	network parameters
\mathcal{E}	tangent space	\mathbf{Q}	Cayley matrix
\mathbf{W}	skew-symmetric matrix	\mathbf{G}	upgradient
\mathbf{D}	diagonal matrix	r	iterations in solving \mathbf{R}

$\mathcal{F}(\mathbf{Z}; \Theta)$, where \mathbf{Z} represents the input that could be in the form of feature matrix or original images. The hash function is parameterized by network parameters Θ including weights and biases. The mathematical symbols used in this chapter are summarized in Table 4.1 for the ease of explanation.

4.2.1 Deep Video Feature Learning

In this section, we adopt two different deep architectures: LSTM [73] and TSN [200] to generate the video features, termed as UDVH-LSTM and UDVH-TSN independently in the next subsections.

4.2.1.1 UDVH-LSTM

LSTM [73] has been widely deployed in the sequence study for its powerful ability in dealing with the sequential inputs. This characteristic makes LSTM very popular in video representation learning, video summarization, video captioning, and speech recognition [151, 175]. Compared with the traditional Recurrent Neural Network (RNN), memory blocks: input gate, output gate and forget gate, are integrated into LSTM, which enable to store useful information such that the long-range temporal relationship could be exploited [26, 73, 175]. Particularly, the input gate controls the flow of input activations into the memory cell and the output gate controls the output flow of cell activations into the rest of the network. The forget gate decides what kind of information should be removed from the cell state to adjust (forget or reset) the

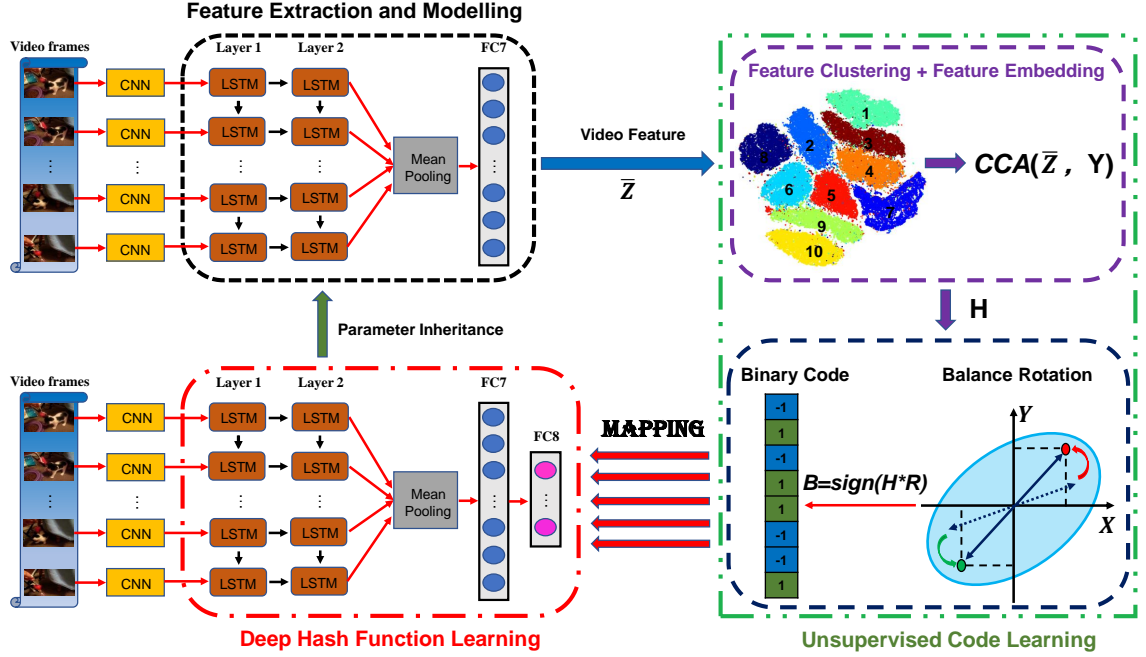


Figure 4.2: The basic framework of UDVH-LSTM. The whole process consists of three subsections: video feature extraction, unsupervised code learning and deep hash function learning, which are performed iteratively to obtain the solution.

cell’s memory adaptively [48]. The final output of a single LSTM unit is actually depended on the sequential input, the previous and current cell states.

The proposed framework of UDVH-LSTM is given in Fig. 4.2. Two-layer stacked LSTM units are constructed to tackle the frame-level features extracting from the pre-trained VGG-19 model [164] for the temporal-awareness. It also allows the sequential inputs to go through more nonlinear computations at every step to obtain better feature representation [48]. In this case, the frame-level features of those training videos are first encoded by the stacked LSTM units sequentially. Then the encoded frame features are averagely fused by the connected FC7 layer to generate the video feature matrix $\bar{\mathbf{Z}} \in \mathbb{R}^{N \times 1024}$ for the upcoming code learning. Thus, a robust representation for videos can be obtained by taking both spatial and temporal information into account. FC8 layer is added at the end of FC7 as the hash layer. In this case, the deep hash function is defined as $\mathcal{F}(\mathbf{Z}; \Theta)$, where \mathbf{Z} denotes the frame-level CNN features. The deep parameters, including weights and biases from the fc layers (FC7 and FC8) and the stacked LSTM units for UDVH-LSTM, are represented by Θ uniformly for the concise description.

4.2.1.2 UDVH-TSN

Compared to LSTM, TSN evaluates the video-level representation with the two-stream ConvNets networks [200] consisting of spatial and temporal networks. Here, spatial networks collect the spatial information of each image, whereas temporal networks model the relationships of successive frames over time with the optical flow fields provided. By fusing these two types of features, TSN ends up exploring both spatial and temporal information to represent videos. To address the over-fitting problem when training deep convolution neural networks on small datasets, several strategies are adopted in training TSN: 1) Cross modality Pre-training is introduced in the initializations of two ConvNets, where Pre-trained models on ImageNet and RGB frames are utilized as the initial models for spatial and temporal ConvNets, respectively; 2) Compared to the shallow structure in [163], TSN adopts the Inception with Batch Normalization (BN-Inception) [79] in the network structure, which speeds up the training process significantly by converting the activations of the mean and variance in each batch into the standard Gaussian distribution. In the meanwhile, a good trade-off between algorithm accuracy and efficiency can be achieved by re-evaluating those values in the certain layers with the proposed partial BN [200]; 3) they employ two new methods: corner cropping and scale jittering, in the data augmentation to avoid the severe overfitting problems in traditional two-stream based networks. To make it compatible with the proposed framework, we modify the traditional TSN presented in [200], where the last *fc-action* layers of the original TSN are removed and the pooled features from the *global-pool* layers in spatial and temporal ConvNets are fused averagely. Afterward, such features are fed to two *fc* layers (FC7 and FC8) like UDVH-LSTM, thus constructing the network architecture of UDVH-TSN. The proposed framework is shown in Fig. 4.3.

As discussed in the previous descriptions, we define the to-be-learned deep hash function as $\mathcal{F}(\mathbf{Z}; \Theta)$, where the deep parameters including weights and biases from the *fc* layers (FC7 and FC8) and modified TSN for UDVH-TSN are represented by Θ uniformly for the concise description. \mathbf{Z} denotes the input streams, which are the original RGB frames and optical flows (See Section 4.3.3). Those network parameters will be updated automatically during each iteration of hash function learning, thus producing the desirable outputs from the last *fc* layer with the dimension of k after the training process. Finally, the binary representation for the input can be obtained by calculating $\text{sign}(\mathcal{F}(\mathbf{Z}; \Theta))$.

The basic structures of UDVH-LSTM and UDVH-TSN are summarized in Table 4.2. The only difference between two networks is that they adopt LSTM and TSN

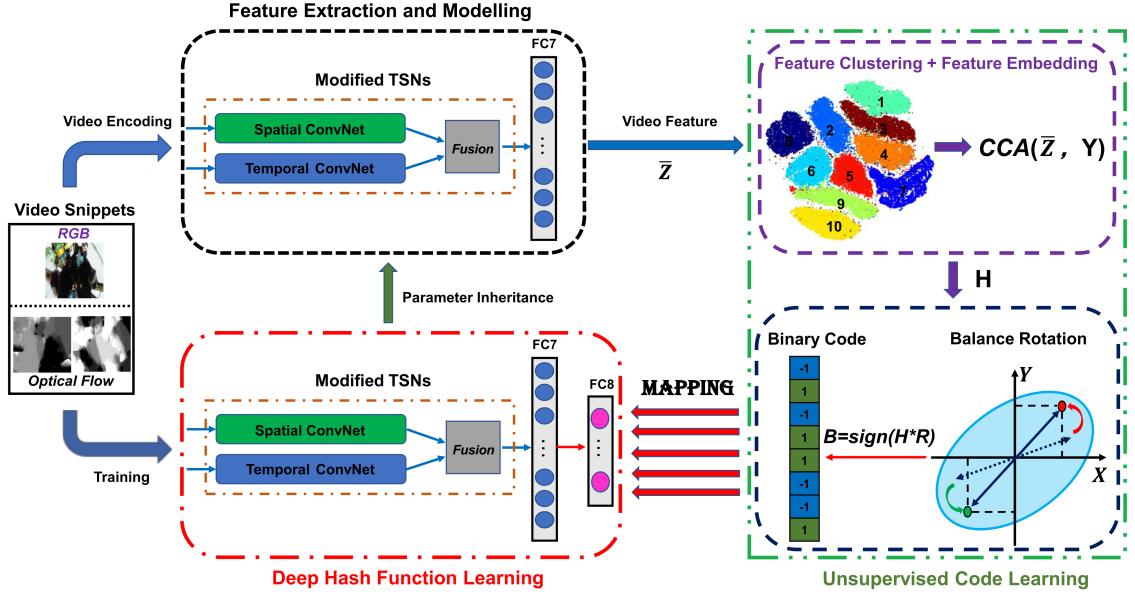


Figure 4.3: The basic framework of UDVH-TSN. The only difference between UDVH-TSN and UDVH-LSTM is that they adopt LSTM and TSN separately to evaluate the video representation.

Table 4.2: The network configurations UDVH-LSTM and UDVH-TSN. Other layers like pooling and activation are omitted for concise descriptions.

Method	Network Configuration
UDVH-LSTM	CNN (VGG-19 [164]), 2-layer stacked vanilla LSTM units, fusion layer, 2 fully-connected layers
UDVH-TSN	Modified TSNs including spatial and temporal ConvNets [200], fusion layer, 2 fully-connected layers

separately to evaluate the video representation. In the following methodology part, these two networks are represented uniformly to illustrate the proposed algorithm.

4.2.2 Feature Embedding with Pseudo Labels

At the beginning of the proposed unsupervised code learning, we first roughly estimate the feature distribution via exploring the pseudo labels, which aims at improving the effectiveness of the feature embedding afterwards [241]. Particularly, k-means is adopted to categorize the video features $\bar{\mathbf{Z}} \in \mathbb{R}^{N \times 1024}$ from FC7 layer such that the similar videos tend to be classified into the same category. To accelerate the clustering process in the experiment, we randomly sample 10,000 video features from \mathbf{Z} to generate C centroids, where C is the cluster number. Then a 1-of- C vector (C dimensions with one 1 and $C - 1$ 0s) can be assigned to each video by measuring the

smallest l_2 -norm distance among the corresponding feature and those centroids [244]. Such dedicated pseudo labels $\mathbf{Y} \in \{0, 1\}^{N \times C}$ are generated for the whole training set during each iteration, thus preserving the original neighborhood structure to the maximum extent in the following process of the dimensionality reduction.

Subsequently, the dimensionality of the video features $\bar{\mathbf{Z}}$ is reduced into the required code length (k) via Canonical Correlation Analysis (CCA) to obtain the projected video feature matrix $\mathbf{H} \in \mathbb{R}^{N \times k}$, where the correlation between video features and the corresponding pseudo labels are maximized and well preserved in the low-dimensional space. In contrast to Principle Component Analysis (PCA) [213], CCA is more effective in extracting discriminative information with the robustness in anti-noise [51], thus obtaining more discriminative video features after the projection. We denote the projection matrix as $\mathbf{P} \in \mathbb{R}^{1024 \times k}$ and $\bar{\mathbf{Z}} \in \mathbb{R}^{N \times 1024}$ as the video features from FC7 layer, where \mathbf{P} can be pre-trained with $\bar{\mathbf{Z}}$ and \mathbf{Y} during CCA. According to the above illustrations, the process of the proposed feature embedding can be simplified as:

$$\mathbf{H} = \bar{\mathbf{Z}} \times \mathbf{P} = \bar{\mathbf{Z}} \times CCA(\bar{\mathbf{Z}}, \mathbf{Y}), \text{ s.t. } \bar{\mathbf{Z}} \in \mathbb{R}^{N \times 1024}, \mathbf{Y} \in \{0, 1\}^{N \times C}. \quad (4.1)$$

However, similar to the conventional methods that reduce the feature dimensionality via projection schemes, CCA will also concentrate most information on a few top eigenvectors, thus unbalancing the projected data and lowering the hash code quality drastically [221]. Consequently, we propose a novel rotation matrix to alleviate this imbalanced issue, which will be elaborated in the next subsection.

4.2.3 Balanced Rotation

As shown in Fig. 4.1(b), the video features are more dispersed in terms of distribution compared to image features, which results in larger variance fluctuations among dimensions. However, in most previous works that allocate the same number of bits to encode such imbalanced features, the dimensionality containing less information (low variance) will make the same contribution on calculating the Hamming distance as those with rich information (high variance), thus significantly reducing the quality of the hash code [87, 118, 221]. To solve the problem described in Fig. 4.1(a), this paper proposes a novel balanced rotation that balances the information within each dimension of video features before undergoing the quantization. The detailed formula for the proposed rotation is presented as follows. Firstly, the effect of rotation on the variance of data is investigated. Given the optimal projection \mathbf{P} , the

projected data $\mathbf{H} = \bar{\mathbf{Z}}\mathbf{P} \in \mathbb{R}^{N \times k}$ can be calculated by Eq. (4.1). Assuming the video features are zero-centralized, i.e., $\sum_{i=1}^N \bar{\mathbf{Z}}_i = 0$, the variance of the j -th dimension is $v_j = \frac{1}{N} \sum_{i=1}^N \mathbf{H}_{ij}^2$. Given an orthogonal rotation matrix \mathbf{R} and the adjusted data $\mathbf{H}_r = \mathbf{H}\mathbf{R}$, the new variance of each dimension is updated as v_j' , where $\mathbf{R}\mathbf{R}^T = \mathbf{I}_k$ and $\mathbf{I}_k \in \mathbb{R}^{k \times k}$ is the identity matrix. Then we obtain the following lemma.

Lemma 1: *The sum of variances on all dimensions is invariant after rotation for the centralized data.*

Proof to Lemma 1: Particularly, the sum of variances in all dimensions after rotation can be calculated as below:

$$\begin{aligned}
 N \sum_{j=1}^k v_j' &= \sum_{j=1}^k \sum_{i=1}^N (\mathbf{H}\mathbf{R})_{ij}^2 = \text{tr}(\mathbf{H}\mathbf{R}\mathbf{R}^T\mathbf{H}^T) = \text{tr}(\mathbf{H}\mathbf{H}^T) = \|\mathbf{H}\|_F^2 \\
 &= \sum_{j=1}^k \sum_{i=1}^N (\mathbf{H})_{ij}^2 \quad (4.2) \\
 &= N \sum_{j=1}^k v_j,
 \end{aligned}$$

where $\text{tr}(\cdot)$ denotes the trace norm. As observed from the above equation, the total variance remains unchanged with the original properties (e.g., global Euclidean or local manifold structure) preserved after the orthogonal rotation [51, 81]. The purpose of the proposed rotation scheme is to balance the variance of each dimension in the rotated data \mathbf{H}_r , where the degree of balance can be measured by the variance of standard deviation (VSD) of each dimension. *Theoretically, smaller VSD implies more balanced data.* If all dimensions have the same data variance or standard deviation, the VSD is 0. Denote the standard deviation (SD) of the j -th dimension as $s_j = \sqrt{v_j}$, the VSD is computed as:

$$\text{VSD} = \frac{1}{k} \sum_{j=1}^k (s_j - \bar{s})^2, \quad (4.3)$$

where \bar{s} is the mean of SD. Hence, the goal of balancing the variances can be achieved by finding a rotation that minimizes the VSD. However, it is not intuitive by solving Eq. (4.3) directly. This problem can be simplified because of:

Lemma 2: *Minimizing the VSD of the data by a rotation is equivalent to maximizing the sum of standard deviation (SSD).*

Proof to Lemma 2: Based on the Lemma 1, we have $\sum_{j=1}^k s_j^2 = \sum_{j=1}^k v_j = c$, where c is a constant. Then, the VSD of the data in Eq. (4.3) can be further expanded as:

$$\begin{aligned}
 \text{VSD} &= \frac{1}{k} \sum_{j=1}^k (s_j - \bar{s})^2 = \frac{1}{k} \sum_{j=1}^k s_j^2 + \bar{s}^2 - \frac{2}{k} \sum_{j=1}^k s_j \bar{s} \\
 &= \frac{c}{k} + \bar{s}^2 - 2 \left(\frac{1}{k} \sum_{j=1}^k s_j \right) \bar{s} \\
 &= \frac{c}{k} - \bar{s}^2 \\
 &= \frac{c}{k} - \left(\frac{1}{k} \sum_{j=1}^k s_j \right)^2 \\
 &= \frac{c}{k} - \frac{1}{k^2} \text{SSD}^2.
 \end{aligned} \tag{4.4}$$

From Lemma 2, it is clear that minimizing VSD equals maximizing SSD, where the SSD can be further expressed as the matrix form:

$$\text{SSD} = \sum_{j=1}^k s_j = \sum_{j=1}^k \sqrt{\frac{1}{N} \sum_{i=1}^N \mathbf{H}_{ij}^2} = \frac{1}{\sqrt{N}} \|\mathbf{H}^T\|_{2,1}, \tag{4.5}$$

where $\|\cdot\|_{2,1}$ is the $l_{2,1}$ -norm of \mathbf{H}^T . Considering Eq. (4.2), Eq. (4.4) and Eq. (4.5) jointly, the rotation matrix which aims at balancing the variance of each dimension can be learned from the elegant optimization formulation:

$$\max_{\mathbf{R}} \|\mathbf{R}^T \mathbf{H}^T\|_{2,1}, \text{ s.t. } \mathbf{R} \mathbf{R}^T = \mathbf{I}_k, \tag{4.6}$$

where $\mathbf{I}_k \in \mathbb{R}^{k \times k}$ is the identity matrix. Eq. (4.6) is also the objective function of the proposed balanced rotation and the corresponding optimization process regarding \mathbf{R} will be elaborated in the next subsection.

4.2.4 Objective Function and Optimization

In this section, we introduce the objective function of UDVBH and its optimization process in details. The core idea behind such self-taught frameworks is that the binary code \mathbf{B} generated by balanced rotation is utilized iteratively to guide the deep hash function learning, where \mathbf{B} is expected to preserve the local structures and balanced properties from the processes of feature embedding and balanced rotation, respectively [24]. For the purpose of sharing those properties within the hash function construction, the learning objective of hash function learning is defined as minimizing

the loss between the output of $\mathcal{F}(\mathbf{Z}; \Theta)$ and the learnt binary code \mathbf{B} , where l_2 -norm distance is used as the measurement. The process can be formulated as following:

$$\mathcal{L}_1 = \min_{\Theta, \mathbf{B}} \|\mathcal{F}(\mathbf{Z}; \Theta) - \mathbf{B}\|_F^2, \text{ s.t. } \mathbf{B} \in \{-1, +1\}^{N \times k}. \quad (4.7)$$

Moreover, the balanced rotation is also considered in this case to address the issue of imbalanced variances during the unsupervised code learning, where its learning objective is formulated as Eq. (4.6):

$$\mathcal{L}_2 = \max_{\mathbf{R}} \|\mathbf{R}^T \mathbf{H}^T\|_{2,1}, \text{ s.t. } \mathbf{R} \mathbf{R}^T = \mathbf{I}_k. \quad (4.8)$$

Without loss of generality, we can integrate Eq. (4.7) and Eq. (4.8) together to express the overall objective function of UDVH following previous hashing frameworks as below:

$$\begin{aligned} \mathcal{L} = \mathcal{L}_1 + \lambda \mathcal{L}_2 = \min_{\Theta, \mathbf{B}, \mathbf{R}} & \|\mathcal{F}(\mathbf{Z}; \Theta) - \mathbf{B}\|_F^2 - \lambda \|\mathbf{R}^T \mathbf{H}^T\|_{2,1}, \\ \text{s.t. } & \mathbf{B} \in \{-1, +1\}^{N \times k}, \mathbf{R} \mathbf{R}^T = \mathbf{I}_k, \mathbf{H} \in \mathbb{R}^{N \times k}, \end{aligned} \quad (4.9)$$

where λ is the balance parameter between two terms to provide the general expression of the objective function, i.e., Eq. (4.9). The value of λ is fixed as 1 during the optimization.

With respect to the optimization, instead of optimizing the objective function in a single step, an alternating approach that differs from previous works is proposed in this section, where those two sub-objective functions, namely \mathcal{L}_1 and \mathcal{L}_2 , are optimized iteratively and the deep hash functions can be built by updating the parameters, including \mathbf{R} , \mathbf{B} and Θ , to facilitate the unique self-taught learning process. This enables the interaction between deep feature learning and hash function learning during the optimization procedure so that the parameters of both deep networks and hash function can be jointly optimized. More importantly, our optimization approach transforms the hash function learning into a regression problem with binary code \mathbf{B} as the regression target, thus avoiding the use of a relaxation strategy to solve the non-convex equation with binary constraints in most published works [24, 43, 51, 55, 109, 195, 246]. Following the above descriptions, our optimization of UDVH can be solved by iteratively optimizing \mathcal{L}_1 and \mathcal{L}_2 . The optimization goal of \mathcal{L}_2 can be achieved in the following alternating steps.

4.2.4.1 R Step

With given video feature $\mathbf{H} \in \mathbb{R}^{N \times k}$, Eq. (4.8) can be viewed as the orthogonality constrained $l_{2,1}$ -norm maximization problem, which can be efficiently solved by the gradient flow method in [212]. Following [212], a feasible set for \mathbf{R} is defined as:

$$\mathcal{M}_k = \{\mathbf{R} \in \mathbb{R}^{k \times k} : \mathbf{R}\mathbf{R}^T = \mathbf{I}_k\}, \quad (4.10)$$

which is also called the *Stiefel* manifold. Then the tangent space for \mathcal{M}_k can be formulated as:

$$\mathcal{E}_R = \{\mathbf{E} \in \mathbb{R}^{k \times k} : \mathbf{E}^T \mathbf{R} + \mathbf{R}^T \mathbf{E} = 0\}. \quad (4.11)$$

Here, the basic idea is to find an optimal direction in the tangent space of the current point \mathbf{R} , then project that direction to the feasible manifold, and replace the current point with the projected one. Finally, a stationary point can be achieved by repeating the above steps iteratively. To optimize the problem (4.12), the convergence lemma is illustrated as: *given a point \mathbf{R}_r in the feasible set, the below updating rule will lead to larger value unless it has arrived at a stationary point, namely:*

$$\mathbf{R}_{r+1} = \mathbf{Q}_r \mathbf{R}_r, \quad (4.12)$$

where \mathbf{Q}_r is the *Cayley* transformation matrix defined as:

$$\mathbf{Q}_r = (\mathbf{I}_k + \frac{\tau}{2} \mathbf{W}_r)^{-1} (\mathbf{I}_k - \frac{\tau}{2} \mathbf{W}_r). \quad (4.13)$$

In the above equation, \mathbf{W} is the skew-symmetric matrix, which can be calculated with the upgradient \mathbf{G} . They are formulated as follows:

$$\mathbf{W}_r = \mathbf{G}_r \mathbf{R}_r^T - \mathbf{R}_r \mathbf{G}_r^T, \quad (4.14)$$

$$\mathbf{G}_r = -\mathbf{H}_r^T \mathbf{H}_r \mathbf{R}_r \mathbf{D}_r, \quad (4.15)$$

$$\mathbf{D}_r = \text{diag}(\frac{1}{N^{0.5} s_{r1}}, \dots, \frac{1}{N^{0.5} s_{ri}}, \dots, \frac{1}{N^{0.5} s_{rk}}). \quad (4.16)$$

Here, the subscript r denotes the r -th iteration. τ is a step size satisfying *Armijo-Wolfe* conditions [135], which only controls the convergence rate in updating \mathbf{R} . s_{ti} represents the standard deviation of $(\mathbf{H}\mathbf{R}_r)_{*i}$ and \mathbf{D} is the diagonal matrix. The above steps are iteratively executed until convergence and the obtained \mathbf{R} is the rotation matrix.

Algorithm 2 Unsupervised Deep Video Hashing

Input: The input matrix \mathbf{Z} ; Randomly initialize deep parameters Θ ; Initialize $\mathbf{R}_0 = \mathbf{I}_k$ and $r = 0$; Code length k ;

Output: $\mathcal{F}(\mathbf{Z}; \Theta)$: deep hash function;

```

1: for  $t = 1$  to  $T$  do
2:   Compute feature matrix  $\mathbf{H}_t$  according to Eq. (4.1);
3:   repeat
4:     Compute  $\mathbf{D}_r$  by Eq. (4.16);
5:     Compute  $\mathbf{G}_r$  by Eq. (4.15);
6:     Compute  $\mathbf{W}_r$  by Eq. (4.14);
7:     Compute  $\mathbf{Q}_r$  by Eq. (4.13);
8:     Compute  $\mathbf{R}_{r+1}$  by Eq. (4.12);
9:      $r = r + 1$ ;
10:  until Convergence;
11:  Update rotation matrix  $\mathbf{R}_t = \mathbf{R}_{r+1}$ ;
12:  Update binary code  $\mathbf{B}_t$  according to Eq. (4.17);
13:  Update deep parameters  $\Theta_t$  according to Eq. (4.18);
14:  Until Convergence;
15:   $t = t + 1$ ;
16: end for
17: return  $\mathcal{F}(\mathbf{Z}; \Theta)$ ;

```

4.2.4.2 B Step

After solving \mathbf{R} , the variance of each dimension in the projected video feature \mathbf{H} is balanced and consequently the binary code \mathbf{B} can be calculated via thresholding:

$$\mathbf{B} = \text{sign}(\mathbf{H} \times \mathbf{R}), \text{ s.t. } \mathbf{H} \in \mathbb{R}^{N \times k}, \mathbf{R} \in \mathbb{R}^{k \times k}. \quad (4.17)$$

The obtained balanced codes are prepared to be utilized as the learning objective of network training afterwards.

4.2.4.3 Θ Step

With fixed \mathbf{Z} , \mathbf{R} and \mathbf{B} , the optimization of \mathcal{L}_1 in Eq. (4.7) is further derived as below with the only argument Θ :

$$\min_{\Theta} \|\mathcal{F}(\mathbf{Z}; \Theta) - \mathbf{B}\|_F^2, \text{ s.t. } \mathbf{B} \in \{-1, +1\}^{N \times k}. \quad (4.18)$$

This minimization problem can be solved by fine-tuning the deep network with Stochastic Gradient Descent (SGD) [8] until it gets converged, where the Euclidean loss is minimized via mini-batch back-propagation and the low bound can be found (See 4.3.3). The network parameter Θ is updated simultaneously with the balanced

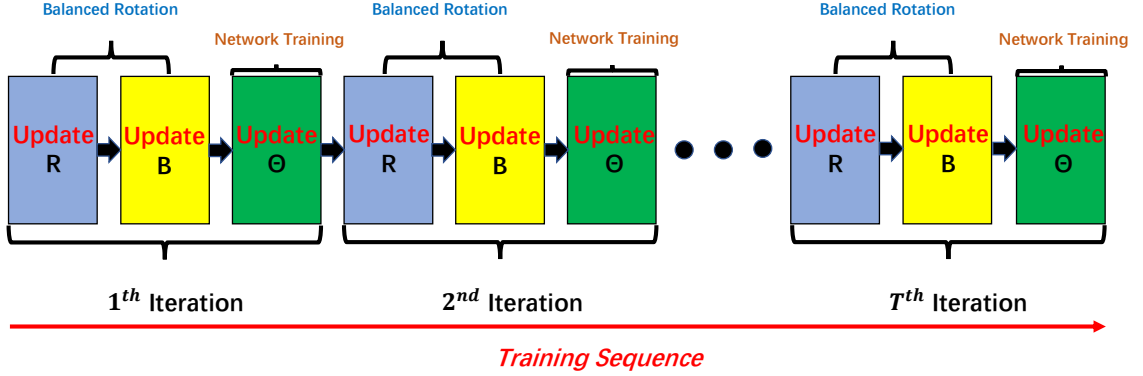


Figure 4.4: The graph illustration of Algorithm 2.

properties from \mathbf{B} preserved after convergence, such boosting the quality of hash code in the query process. By repeating above steps iteratively, the deep hash function can be built finally after several iterations, where $t = 3 \sim 5$ in this work. Given a query video \mathbf{Z}_q , the hash code can be obtained via compute $\text{sign}(\mathcal{F}(\mathbf{Z}_q; \Theta))$. The step by step description of the proposed optimization scheme is summarized in Algorithm 2 with its graph illustration in Fig. 4.4.

4.2.5 Complexity Analysis

The time complexity of Algorithm 2 basically consists of three parts as discussed in Section 4.2: deep network training, feature embedding and balanced rotation. However, calculating the time complexity for first two terms is not straightforward because the costs in network training and feature clustering are affected dramatically by the hardware conditions, which will be discussed later in the Section 4.3. Here, we focus on the time complexity in solving the learning objective of balanced rotation. In particular, the optimization starts by solving \mathbf{R} according to Eq. (4.12)~(4.16), in which the time complexity for Eq. (4.16) is $\mathcal{O}(Nk)$ and Eq. (4.15) is $\mathcal{O}(k^3)$, while computing $\mathbf{H}^T \mathbf{H}$ requires $\mathcal{O}(Nk^2)$, which can be pre-computed and remains unchanged with iterations. For Eq. (4.12)~Eq. (4.14), the time complexity is the same as $\mathcal{O}(k^3)$. Thus, the overall complexity in updating \mathbf{R} is $\mathcal{O}(Nk^2 + t(Nk + k^3))$ after t -th iteration which is linear to the size of training data. Actually, considering that k and t are usually quite small in the algorithm, optimizing Eq. (4.9) can be highly efficient. In addition, the time complexity is $\mathcal{O}(k^3)$ in Eq. (4.17) in solving \mathbf{B} , which indicates the fast speed of the code learning.

4.2.6 Discussion

Now, we discuss the connection between BR and some previous works. The first one is Iterative Quantization (ITQ) [51], which finds a rotation to minimize the quantization error. In fact, ITQ shares a very similar formulation to BR. If closely looking at the objective function of ITQ, we have:

$$\begin{aligned}\|\mathbf{B} - \mathbf{H}\mathbf{R}\|_F^2 &= \|\text{sgn}(\mathbf{H}\mathbf{R}) - \mathbf{H}\mathbf{R}\|_F^2 \\ &= \text{const} - 2\|\mathbf{R}^T \mathbf{H}^T\|_{1,1}.\end{aligned}\tag{4.19}$$

Hence, the optimization problem of ITQ can be rewritten into the following formulation,

$$\max_{\mathbf{R}} \|\mathbf{R}^T \mathbf{H}^T\|_{1,1}, \text{ s.t. } \mathbf{R}\mathbf{R}^T = \mathbf{I}_k.\tag{4.20}$$

Despite the similar formulations, there are several important differences between ITQ and BR. 1) BR focuses on balancing the variance which can explicitly address the imbalance between dimensions. Although ITQ maximizes the total variance of the data in an indirect way via minimizing the quantization errors, the bits are quite imbalanced; 2) ITQ adopts an iterative strategy with two matrix variables for optimization which can only achieve a local optimal while BR has one matrix variable and a more effective optimization algorithm so that it can achieve better solution. The other two related works are Isotropic Hashing (IsoH) [88] and Harmonious Hashing (HamH) [221]. IsoH aims to find a rotation to balance the variance by minimizing the reconstruction error of the covariance matrix and a diagonal matrix. But the reconstruction on a small covariance matrix seems restricted, so it may be unstable in large-scale and high-dimensional experiments [221] due to the overfitting problem. HamH seeks for a rotation to minimize the distance between the rotated data and a perfectly balanced matrix. However, such strict requirement and its non-iterative optimization algorithm may fail to find a good enough solution as BR which limits its performance, though it may balance the data to some extent. In addition, IsoH is based on PCA projection and is derived from spectral hashing [211]. There is no clue that they have stable results based on other projection learning methods. Recently, bit-independent constraint has been incorporated in the nonlinear discrete optimization, where the l_2 -norm distance is minimized between the hash code and the real-valued matrix set [24]. Although the information content is increased inevitably in this case, the quantization errors are accumulated during the iterative optimization

and the aforementioned imbalanced variance within each dimension is still an open issue that affects the hash code quality critically.

4.3 Experiments

In this section, three large-scale video datasets are adopted in the experiments, while the corresponding parameters settings are introduced carefully. Then systematic evaluation and analysis of the proposed frameworks: UDVH-LSTM and UDVH-TSN, are illustrated compared with some state-of-the-art hashing methods.

4.3.1 Datasets and Experimental Setting

We validate the proposed deep hashing learning framework on several large-scale public datasets for video retrieval, including FCVID¹, YFCC² and ActivityNet³. The basic information about those datasets are introduced as below.

4.3.1.1 FCVID

Fudan-Columbia Video Dataset (FCVID) [140] collects 91,223 videos from *Youtube*, which are classified into 239 categories with accurate manual labels. A wide range of generic topics are included in this datasets, such as sports, events and various scenes with the average length of 167 seconds. We randomly select 45,611 videos for the train split in the unsupervised learning process and the rest is used as the test split in the retrieval.

4.3.1.2 YFCC

Yahoo Flickr Creative Common (YFCC) [181] is one of the largest public video dataset available in real-world, which contains over 0.8M video clips with the average length of 37 seconds as claimed. However, 0.7M videos are downloaded and processed in this experiment except for some corrupted videos and invalid download links. Particularly, we split the dataset into two parts: 0.6M unlabeled videos as the train split in the unsupervised learning and 0.1M labeled videos provided by [240] as the test split in the retrieval, where there are 80 popular scenes collected from the third level of MIT SUN scene hierarchy [220] in the second part.

¹<http://http://bigvid.fudan.edu.cn/FCVID/>

²<https://webscope.sandbox.yahoo.com/>

³<http://activity-net.org/>

4.3.1.3 ActivityNet

ActivityNet [14] covers a wide range of complex human activities in the daily living, which contains around 20,000 video clips that classified into 200 categories. Particularly, those videos are split into training, validation and testing sets with 10,024, 4,926 and 5,044 videos individually. Generally we follow the settings of above datasets, where the whole ActivityNet dataset is split into train and test sets averagely, about 10,000 videos for each set. However, for the purpose of making the best of dataset, we use the original testing videos in the unsupervised training for the labels of those are not released and take some instances from training videos to construct the new test set. For all datasets, 1,000 videos randomly picked up from the test split are used as the query instances and others form the gallery in the online retrieval.

4.3.2 Baselines

Several existing hashing methods are adopted as baselines in the experiment, which are Anchor Graph Hashing (AGH) [118], Submodular video hashing (SubMod) [18], Iterative Quantization (ITQ) [51] Spectral Hashing (SP) [211], Multiple Feature Hashing (MFH) [171], Deep Hashing (DH) [43] and Self-Supervised Temporal Hashing (SSTH) [240]. UDVH-LSTM⁴ is also used as a baseline in evaluating UDVH-TSN. Deep Video Hashing (DVH) [109] and Nonlinear Structural Hashing (NSH) [24] are not considered in this case because both of them are supervised methods. For the consistency of comparison, the experiments are carried out with the identical data sets and the best performance is tuned according to parameter settings in their papers.

4.3.3 Implementation Details

Without loss of generality, 20 frames are uniformly selected as the representation of video clips for each instance above datasets. In the feature embedding, 10,000 video features from \mathbf{Z} are randomly picked up and utilized on all datasets. During the code optimization, r and τ are set to 100 and 0.0005. The experiments are conducted using Matlab 2014a on the Ubuntu server configured with Intel Core i7-5960X CPU, 64 GB of RAM, and TITAN 1080i GPUs. In the network training of UDVH-LSTM, the neuron number of LSTM is set to 1,024 with a step size of 20. The initialization methods of LSTM weights and bias are uniform and constant. The output numbers of FC7 and FC8 layers are set to 1024 and k , respectively. Gaussian distribution and

⁴CNN features are replaced with the frame-level features from the pre-trained TSN models in the following experiments.

constant are applied for the initialization of their weights and bias, respectively. The base learning rate is set to 10^{-3} with momentum and weight decay tuned as 0.9 and 0.0005. The batch size of the video sequence is fixed to 50. The maximal training iteration is set to 15,000. Usually, the proposed networks converge within 10 epochs during each loop (t) of the hash function learning.

There are minor differences in the implementation processes for network training for UDVH-TSN when compared to UDVH-LSTM, which are detailed as follows. In UDVH-TSN, we generally follow the parameter settings in the released codes⁵ and its paper of temporal segment network [200] with minor adjustments in the network model, which is illustrated in Sec 4.2. Particularly, in this case, we utilize the original RGB frames and extract their optical flow⁶ [236] images from videos in above benchmark datasets by *OpenCV* as the input streams to the spatial and temporal networks, where those network models are pre-trained on UCF101 [174] dataset according to [200]. The modified version of *Caffe* [200] is also kindly provided by the authors to accelerate the training process. It is worth mentioning that we utilize the original RGB and optical flow frames instead of using extracted CNN feature vectors (VGG-19 [164]) in UDVH-LSTM and evaluate the video accordingly by the uniformly-sampled 20 frames following [200] for fair comparison in the experiments.

The pooled video features of 1024-d from *global-pool* layers of the pre-trained TSN models (including spatial and temporal ConvNets) are averagely fused and utilized in evaluating the performance of some baselines, such as AGH, ITQ, SP, and DH. While for SSTH and UDVH-LSTM, we extract the frame-level features (i.e., 20 frames uniformly sampled from video clips) from the pre-trained TSN models and then fuse them frame-by-frame accordingly to obtain the pooled 20 frame features to facilitate their training processes.

4.3.4 Evaluation Metrics

To evaluate the effectiveness of hashing methods, mean Average Precision at top K (mAP@K) retrieved videos is adopted as the main performance metric following [240], which is defined as the mean of average precision of returned relevant videos number in the top K results [140]. Moreover, Precision-Recall (PR) curve and Precision at top 100 (Precision@100) returned samples are adopted as assisted measurements for systematical evaluation [31]. All of those hashing methods are estimated based on four different bit sizes, i.e, 16, 32, 64, 128.

⁵<https://github.com/yjxiong/temporal-segment-networks>

⁶We adopt TV- L^1 optical flow algorithm as [200].

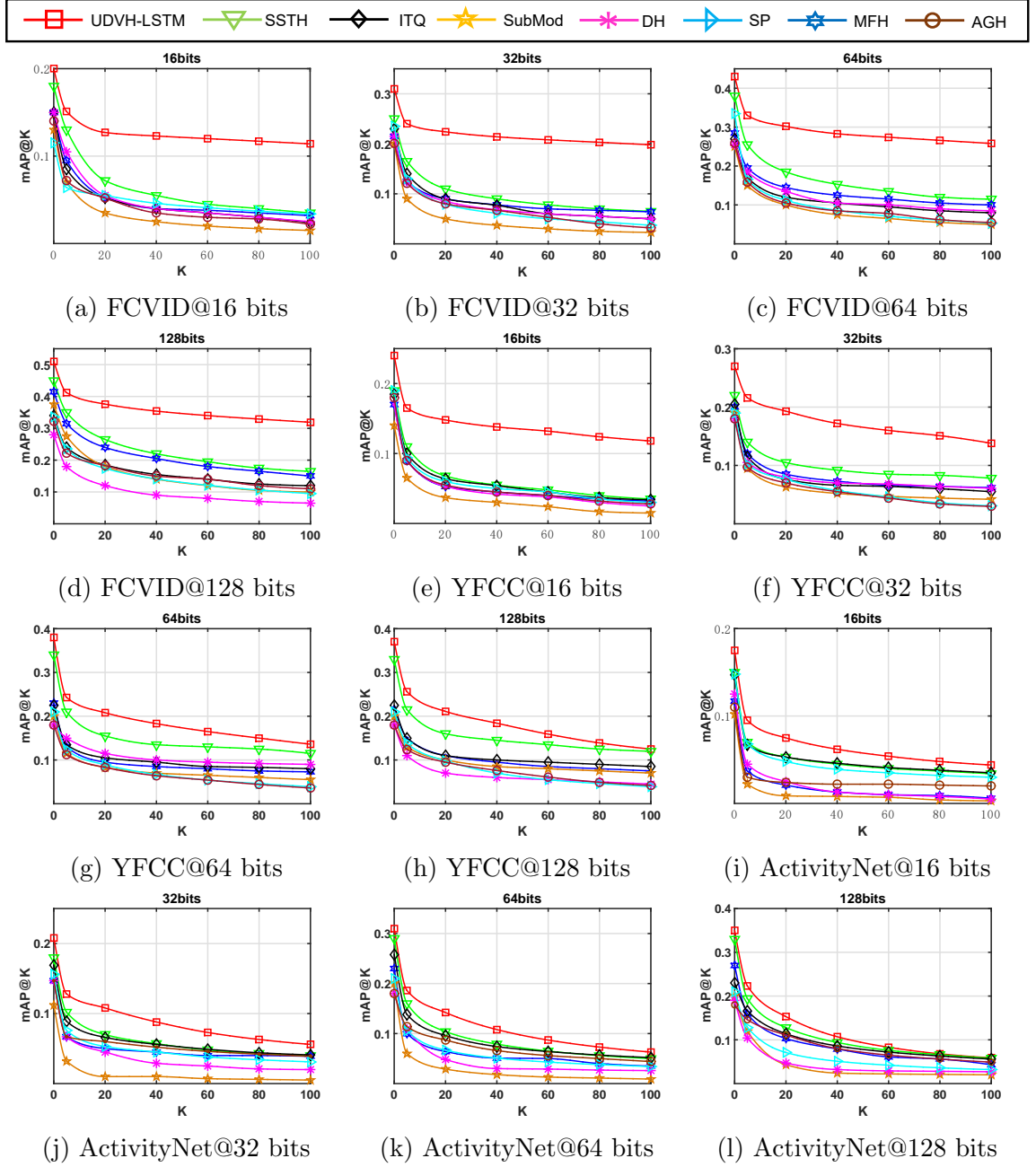


Figure 4.5: The mAP@K curves at different bit sizes under UDVH-LSTM.

4.3.5 Comparison with State-of-The-Arts

To demonstrate the superiority of our methods: UDVH-LSTM and UDVH-TSN, we further compare them with some competitive baselines on three large-scale video datasets independently.

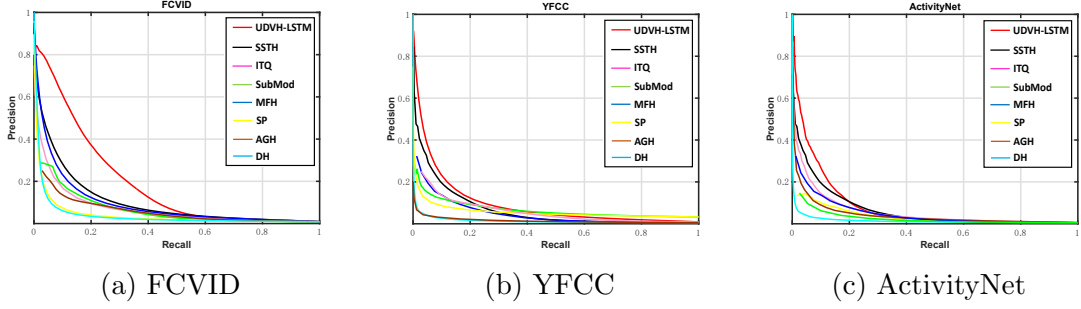


Figure 4.6: The Precision-Recall curves at 128 bits under UDVH-LSTM.

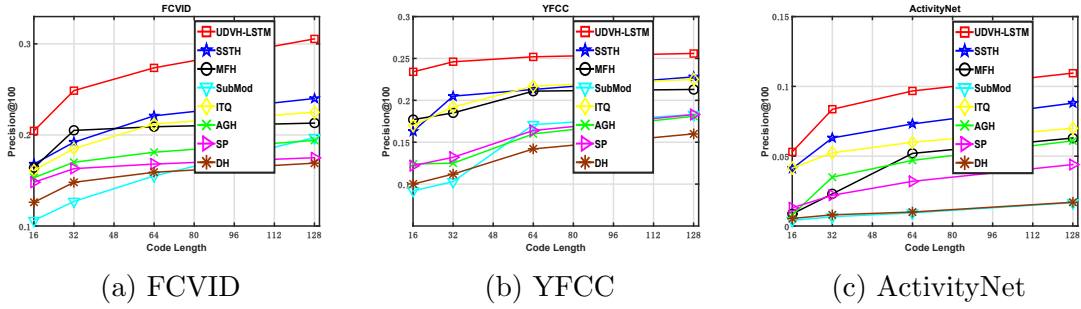


Figure 4.7: The Precision@100 curves at various bit sizes under UDVH-LSTM.

4.3.5.1 Results from UDVH-LSTM

Experiment I on FCVID: We first report the comparison results on FCVID dataset, where Fig. 4.5(a)~(d) show the mAP@K curves at various code lengths on the datasets when using different methods separately. Obviously, the proposed method outperforms the state-of-the-art hashing approaches in all cases substantially. To be specific, the mAP@5 value of our algorithm is 41.2% when using 128-bit code on FCVID, which is about 6% higher than the value achieved by the most comparable SSTH. Moreover, the mAP values decline slowly in contrast to the other methods when increasing the returned video number, which indicates the stability of UDVH-LSTM.

In order to validate the effectiveness of the proposed framework, we also plot the curves of Precision-Recall at 128 bits (Fig. 4.6(a)), Precision at top 100 returned videos (Fig. 4.7(a)) separately. As can be seen, the best performance is still achieved by UDVH-LSTM under those evaluation metrics.

Experiment II on YFCC: Then the experiments are conducted on YFCC dataset and the results are illustrated as follows. Theoretically, the best performance should be given for all hashing methods on YFCC among three datasets because of more video clips are utilized in the stage of unsupervised training. However, surprisingly,

the retrieval performance drops down for all hashing methods compared to those on FCVID dataset, where the difference between the mAP@5 values of UDVH-LSTM and SSTH at 128 bits is trivially reduced to around 4% for instance according to Fig. 4.5(h). The main cause is that most videos in YFCC dataset are taken by mobile equipment from the amateurs instead of the professionals in FCVID [181]. Video retrieval has become a more challenging task because of the low-quality videos when testing on YFCC dataset.

In Fig. 4.6(b) and 4.7(b), we also plot the PR and Precision@100 curves on YFCC dataset as additional evaluations. According to the figures, dominant performance still has been achieved by UDVH compared to other hashing methods when handling such challenging retrieval tasks on YFCC dataset, which implies the robustness of the proposed framework.

Experiment III on ActivityNet: Regarding the experimental results on ActivityNet dataset, the worst performance has been achieved by all hashing methods among three datasets. However, UDVH-LSTM still holds the dominant position with around 3% higher than the most comparable SSTH at the code length of 128 and the gap continues to increase when the code size reduces. A reasonable explanation for the bad performance on ActivityNet is that the dataset contains too many untrimmed videos with enormous intra-class variance, which lowers the hash code quality heavily and makes the retrieval more intractable [14, 23].

Followed by the same experiments on the above datasets, other curves such as PR and Precision@100 are plotted in Fig. 4.6(c) and 4.7(c) separately to verify the superiority of UDVH-LSTM. As can be seen from these figures, our framework still ranks top compared to other baselines, even in dealing with such challenging tasks.

In Fig. 4.8, the detailed performance of top-5 returned videos achieved by two video hashing methods, UDVH-LSTM and SSTH, is illustrated when using 128 bits. The results of FCVID [140] are used as examples here. Given three different query videos, the proposed method correctly finds five similar videos for each of them whereas SSTH makes mistakes in recognizing *swimming amateur* and *dolphin*.

4.3.5.2 Results from UDVH-TSN

Experiment I on FCVID: Consistent with UDVH-LSTM, the comparison results on FCVID dataset between baselines and UDVH-TSN are reported. Fig. 4.9(a)~(d) show the mAP@K curves at various code lengths on the datasets when using different methods separately. As can be seen, the proposed UDVH-TSN outperforms the state-of-the-art baselines at all bit sizes significantly. Notably, the mAP@20 value of









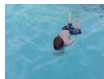

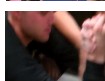


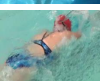
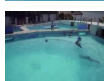

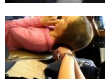



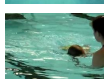

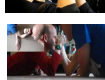



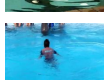
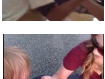
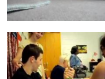
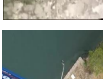

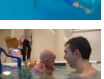

Query	Arm Wrestling		Bungee Jumping		Swimming Amateur	
						
Top-5 Retrieval Result						
						
						
						
						
	UDVH-LSTM	SSTH	UDVH-LSTM	SSTH	UDVH-LSTM	SSTH

Figure 4.8: Top-5 retrieval results when using SSTH and UDVH-LSTM at the code length of 128 bits.

UDVH-TSN is 50.4% at the code length of 128-bit on FCVID, which is 4.8% higher than 45.6% achieved by UDVH-LSTM. That mainly owes to the effective video-level representation learning via the combination of spatial and temporal ConvNets provided by TSN. For the most comparable external competitor SSTH, the gap against UDVH-TSN increases to 8.9%, which indicates the tremendous boost on the system performance achieved by UDVH-TSN. For those non-deep methods, satisfactory results still have been obtained because of the powerful feature modeling of the two-stream architecture adopted in the framework.

Without loss of generality, the PR curves at 128 bits (Fig. 4.10(a)) and Precision@100 (Fig. 4.11(a)) of all baselines are provided separately. As can be seen, the best performance is still achieved by UDVH-TSN under those evaluation metrics, which is consistent with the results from mAP@K curves. For example, the precision@100 value of UDVH-TSN at 64 bits is 51%, which is at least about 3.6% higher than the most comparable baselines.

Experiment II on YFCC: Next, extensive experiments are conducted on YFCC dataset and the results are presented as below. Similar to UDVH-LSTM, significant performance drop appears on YFCC dataset. The mAP@K curves are plot in Fig. 4.9(h). Particularly, the gap between the mAP@20 values of UDVH-TSN (29.8%) and UDVH-LSTM (26.1%) at 128 bits is reduced to around 3.7%, while

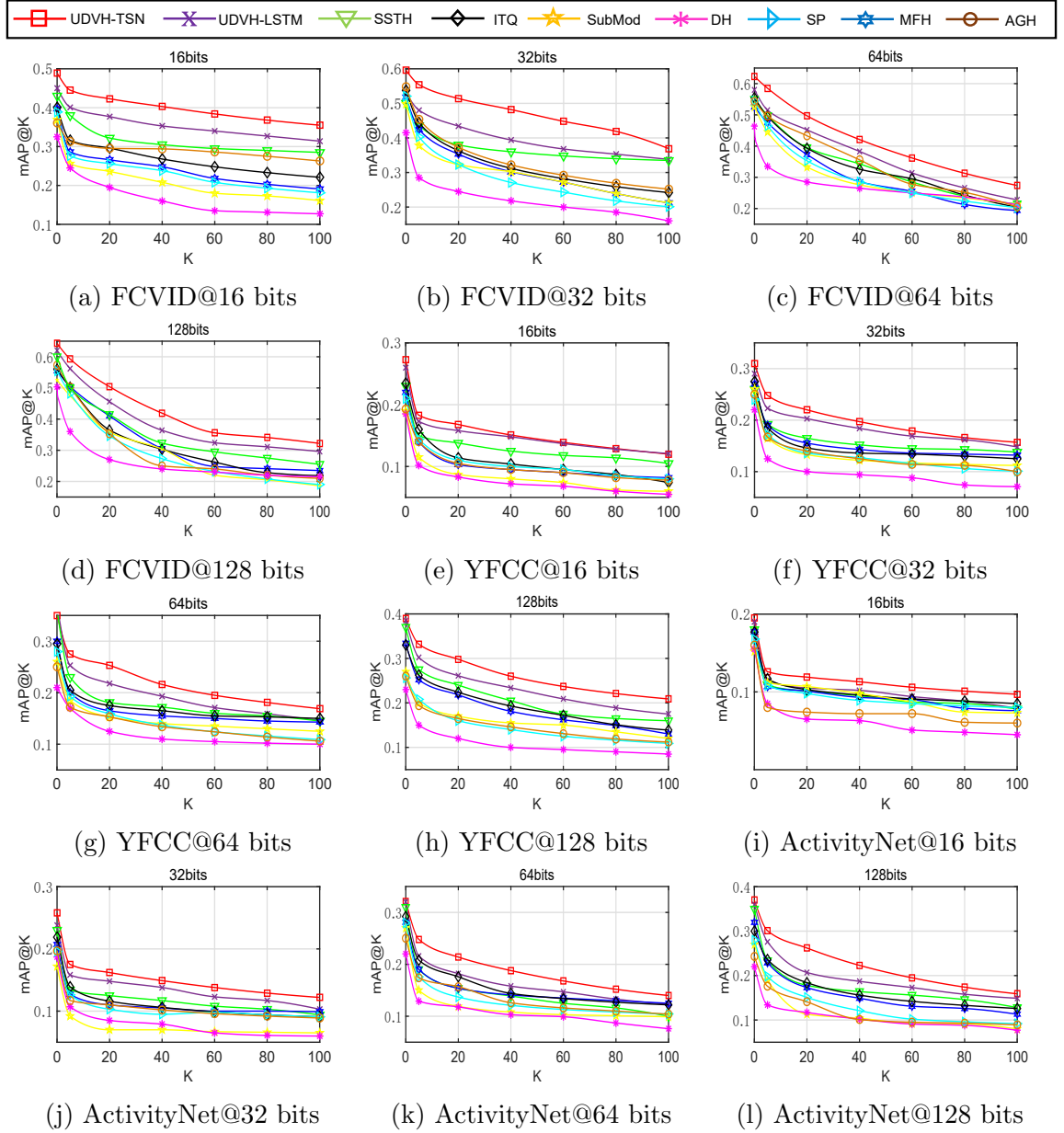


Figure 4.9: The mAP@K curves at different bit sizes under UDVH-TSN.

the value is 5.8% when compared with SSTH (24%). As discussed in the previous section, the main contributing factor is that YFCC dataset contains too many low-quality videos [181]. However, it is worth noting that UDVH-TSN still dominates those baselines with inspiring improvements, even dealing with extremely challenging tasks.

Consistent with Experiment I, the PR (Fig. 4.10(b)) and Precision@100 (Fig. 4.11(b)) curves on YFCC dataset are plotted as additional evaluations. Specifically, the precision@100 value at 64 bits when using UDVH-TSN is 30.2%, which is 3.9%

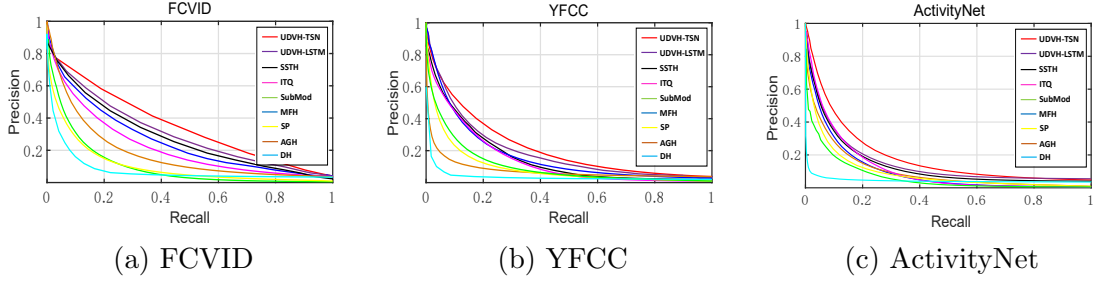


Figure 4.10: The Precision-Recall curves at 128 bits under UDVH-TSN.

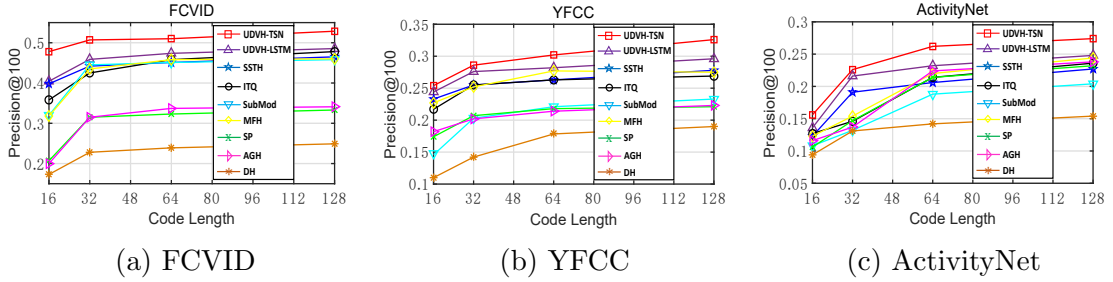


Figure 4.11: The Precision@100 curves at various bit sizes under UDVH-TSN.

higher than that under SSTH (26.3%). According to the figures, dominant performance still has been delivered by the proposed frameworks compared to other hashing methods on such challenging retrieval tasks, which consolidates the superiority of UDVH-TSN.

Experiment III on ActivityNet: Following the results from the above two datasets, the worst performance has been achieved by all hashing methods on ActivityNet dataset. As shown in Fig. 4.9(c), the mAP@20 value of UDVH-TSN is 26.2% on ActivityNet, which is much lower than those on FCVID (50.4%) and YFCC (29.8%). The main reason is that ActivityNet contains too many untrimmed videos with enormous intra-class variance, which lowers the hash code quality heavily and makes the retrieval more intractable [14, 24] compared to the experiments on FCVID and YFCC. When evaluating the performance on the same dataset under various baselines, UDVH-TSN still delivers the best performance according to the mAP@K figures as expected. Specifically, the mAP@20 values for the two representative competitors: UDVH-LSTM and SSTH, are 20.7% and 17.8%, respectively, which are at least 5.5% lower than 26.2% achieved by the proposed framework. Although the improvement is a bit limited, the retrieval performance by UDVH-TSN is still better than the most existing hashing methods. Surprisingly, even compared to some supervised methods like NSH [24] that utilize ActivityNets as the benchmark, UDVH-TSN still gives

promising results. For example, the mAP value at 64 bits of UDVH-TSN is 22.1%, which is 7.3% higher than 14.8% as reported in their paper of NSH.

Without loss of generality, the curves of PR and Precision@100 are plotted in Fig. 4.10(c) and Fig. 4.11(c) separately to further verify the superiority of the proposed UDVH-TSN. As can be seen from those figures, our framework consistently ranks top compared to other baselines. For instance, the precision@100 at 64 bits reaches 26.1% under the proposed framework and the difference over SSTH (20.4%) is 5.7%. Those results show the superiority of UDVH-TSN clearly.



Figure 4.12: Top-5 retrieval results when using SSTH and UDVH-TSN at the code length of 128 bits, where examples are randomly selected from the video datasets. The left column shows the query videos, the middle blocks and right blocks show the top-5 returned videos by SSTH and UDVH-TSN, respectively. Red rectangles indicate mistakes.

In Fig. 4.12, the detailed retrieval results of top-5 returned videos when using 128 bits achieved by two comparable video hashing frameworks, UDVH-TSN and SSTH [240], are revealed. We randomly pick up some examples from the video datasets. Given the query videos of different topics, the proposed algorithm makes fewer mistakes in retrieving similar videos for each query compared with SSTH. The major reason is that the neighborhood structure in the original data is well preserved in UDVH-TSN by incorporating the clustering into the unsupervised hashing process, thus producing more discriminative and effective binary codes for the nearest neighbor search tasks.

4.3.5.3 Discussion

Above all, the proposed UDVH-TSN achieves excellent retrieval performance and outperforms the state-of-the-arts under extensive experiments on the testing datasets.

Dataset	UDVH-LSTM				
	$C=100$	$C=200$	$C=400$	$C=800$	$C=1600$
FCVID [140]	0.285	0.336	0.376	0.371	0.375
YFCC [181]	0.133	0.179	0.221	0.22	0.219
ActivityNet [14]	0.104	0.124	0.144	0.143	0.142

(a) UDVH-LSTM

Dataset	UDVH-TSN				
	$C=100$	$C=200$	$C=400$	$C=800$	$C=1600$
FCVID [140]	0.362	0.482	0.504	0.504	0.502
YFCC [181]	0.201	0.274	0.293	0.289	0.291
ActivityNet [14]	0.145	0.198	0.262	0.257	0.261

(b) UDVH-TSN

Table 4.3: mAP@20 of different cluster numbers at 128 bits on three datasets under UDVH-LSTM and UDVH-TSN.

It is also worth pointing out that DH [43], another deep-based hashing method in those baselines, yields unsatisfactory performance on all datasets, where three speculations are illustrated for those results as follows. First of all, it is essentially an image hashing method, which cannot produce the video hash codes with temporal awareness [240]. Secondly, DH attempts to generate the binary codes via optimizing the approximate codes in the loss function, which lowers the hash code quality because of the huge gaps between the binary and real-valued spaces. The last one is that DH still fails to preserve the similarity structure from the original data in the code learning, which shares similar drawback in SSTH. For UDVH-LSTM using TSN features, it performs slightly worse than UDVH-TSN, which is partly because of the motion information captured by the optical flows are different from LSTM could discover. Moreover, the flow percepts from the temporal ConvNets overfits heavily, which are likely to be overtrained by LSTM, thus leading to less favorable on the test sets [175].

4.3.6 Architecture Investigation

4.3.6.1 Parameter Analysis

We first investigate the influence roughly on retrieval performance when selecting different cluster numbers ($C=100, 200, 400, 800, 1600$) on three datasets under the frameworks of UDVH-LSTM and UDVH-TSN. As shown in Table 4.3, mAP@20 increases dramatically with the bigger cluster number picked up at the beginning and then achieves the best performance when we have about 400 categories. This indicates

the cluster number indeed affects the system performance, where the similar videos will be classified into the same category by the sophisticated clustering strategy, thus producing similar hash codes for those videos. It is a remarkable fact that the performance gets saturated when the cluster number reaches 400, even slight decline after 400, for all datasets under both UDVH-LSTM and UDVH-TSN. The reason might be that those datasets contain less categories than 400, which leads to a quick saturation and makes the parameter easy to be configured as well.

4.3.6.2 Binarization Investigation

In this section, the retrieval performance when using balanced rotation in the binarization is presented carefully, which validates its positive impacts and the claimed contributions. In this experiment, three competitive combinations are adopted: CCA-ITQ, PCA-BR and CCA-BR, where the mAP@K results are illustrated in Table 4.4(a) and 4.4(b) when using those binarization strategies under UDVH-LSTM and UDVH-TSN separately at the code length of 128.

The results show that the combination of CCA and BR outperforms the other two methods on the datasets substantially. Two conclusions can be drawn based on this phenomenon: 1) Compared with the PCA methods, better performance has been achieved by applying CCA in the dimensionality reduction since more valuable information is concentrated and preserved in the top eigenvectors when utilizing the dedicated supervisory information (pseudo labels) from the clustering. It turns out that the choice of CCA is more sensible and effective than PCA; 2) The difference between the results of CCA-ITQ and CCA-BR clearly demonstrates that the retrieval performance can be improved by balancing the variances of video features via the proposed balanced rotation, which indicates the superiority of such framework.

In Fig. 4.13, the variance of each dimension on FCVID at 128 bits by using CCA, CCA-ITQ and CCA-BR under UDVH-TSN is given as an example to validate the above conclusions. Again, it is pretty clear that our BR scheme can always guarantee the flat variance of each dimension, regardless of which video features are used.

4.3.6.3 Loss Function

As shown in Table 4.5, we evaluate the system performance under various loss functions: ℓ_2 -norm, ℓ_1 -norm and cross-entropy, during the network training of UDVH-LSTM and UDVH-TSN. Compared to using ℓ_2 -norm in the objective functions, ℓ_1 -norm is supposed to be more robust in anti-noising and cross-entropy converts the hash function learning into classification problems. According to the results, however,

Method	FCVID				YFCC				ActivityNet			
	K=5	K=20	K=40	K=60	K=80	K=100	K=5	K=20	K=40	K=60	K=80	K=100
PCA-BR	0.358	0.302	0.283	0.274	0.266	0.258	0.194	0.166	0.121	0.081	0.07	0.061
CCA-ITQ	0.387	0.342	0.337	0.325	0.321	0.313	0.219	0.198	0.157	0.159	0.146	0.135
CCA-BR	0.412	0.363	0.354	0.34	0.329	0.319	0.256	0.221	0.184	0.159	0.149	0.13

(a) UDVH-LSTM

Method	FCVID				YFCC				ActivityNet			
	K=5	K=20	K=40	K=60	K=80	K=100	K=5	K=20	K=40	K=60	K=80	K=100
PCA-BR	0.463	0.366	0.291	0.241	0.205	0.179	0.298	0.258	0.235	0.219	0.198	0.183
CCA-ITQ	0.555	0.487	0.412	0.356	0.303	0.297	0.304	0.276	0.24	0.222	0.207	0.196
CCA-BR	0.593	0.504	0.42	0.362	0.347	0.328	0.326	0.293	0.246	0.228	0.215	0.202

(b) UDVH-TSN

Table 4.4: mAP@K of 128 bits when using various banarization schemes separately in the code learning of UDVH-LSTM and UDVH-TSN.

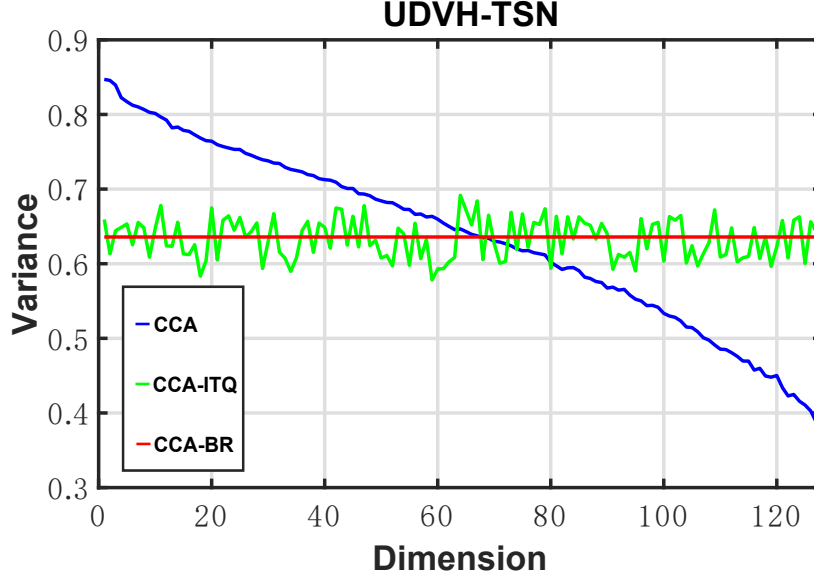


Figure 4.13: The variance distribution from FCVID at 128 bits under UDVH-TSN.

Table 4.5: mAP@20 at 128 bits when applying various loss functions accordingly during the hash function learning of UDVH-LSTM and UDVH-TSN.

Loss Function	UDVH-LSTM			UDVH-TSN		
	FCVID	YFCC	ActivityNet	FCVID	YFCC	ActivityNet
$l_2 - norm$	0.456	0.261	0.207	0.504	0.293	0.262
$l_1 - norm$	0.442	0.256	0.186	0.483	0.279	0.237
$cross - entropy$	0.447	0.248	0.192	0.501	0.291	0.252

the effects of using various loss functions are not obvious. Generally, ℓ_2 -norm achieves the best performance with limited improvement (less than 3%) on the same dataset.

4.3.7 Feature Selection

In this section, we make brief comparisons on the performance variations when using CNN and TSN features on three datasets, where the mAP@5 results at various code lengths for the image-based (e.g., ITQ, AGH, DH) and video-based hashing (SSTH, UDVH-LSTM and UDVH-TSN) methods are plotted in Fig. 4.14 and Fig. 4.15, respectively. Particularly, CNN features are extracted from the pre-trained VGG-19 [164] model and the corresponding results when adopting CNN features are directly reported from [217]. According to those figures, the great improvements have been achieved by using TSN features in both shallow and deep based baselines compared to those on CNN features, which mainly owe to the powerful modelling ability on

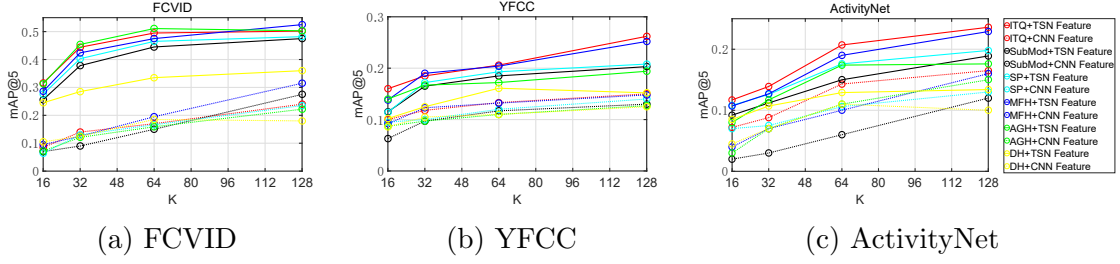


Figure 4.14: The mAP@5 variations when using CNN and TSN features separately in the image-based hashing methods, where the solid and dot lines represent the results on TSN and CNN features, respectively.

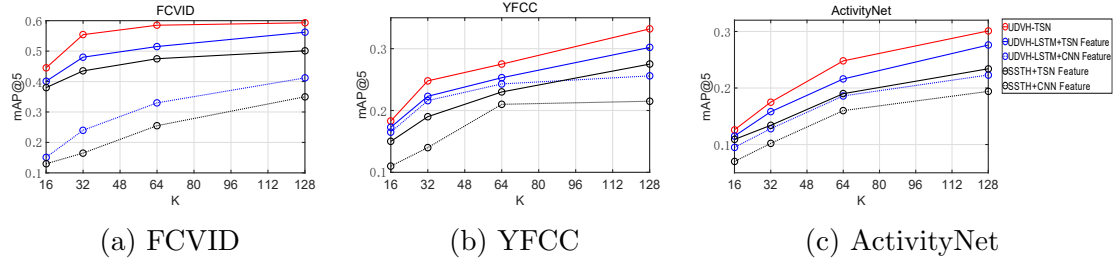


Figure 4.15: The mAP@5 variations when using CNN and TSN features separately in the video-based hashing methods, where the solid and dot lines represent the results on TSN and CNN features, respectively.

temporal information from TSNs. Here, we focused on the video-based methods, including SSTH, UDVH-LSTM and UDVH-TSN, in Fig. 4.15, to estimate the feature impact on video hashing. Specifically, as shown in Fig. 4.15(a), the mAP@5 values at 128 bits on FCVID are 50.1% and 57.1% for SSTH and UDVH-LSTM, which are over 15% higher than the values achieved when using CNN features. For the other two datasets, the gaps are reduced to less than 10% because of the poor data quality as discussed in previous sections.

Moreover, the UDVH-based methods substantially yield better retrieval performance over SSTH, which is consistent with the results reported in Fig. 4.9 and further demonstrates the superiority of our self-taught training strategy involving clustering and balanced rotation. For examples, the mAP@5 values at the code length of 128 are 59.3% and 57.1% for UDVH-TSN and UDVH-LSTM on FCVID, which are at least 7% higher than that in SSTH (50.1%).

4.3.8 Efficiency Analysis

The efficiency issue is addressed in this part because it prevents such deep video hashing frameworks from being widely deployed in the real-world retrieval applications.

Table 4.6: The training time at various code lengths on FCVID when using SSTH, UDVH-LSTM and UDVH-TSN. The time unit for network training is hour (h) and the rest processes are reported in second (s).

Method	16 bits			32 bits			64 bits			128 bits		
	FC	BR	NT	FC	BR	NT	FC	BR	NT	FC	BR	NT
SSTH [240]	/	/	8.21h	/	/	8.23h	/	/	8.48h	/	/	8.55h
UDVH-LSTM	260.5s	0.38s	0.39h	263.2s	0.6s	0.39h	262.4s	1.1s	0.41h	265.1s	2.02s	0.41h
UDVH-TSN	267.3s	0.42s	0.74h	265.2s	0.62s	0.79h	270.2s	1.05s	0.81h	271s	2.1s	0.8h

As mentioned above, SSTH, UDVH-LSTM and UDVH-TSN are specifically designed for video hashing in those baselines, thus making the comparisons among them more persuasive. Consequently, the training expenses on FCVID dataset at various code lengths when using those methods are summarized in Table 4.6. There are three sub processes during the training of UDVH-LSTM and UDVH-TSN: feature clustering (FC), balance rotation (BR) and network training (NT). For SSTH [240], we generally follow the settings in the paper and its released code⁷ while adjusting network parameters accordingly and only the network training is required. The experiments are conducted on the same hardware configuration reported previously.

As can be seen, the training times on FCVID dataset are around 0.5 and 0.9 hours per loop, which are calculated as the sum of the time values in those sub processes, when using UDVH-LSTM and UDVH-TSN individually. Considering the deep architectures usually converge within $t = 5$ loops for the UDVH-based methods, the total training cost is less than 4.5 hours, which is much shorter than training SSTH (over 8 hours), thus indicating the high efficiency of the proposed hashing framework. The most time-consuming parts of SSTH include the backpropagation of Binary LSTM units when updating their binary outputs and the complex network structures [240], which are not adopted in our algorithm. Moreover, it is worth noting that the proposed balanced rotation only costs a few seconds, which is highly competitive in the hashing applications.

4.4 Chapter Summary

In this chapter, we have presented a novel video hashing method termed Unsupervised Deep Video Hashing (UDVH) for fast large-scale similarity search. In contrast to the previous video hashing approaches, feature learning and hash function learning are jointly integrated in a self-taught manner and optimized in an alternating way within the deep architecture of UDVH. With the balance rotation applied in processing

⁷<https://github.com/hanwangzhang/BLSTM>

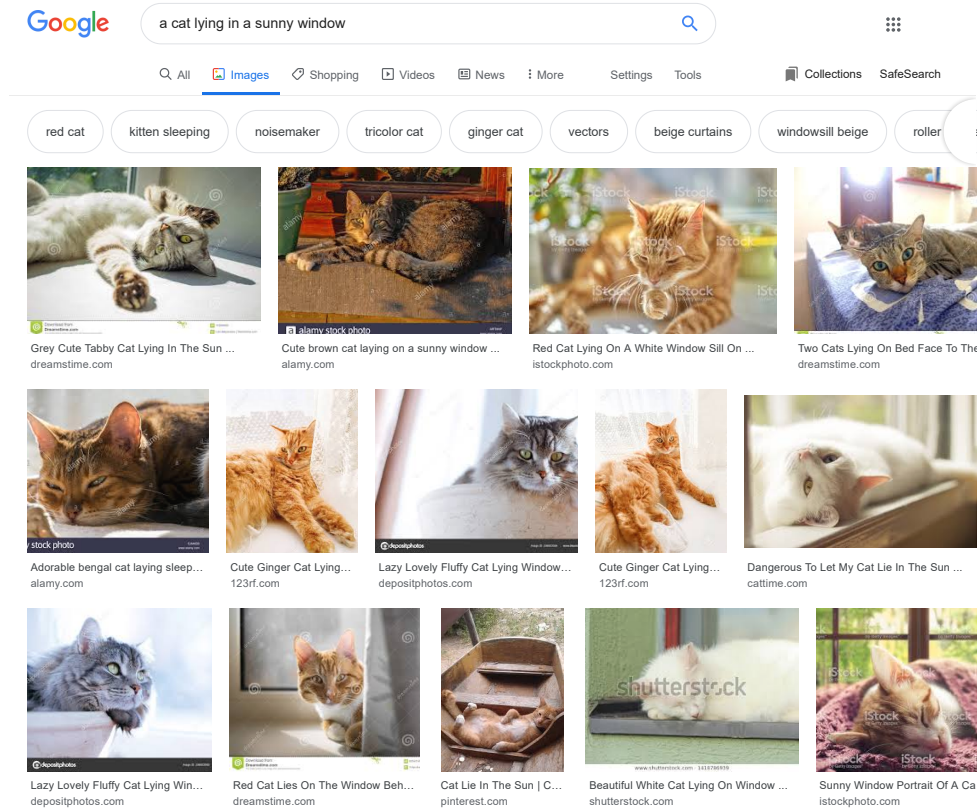


Figure 4.16: An example of cross-modality similarity search: text query image, in Google Images.

the video features, the variance of dimensions when projecting them into a low-dimensional space can be balanced, which improves the code quality. Our approach yields outstanding performance in the extensive experiments on three video datasets, which outperform the state-of-the-arts significantly in terms of retrieval accuracy and efficiency. In the future work, we will apply the proposed feature binarization to deal with other applications such as object detection [63,64], object tracking [33,62], and feature fusion [61,245].

The chapter above focuses on the research problems of video-to-video retrieval, which is a special case of the single-modality nearest neighbor search. However, the data from query and gallery could be varied in different forms in the existing search engines. For example, a short sentence *“a cat lying in a sunny window”* is typed into the search bar of Google Images⁸ and some relevant images are returned, as shown in Fig. 4.16. This is a typical search case of text query image in the cross-modality retrieval. In the next chapter, we will extend the research topic from the

⁸<https://images.google.com/>

single-modality retrieval to the cross-modality retrieval, which aims at proposing novel solutions in the cases that the query and gallery data come from different modalities.

Chapter 5

Deep Cross Modal Hashing

5.1 Introduction

With the explosive growth of multimedia content, such as text, image/video, and audio, cross-media retrieval is becoming increasingly attractive, which allows users to get the results with various media types by submitting one query of any media type. In the context of big data, we need retrieval algorithms that are able to accurately search in the large-scale datasets, and meanwhile, ensure the costs related to the processing overhead and storage requirements do not grow with the quantity of the data being produced. Hashing, which represents the high-dimensional data with the compact binary codes, has drawn a considerable attention in the field of similarity retrieval for its low memory consumption (binary representation) and fast retrieval speed (bit-wise XOR calculation). These properties make the hashing technique a popular solution for many applications [47, 74, 155, 176, 199, 201, 203], where the systems that have been commercialized include Shazam hashing/fingerprinting for music identification [187], Philips video hashing for broadcast monitoring [139] and Civolution SyncNow for cross-media search [60].

Regarding cross-media hashing, one of the main challenges is how to tackle the semantic gaps within different modalities compared to single-modal hashing in Chapter 3. Most existing methods, both in unsupervised [35, 170, 189] and supervised [10, 108, 180, 223] manners, concentrate on learning a common latent space for the multimodal data during the training process such that the heterogeneity among modalities can be minimized [35, 108]. Specifically for unsupervised methods, Cross-View Hashing (CVH) [93], Inter-Media Hashing (IMH) [170] and Linear Cross-Modal Hashing (LCMH) [251] are three of the earliest works in the area of unsupervised cross-modal hashing. The first two methods extend conventional spectral hashing from unimodal to multimodal data and learn the hash function by solving eigenvalue decomposition

with constructed similarity graph. In LCMH, some typical cluster centroids are picked up to represent the original data, thus reducing the computational cost substantially. However, such a process of eigenvalue decomposition in LCMH still compromises the hash code quality because of arbitrary mapping [188].

Consequently, Collective Matrix Factorization Hashing (CMFH) [34] adopts matrix factorization in modelling relations among different modalities, where unified binary codes are being learned via quantizing real-valued unified latent space. Latent Semantic Sparse Hashing (LSSH) [249] updates CMFH by using sparse coding in matrix factorization to learn binary codes for different modalities. However, in both CMFH and LSSH, various relaxation and rounding schemes are utilized in generating hash codes, which usually lead to large quantization errors in the binarization. To deal with large quantization errors in [34], Robust and Flexible Discrete Hashing (RFDH) [189] is proposed to directly optimize and generate the unified binary codes for various views in the unsupervised manner via matrix factorization such that large quantization errors caused by relaxation can be relieved to some extent. However, despite the claimed contributions in RFDH, the neighborhood structures of inter-modal and intra-modal existed in the original data are not well explored.

Although some promising results have been achieved by the previous unsupervised methods, the overall performance is still far below satisfactory from the view of the real-world applications. It is commonly believed that considerable performance gain can be obtained in supervised methods with the aid of dedicated supervision information (e.g, semantic labels, affinity matrix) [199]. Generally, the correlations among different modalities can be enhanced from the label information for unified hash codes in the Hamming space. For example, Semantic Preserving Hashing (SePH) [108] introduces the probability distribution to learn unified hash codes with the semantic affinities. While Discriminant Cross-Modal Hashing (DisCMH) [223] improves the quality of hash codes by means of the label information in shallow linear classifier. However, all the above methods employ a two-step like scheme in learning hash code, which inevitably yields suboptimal results.

Recently, deep learning technology has been widely incorporated in cross-media hashing, where several representative works are discussed briefly in this section [210]. For instance, a stacked auto-encoder architecture is proposed by Correlation Autoencoder Hashing (CAH) [21], where the feature and semantic correlation across modalities are jointly maximized. While another work from Deep Visual-Semantic Hashing (DVSH) [20] employs a metric-based approach to train the visual semantic fusion network with cosine hinge loss. However, the label information is not fully exploited

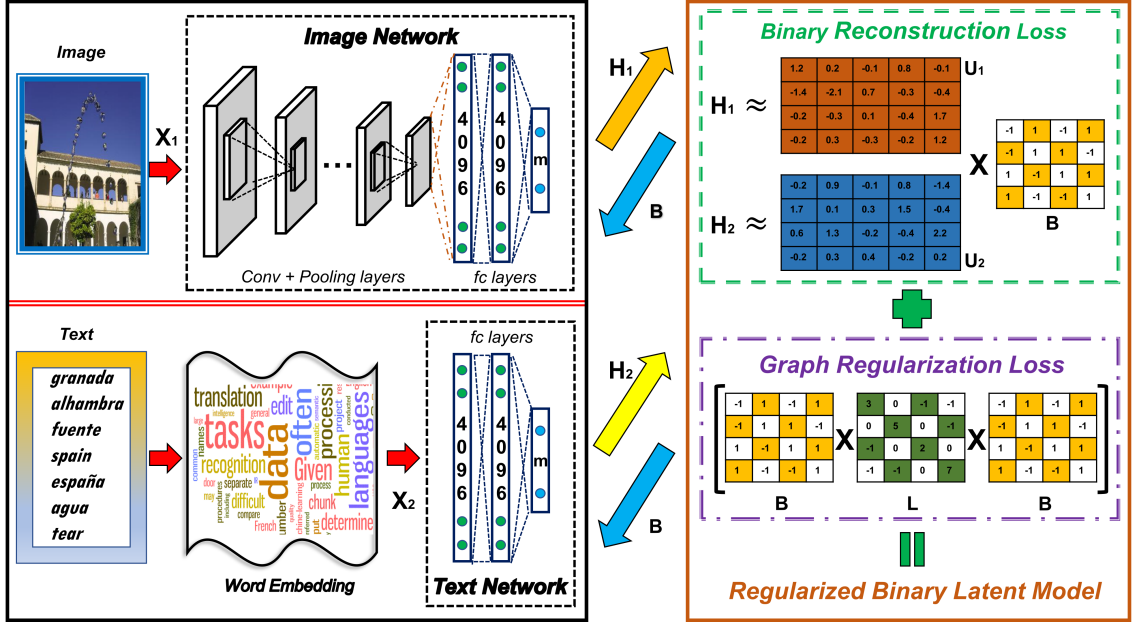


Figure 5.1: The overview of our Self-Supervised Deep Multimodal Hashing. There are three subsections in the training process: deep feature learning (left), deep hash function learning (middle) and regularized binary latent representation learning (right). Specifically, the regularized binary latent model consists of two loss terms: binary reconstruction loss and graph regularization loss. The yellow arrows indicate the deep feature learning. The blue arrows show the iterative directions when learning deep hash functions with the guidance of the unified binary code B . Better viewed in color.

and the performance compromises because of the noisy annotations. Subsequently, Deep Binary Reconstruction (DBRC) [102] suggests another auto-encoder framework for unsupervised cross-modal hashing, which reconstructs the original features from the joint binary representation without considering the similarity relations. Deep Cross-Modal Hashing (DCMH) [84] adopts a negative log likelihood criterion in an end-to-end deep framework, where the similarity structure between real-valued representations is retained. However, such similarity preservation is only performed on approximated hash codes with no restrictions on binary codes directly in the training process, thus reducing the effectiveness of that operation [108, 126, 202, 204, 223].

As discussed above, we summarize three major limitations in the existing cross-modal hashing schemes as follows. Firstly, solving the discrete-constrained objective function usually undergoes a two-step procedure. At the relaxation step, supervised information is exploited to guide continuous hash codes learning, which are converted into binary codes by using rounding technology at the second step. Such a two-

stage solution yields large quantization errors, which will be further magnified after the iterative code learning. Moreover, feature learning and binarization are viewed as two independent steps in most previous methods, thus giving rise to suboptimal results. Last but not least, supervision knowledge cannot be fully explored in the code generation, as well as the hash function learning via simply performing linear mapping between the training data and their labels, which limits the improvement space of the hash code quality for the supervised cross-modal hashing [153, 223]. The situation gets even worse when inaccurate or incomplete labels are provided [108, 126, 202, 204, 223]. Obviously, the retrieval performance would be heavily affected by those drawbacks, thus preventing the existing methods from mass deployment in the real-world applications.

To address the above issues, we propose a novel supervised cross-modal hashing method, termed as **Self-Supervised Deep Multimodal Hashing** (SSDMH), which integrates deep learning and regularized *binary* latent representation model jointly in a unified framework. Specifically, the discrete-constrained objective function is optimized directly without relaxation, and the deep hash functions are built via engaging deep feature learning with code learning in a self-supervised manner. The framework of SSDMH is illustrated in Fig. 5.1 and the corresponding contributions are summarized as follows:

- The matrix factorization based supervised cross-modal hashing method termed as Self-Supervised Deep Multimodal Hashing (SSDMH) is proposed to incorporate the deep feature learning and binarization seamlessly into a unified deep learning framework, where the deep hash functions are being built in a self-supervised manner via projecting the original features from various modalities into a common binary space.
- A novel regularized *binary* latent model is proposed during the code learning, where the discrete unified binary codes can be solved without relaxation and the weights of different modalities are optimized dynamically. Particularly, to make the most advantage of supervision knowledge, we propose to minimize the graph regularization loss, which explicitly preserves the neighborhood structures of the original data and is prone to produce the discriminative hash codes.
- An alternating optimization strategy is adopted in solving the discrete-constrained objective function, where deep parameters and unified binary codes are optimized jointly. Particularly, a novel discrete optimization method, termed as

Table 5.1: The Network Configurations for Two Modalities. Other layers like pooling and activation are omitted for concise descriptions.

Modality	Layer	Description
Image	$conv1 \sim conv5$	Follow the same configuration as AlexNet [90]
	$fc6_I, fc7_I, fc8_I$	$4096-d, 4096-d, m-d$
Text	$fc6_T, fc7_T, fc8_T$	$4096-d, 4096-d, m-d$

Binary Gradient Descent, is proposed to accelerate the optimization speed dramatically, in contrast to the traditional bit-by-bit fashions.

- An unsupervised version of the proposed algorithm, termed Unsupervised Deep Cross-Modal Hashing (UDCMH), is presented to tackle the cross-modal retrieval applications when supervision information is not available. Superior performance has been achieved by the proposed hashing methods from extensive experiments on three datasets.

The reminder of the chapter is organized as follows. We elaborate the proposed SSDMH and UDCMH in Section 5.2. Experimental results along with data analysis are provided in Section 5.3. Finally, the proposed method is concluded in Section 5.4.

5.2 Proposed Method

Fig. 5.1 illustrates the basic structure of the proposed SSDMH. Basically, we extract the deep features from the corresponding deep networks and then utilize those features to generate the unified binary representation via a novel regularized binary latent model. After that, the learned binary code is adopted as supervision information to re-train the previous deep networks, which exhibits the idea of the self-supervised manner. Those processes can be repeated iteratively to obtain the deep hash functions finally. In the next subsections, we will elaborate on the proposed SSDMH.

5.2.1 Problem Definition

Without loss of generality, we use image and text to explain the proposed method. Assuming that the multimodal dataset contains n training instances, which is denoted as $\mathcal{O} = \{\mathbf{X}_i\}_{i=1}^2$, $i = \{1, 2\}$. Each instance has features from the image and text modality, which is represented by $\mathbf{X}_1 = \{x_1^j\}_{j=1}^n$ and $\mathbf{X}_2 = \{x_2^j\}_{j=1}^n$, respectively.

Table 5.2: Mathematical symbols and their short descriptions.

Symbol	Description	Symbol	Description
\mathbf{X}_i	input streams from different modalities	n	number of training samples
\mathbf{S}	similarity matrix	\mathbf{T}	training loop
\mathbf{H}_i	deep features from <i>fc7</i> layer	\mathbf{L}	Laplacian matrix
\mathbf{B}	unified binary code	m	code length
\mathbf{U}_i	latent factor matrix	α_i	weight factors
β, γ, λ	balance parameters	\mathbf{A}_i	affinity matrices
$\mathcal{F}_i(\cdot)$	deep hash functions	\mathbf{D}	diagonal matrix

Consequently, we use $x_1^j \in \mathbb{R}^{d_1}$ to denote the feature vector or the raw pixels of the j -th image and $x_2^j \in \mathbb{R}^{d_2}$ represents the feature vector of the j -th text, where d_1 and d_2 (usually $d_1 \neq d_2$) are the dimensionalities. The affinity matrix $\mathbf{S}_{n \times n} \in [0, 1]$ is also provided as the supervision information, which measures the similarity between data points. In the proposed SSDMH, the aim is to learn the deep hash functions $\mathcal{F}_i(\mathbf{X}_i; \Theta_i)$ that binarize the training data from two modalities into a set of unified binary codes $\mathbf{B} = \{b_i\}_{i=1}^n \in \{-1, +1\}^{m \times n}$, such that the similarities in the original spaces can be preserved. m denotes the code length, \mathbf{X}_i are the input streams to those deep networks for two modalities. Here, the deep network parameters including weights and biases are uniformly defined as Θ_i . The major mathematical symbols used in this chapter are summarized in Table 5.2 for the ease of explanation. Other symbols like \mathbf{G} and \mathbf{K} are applied as the auxiliary parameters in the equation deduction, which are omitted in this table.

5.2.2 Deep Architecture

Considering the favorable feature expressive ability and the deployment flexibility¹, we adopt AlexNet and Multi-Layer Perceptrons (MLP) as the feature modellers for the image and textual modalities, as shown in Fig. 3.2. For the purpose of making the networks compatible with the application, the last fully-connected (*fc*) layers of the original networks are replaced with the new bottleneck layers (*fc_B-I* and *fc_B-T*) comprising m hidden units to facilitate the network training afterwards. *Tanh* function

¹The model sizes for AlexNet and MLP are $< 240\text{MB}$ and $< 90\text{MB}$ after training, which are affordable on most portable devices.

is added at the end of the last layers as the activation function to make the outputs fall into $[-1, 1]$. The network configurations are summarized in Table 5.1. In this paper, the deep architectures not only provide the deep features (e.g. $\mathbf{H}_i \in \mathbb{R}^{4096 \times n}$ from $fc7_I$ and $fc7_T$) in learning the unified binary representation, but also act as the deep hash functions $\mathcal{F}_i(\mathbf{X}_i; \Theta_i)$ to generate hash codes for new queries.

5.2.3 Regularized Binary Latent Model

In the hash code learning, we propose a novel regularized binary latent representation model to generate the unified binary code \mathbf{B} for two modalities. Particularly, the proposed model consists of two loss terms: *binary reconstruction loss* and *graph regularization loss*. We formulate the objective function of the proposed model as below:

$$\min_{\mathbf{B}, \mathbf{U}_i, \alpha_i} \sum_{i=1}^2 \alpha_i^\gamma (\|\mathbf{H}_i - \mathbf{U}_i \mathbf{B}\|_F^2 + \beta \text{Tr}(\mathbf{B} \mathbf{L} \mathbf{B}^T)), \quad (5.1)$$

where β and γ are balance parameters. γ is a positive number controlling the weight of each modality while β estimates the impact of the loss term in (5.1). $\mathbf{H}_i \in \mathbb{R}^{4096 \times n}$ are the deep features extracted from $fc7_I$ and $fc7_T$ layers, $\mathbf{U}_i \in \mathbb{R}^{4096 \times m}$ are the latent factor matrices, $\mathbf{L} \in \mathbb{R}^{n \times n}$ is the Laplacian matrix. $\alpha_i (\alpha_i > 0)$ are the weight factors for two modalities separately and satisfy $\sum_{i=1}^2 \alpha_i = 1$. $\text{Tr}(\cdot)$ is the trace norm. Those terms are elaborated in the next subsections.

5.2.3.1 Binary Reconstruction Loss

As shown in (5.1), the first term measures the reconstruction losses from their latent common binary representation \mathbf{B} to the deep features \mathbf{H}_i , which shares similar idea with CMFH [35]. However, it differs from [35] in two aspects. Firstly, CMFH adopts a two-step approach in generating the unified binary representation, which solves the *real-valued* latent common space $\mathbf{V} \in \mathbb{R}^{m \times n}$ first and binarizes it afterwards, as shown in the top part of (5.2). This inevitably yields the large quantization errors, no matter which rounding schemes are used [108, 189]. However, this problem can be avoided in the proposed model by solving the binary code directly as the bottom of (5.2).

$$\begin{aligned} & \min_{\mathbf{U}_i, \mathbf{V}} \alpha \|\mathbf{H}_1 - \mathbf{U}_1 \mathbf{V}\|_F^2 + (1 - \alpha) \|\mathbf{H}_2 - \mathbf{U}_2 \mathbf{V}\|_F^2 \\ & \Rightarrow \min_{\mathbf{U}_i, \mathbf{B}, \alpha_i} \sum_{i=1}^2 \alpha_i^\gamma \|\mathbf{H}_i - \mathbf{U}_i \mathbf{B}\|_F^2. \end{aligned} \quad (5.2)$$

Moreover, the modality weight is set empirically in CMFH (e.g. $\alpha = 0.5$), while the weights α_i are optimized dynamically in the proposed model. It is more sensible

for the important modality to hold the dominant position in the optimization [125, 189].

5.2.3.2 Graph Regularization Loss

In the second term, we introduce graph regularization to preserve the semantic consistency of data points from multiple modalities, which aims at restricting the neighboring relationships in solving the unified binary code [118]. Particularly, the spectral graph problem can be formulated as:

$$\min \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \|b_i - b_j\|_F^2 \mathbf{S}_{ij} = \text{Tr}(\mathbf{B}\mathbf{L}\mathbf{B}^T), \quad (5.3)$$

where $\mathbf{S} \in \mathbb{R}^{n \times n}$ represents the semantic affinity matrix that can be derived from manual scoring [108], $s_{ij} = 1$ if x_1^i and x_2^j share the same label and otherwise 0. \mathbf{D} is the diagonal matrix whose entries are the column sum of \mathbf{S} , i.e., $\mathbf{D}_{ii} = \sum_{j=1}^n \mathbf{S}_{ij}$. The Laplacian matrix \mathbf{L} can be calculated as $\mathbf{L} = \mathbf{D} - \mathbf{S}$.

5.2.4 Deep Hash Function Learning

Having obtained the unified binary representation \mathbf{B} , the next step is to train the deep hash models ² $\mathcal{F}_i(\mathbf{X}_i; \Theta_i)$ with Euclidean loss layers, which aims at projecting the original features from different views into the common binary space. This strategy integrates the learning processes of deep feature and hash function in a self-supervised manner, thus predicting the discriminative hash codes for new query instances in the testing stage [155]. The problem is formulated as:

$$\min_{\Theta_i} \sum_{i=1}^2 \|\mathcal{F}_i(\mathbf{X}_i; \Theta_i) - \mathbf{B}\|_F^2. \quad (5.4)$$

Particularly, Euclidean distances are minimized between the outputs of the deep networks and the unified binary code \mathbf{B} , while the network parameters Θ_i can be updated through the back-propagation with Stochastic Gradient Descent (SGD). It is worth noting that the original images or features (i.e., \mathbf{X}_i) are fixed and used as the input streams for the deep architecture to facilitate the network training.

²Other predictive models like linear classifier and kernel logistic regression can also be applied here [108, 189], we will leave those in the future research.

5.2.5 Objective Function and Optimization

By combining Eq. (5.1) and Eq. (5.4), the overall objective function of SSDMH is written as below:

$$\min_{\Theta_i, \mathbf{B}, \mathbf{U}_i, \alpha_i} \sum_{i=1}^2 \alpha_i^\gamma (\|\mathbf{H}_i - \mathbf{U}_i \mathbf{B}\|_F^2 + \beta \text{Tr}(\mathbf{B} \mathbf{L} \mathbf{B}^T)) + \lambda \sum_{i=1}^2 \|\mathcal{F}_i(\mathbf{X}_i; \Theta_i) - \mathbf{B}\|_F^2, \quad (5.5)$$

where $\mathbf{B} \in \{-1, +1\}^{m \times n}$. The proposed objective function is an NP-hard problem and cannot be solved directly because of the binary constraints (i.e., \mathbf{B} in this case) [72]. Subsequently, we adopt an alternating strategy to solve Eq. (5.5), where the involved parameters are optimized iteratively by the following steps.

5.2.5.1 \mathbf{U}_i Step

Firstly, by fixing all other variables except for \mathbf{U}_i , Eq. (5.5) is reduced as:

$$\min_{\mathbf{U}_i} \sum_{i=1}^2 \alpha_i^\gamma \|\mathbf{H}_i - \mathbf{U}_i \mathbf{B}\|_F^2, \text{ s.t. } \mathbf{B} \in \{-1, +1\}^{m \times n}. \quad (5.6)$$

Then we calculate the derivation of Eq. (5.6) with respect to \mathbf{U}_i and the closed-form solution of \mathbf{U}_i can be obtained by setting the derivation as 0:

$$\mathbf{U}_i = \mathbf{H}_i \mathbf{B}^T (\mathbf{B} \mathbf{B}^T)^{-1}. \quad (5.7)$$

5.2.5.2 \mathbf{B} Step

Then, with all variables fixed but \mathbf{B} as the only argument, Eq. (5.5) is re-written as:

$$\min_{\mathbf{B}} \sum_{i=1}^2 \alpha_i^\gamma (\|\mathbf{H}_i - \mathbf{U}_i \mathbf{B}\|_F^2 + \beta \text{Tr}(\mathbf{B} \mathbf{L} \mathbf{B}^T)) + \lambda \sum_{i=1}^2 \|\mathcal{F}_i(\mathbf{X}_i; \Theta_i) - \mathbf{B}\|_F^2, \quad (5.8)$$

where $\mathbf{B} \in \{-1, +1\}^{m \times n}$. Then we can expand Eq. (5.8) to:

$$\begin{aligned} & \min_{\mathbf{B}} \sum_{i=1}^2 \alpha_i^\gamma (\text{Tr}(\mathbf{B}^T \mathbf{U}_i^T \mathbf{U}_i \mathbf{B} - 2 \mathbf{B}^T \mathbf{U}_i^T \mathbf{H}_i) + \beta \text{Tr}(\mathbf{B} \mathbf{L} \mathbf{B}^T)) \\ & + \lambda \sum_{i=1}^2 \text{Tr}(\mathbf{B} \mathbf{B}^T - 2 \mathbf{B}^T \mathcal{F}_i(\mathbf{X}_i; \Theta_i)) \\ & = \min_{\mathbf{B}} \|\mathbf{G}^T \mathbf{B}\|_F^2 - 2 \text{Tr}(\mathbf{B}^T \mathbf{Q}) + \text{Tr}(\mathbf{B} \mathbf{K} \mathbf{B}^T), \end{aligned} \quad (5.9)$$

where $\mathbf{G} = [\sqrt{\alpha_1^\gamma} \mathbf{U}_1; \sqrt{\alpha_2^\gamma} \mathbf{U}_2]^T$, $\mathbf{K} = \sum_{i=1}^2 \alpha_i^\gamma \beta \mathbf{L} + 2 \lambda \mathbf{I}_n$, $\mathbf{Q} = \sum_{i=1}^2 (\alpha_i^\gamma \mathbf{U}_i^T \mathbf{H}_i + \lambda \mathcal{F}_i(\mathbf{X}_i; \Theta_i))$. Following discrete cyclic coordinate descent (DCC) [153], we denote b^T

as the j -th row of \mathbf{B} , and \mathbf{B}' the matrix of \mathbf{B} excluding b . Similarly, let g^T be the j -th row of \mathbf{G} , \mathbf{G}' be the matrix of \mathbf{G} excluding g and q^T be the j -th row of \mathbf{Q} , then we have

$$\begin{aligned} \min_b (g^T \mathbf{G}'^T \mathbf{B}' - q^T) b + b^T \mathbf{K} b &= \min_b p^T b + b^T \mathbf{K} b, \\ \text{s.t. } p &= (\mathbf{B}'^T \mathbf{G}' g - q) \in \mathbb{R}^n, \mathbf{B}' \in \{-1, +1\}^{(m-1) \times n}. \end{aligned} \quad (5.10)$$

Obviously, the above equation can be considered as the classical Binary Quadratic Programming (BQP) problem in most previous papers and it can be optimized via solving each entry of b sequentially (flip one entry per time) as described in some coordinate descent based methods [126, 153, 156].

However, those methods usually suffer from the slow convergence issues, especially for the cases with long code. In this work, we propose a new solution called *Binary Gradient Descent* (BGD), which is detailed in the following paragraphs, to accelerate the convergence in optimizing Eq. (5.10).

Suppose that the current value of b is b_0 and the new value b_1 can be obtained by adding an offset Δ to b_0 , namely $b_1 = b_0 + \Delta$. We substitute b_0 and b_1 into Eq. (5.10), the deviation \mathcal{L} between the values of Eq. (5.10) is calculated as follows:

$$\begin{aligned} \mathcal{L} &= b_1^T \mathbf{K} b_1 + p^T b_1 - b_0^T \mathbf{K} b_0 - p^T b_0 \\ &= (b_0 + \Delta)^T \mathbf{K} (b_0 + \Delta) + p^T (b_0 + \Delta) - b_0^T \mathbf{K} b_0 - p^T b_0 \\ &= 2\Delta^T \mathbf{K} b_0 + \Delta^T \mathbf{K} \Delta + p^T \Delta \\ &= \Delta^T \mathbf{K} \Delta + (2\mathbf{K} b_0 + p)^T \Delta. \end{aligned} \quad (5.11)$$

Since there is only one entry with the value³ of -2 or 2 in Δ , then we have $\Delta^T \mathbf{K} \Delta = 4\mathbf{K}_{j,j}$, where j is the index for the entry that is non-zero in Δ . Thus, Eq. (5.11) can be reformed as:

$$\mathcal{L} = 4\text{diag}(\mathbf{K}) + (2\mathbf{K} b_0 + p)^T \Delta, \quad (5.12)$$

where $\text{diag}(\mathbf{K})$ preserves the diagonal elements of \mathbf{K} . Therefore, the deviation \mathcal{L} must be negative if we try to find b_1 to make the objective function descent and it can be obtained by calculating another vector h regarding each entry in b_0 as:

$$h = 4\delta + (2\mathbf{K} b_0 + p) \odot d, \quad (5.13)$$

where δ is the column vector of all diagonal elements of \mathbf{K} , \odot denotes the entry-wise multiplication of the vector, and d satisfies: 1) if the j -th entry of b_0 is 1, then

³The position of -2 or 2 in Δ is based on the corresponding entry in b_0 so as to change -1 to 1 with 2 or 1 to -1 with -2 . All the rest entries are 0 in Δ .

$d_j = -2$; 2) if the j -th entry in b_0 is -1 , then $d_j = 2$. The optimization process will be completed if the smallest value in h is non-negative, otherwise we only retain the value of the corresponding entry in d , and set other entries to 0 to obtain Δ . After getting b_1 with $\Delta + b_0$, we update b_0 above and re-calculate the new Δ accordingly. Essentially, the proposed method flips all the entries by repeating the above computations and selects the entry that is most likely to make the objective function descend in a monotonic discrete manner. As observed from the experiments, it usually requires $n/2$ updates on the entries such that the objective function descends. The proposed BGD only needs 1 iteration to make Eq. (5.10) descent with faster converging speed compared to the later cases that require at least n iterations, thus obtaining the local optimal solution efficiently.

5.2.5.3 α_i Step

With other parameters fixed except for α_i , we formulate Eq. (5.5) as below:

$$\min_{\alpha_i} \sum_{i=1}^2 \alpha_i^\gamma \mathbf{E}_i, \text{ s.t. } \alpha_i > 0, \quad (5.14)$$

where $\mathbf{E}_i = \|\mathbf{H}_i - \mathbf{U}_i \mathbf{B}\|_F^2 + \beta \text{Tr}(\mathbf{B} \mathbf{L} \mathbf{B}^T)$. Subsequently, the Lagrange function of Eq. (5.14) can be formulated as:

$$\sum_{i=1}^2 \alpha_i^\gamma \mathbf{E}_i - \mu \left(\sum_{i=1}^2 \alpha_i - 1 \right), \quad (5.15)$$

where μ is the Lagrange multiplier. Taking $\sum_{i=1}^2 \alpha_i = 1$ into consideration, the optimal solution of α_i is derived as:

$$\alpha_i = \frac{\left(\frac{1}{\mathbf{E}_i}\right)^{\frac{1}{\gamma-1}}}{\sum_{i=1}^2 \left(\frac{1}{\mathbf{E}_i}\right)^{\frac{1}{\gamma-1}}}. \quad (5.16)$$

5.2.5.4 Θ_i Step

When fixing all other parameters but Θ_i , the objective function (5.5) is reduced to

$$\min_{\Theta_i} \sum_{i=1}^2 \|\mathcal{F}_i(\mathbf{X}_i; \Theta_i) - \mathbf{B}\|_F^2, \text{ s.t. } \mathbf{B} \in \{-1, +1\}^{m \times n}, \quad (5.17)$$

where the deep hash functions $\mathcal{F}_i(\mathbf{X}_i; \Theta_i)$ can be solved by optimizing the network parameters Θ_i under the guidance of the unified binary code via mini-batch back-propagation [155]. Repeating the above optimization processes until convergence,

Algorithm 3 Self-Supervised Deep Multimodal Hashing

Input: Input streams \mathbf{X}_i , code length m , parameters β and γ , affinity matrix \mathbf{S} .
 Randomly initialize binary code \mathbf{B} , latent matrices \mathbf{U}_i and deep parameters Θ_i .
 Set weights $\alpha_i = [0.5, 0.5]$, $i = \{1, 2\}$.
Output: Deep hash functions $\mathcal{F}_i(\mathbf{X}_i; \Theta_i)$;

- 1: **for** $T = 1$ to 5 **do**
- 2: Extract the feature matrices \mathbf{H}_i from $fc7_I$ and $fc7_T$ layers of two deep networks, respectively;
- 3: **for** $t = 1$ to 5 **do**
- 4: Update the latent factor matrices \mathbf{U}_i by Eq. (5.7);
- 5: Update the unified hash code \mathbf{B} by Eq. (5.8)~(5.13);
- 6: Update the weight factors α_i by Eq. (5.16);
- 7: **end for**
- 8: Update the network parameters Θ_i by Eq. (5.17);
- 9: **end for**
- 10: **return** $\mathcal{F}_i(\mathbf{X}_i; \Theta_i)$;

the deep hash functions can be learned and deployed for the large-scale multimodal retrieval application. When giving new query instances $\mathbf{X}_i^q \notin \mathcal{O}$, the new hash codes can be obtained by calculating $\text{sign}(\mathcal{F}_i(\mathbf{X}_i^q; \Theta_i))$. The proposed SSDMH is summarized in Algorithm 3.

5.2.6 Computational Complexity

The computational complexity of SSDMH is composed of two parts: learning binary code and deep hash function. However, it is not straightforward to calculate the time complexity for network training, which depends on many external conditions. Regarding the regularized binary latent model, the computational complexity is $O(d^2n + dn)$ during each optimization iteration and $d = \max\{d_1, d_2, m\}$. In total, the training complexity is $O((d^2n + dn)t)$, where t is the maximum iteration (less than 5) when updating the binary code.

5.2.7 Extension to Unsupervised Cross-Modal Hashing

In the real application scenarios, it is difficult to collect the accurate labels for all the data samples in the training set for the similarity retrieval tasks. In this section, we extend our cross-modal retrieval algorithm to a unsupervised deep cross-modal hashing (UDCMH) framework. The difference between SSDMH and UDCMH mainly lies in different ways of the affinity matrix construction. To be specific, the matrix is built

based on manual scoring as Eq. (5.3) in the supervised method. In UDCMH, however, the associated Laplacian matrices $\mathbf{L}_i \in \mathbb{R}^{n \times n}$ are constructed for each modality independently, which are further defined as below:

$$\mathbf{L}_i = \text{diag}(\mathbf{A}_i \mathbf{1}) - \mathbf{A}_i, \quad (5.18)$$

where $\mathbf{A}_i \in \mathbb{R}^{n \times n}$ represent the affinity matrices for two modalities, $\text{diag}(\mathbf{A}_i \mathbf{1})$ are the diagonal matrices with each diagonal element being calculated as the sum of values in the corresponding row of \mathbf{A}_i and $\mathbf{1} = [1, \dots, 1]^T \in \mathbb{R}^n$. The Laplacian constraints in Eq. (5.1) can be unfolded as:

$$\text{Tr}(\mathbf{B} \mathbf{L}_i \mathbf{B}^T) = \frac{1}{2} \sum_{j,k=1}^n (\mathbf{A}_i)_{j,k} \|\mathbf{B}_{*,j} - \mathbf{B}_{*,k}\|_F^2, \quad (5.19)$$

where $\mathbf{B}_{*,j}$ and $\mathbf{B}_{*,k}$ represent the j -th and k -th columns of \mathbf{B} .

Unlike the Laplacian constraints constructed via anchor graph [103, 118] and Laplacian Eigenmaps [170] that only consider the nearest neighbors of data, both the nearest and farthest neighbors are exploited by kNN for each data point from the feature matrix (e.g., \mathbf{H}_1 and \mathbf{H}_2). Then, the corresponding values are set to 1 and -1 separately for those neighbors with balanced weights assigned in building the affinity graph. In such a way, the neighborhood structure can be considered comprehensively and well preserved in Laplacian constraints when optimizing the binary codes. Moreover, by utilizing the affinity matrices with negative weights against traditional non-negative ones, the trivial solutions, which usually yield identical columns in optimizing \mathbf{B} when $(\mathbf{A}_i)_{j,k} > 0$, can be avoided and thus more compact objective function can be formulated without the orthogonal constraints.

By substitute L_i into the objective function (5.5), the optimization process can be solved by following Algorithm 3 with minor changes [216].

5.3 Experiment

In this section, extensive experiments are conducted on three datasets to evaluate the performance of the proposed SSDMH. The comparison results between several unsupervised methods and UDCMH are also reported at the end of this section.

5.3.1 Dataset Descriptions

5.3.1.1 Wiki

Wiki⁴ [146] dataset is made up of 2,866 image-text pairs collected from *Wikipedia*. Each image is represented by a 128-dimensional SIFT feature vector and a 10-dimensional topic vector is given to describe the text. These pairs contain 10 semantic categories and each pair is manually assigned to one of them. All data pairs are split into the training and query sets with the sizes of 2,173 and 693.

5.3.1.2 MIRFlickr

MIRFlickr⁵ [77] dataset collects 25,000 instances from *Flickr*, which are annotated by at least one of 24 provided labels. A 100-dimensional SIFT descriptor is provided to represent each image, while the text is expressed as a 500-dimensional tagging vector. We randomly select 2,000 image-text pairs as the queries and use the remaining pairs for training.

5.3.1.3 NUS-WIDE

NUS-WIDE⁶ [25] dataset contains 269,648 images and each image is associated with a textual tag. Those instances are manually labeled with 81 different concepts. Following [35, 108], we only retain the instances annotated with the 10 most frequent concepts, thus preserving 186,577 image-tag pairs for the experiment. Each image is represented by a 500-dimensional SIFT feature vector and an index vector of the most frequent 1,000 tags is provided for each text. Finally, 2,000 image-tag pairs are randomly picked up as the queries and the rest pairs are used for training. Each pair is labeled with at least one of the 10 concepts and two image-tag pairs are considered to be similar if one of labels matched.

Following previous works, we extract deep features from pre-trained networks for both image and text modalities to improve the retrieval performance in the experiment. Implementation details will be revealed in the next section.

5.3.2 Experiment Settings

We compare the proposed SSDMH with some extremely competitive works published previously, including IMH [170], RFDH [189], DBRC [102], DCMH [84], CAH [21],

⁴<http://www.svcl.ucsd.edu/projects/crossmodal/>

⁵<http://press.liacs.nl/mirflickr/mirdownload.html>

⁶<https://lms.comp.nus.edu.sg/research/NUS-WIDE.htm>

SCMFH⁷ [35], DisCMH [223] and SePH_{km}⁸ [108] in the experiments. For the fair comparison, the identical training and query sets are utilized in the performance evaluation and the best results are reported by adopting and tuning the suggested parameters in their papers. Regarding the evaluation metrics, we generally adopt two widely-used criteria in the multimodal retrieval: mean Average Precision (mAP) and Precision-Recall (PR) curve, as the main metrics in the following experiments [35,108]. The number of top returned instances is set to 50 when calculating MAP. In this paper, we focus on two cross-media retrieval tasks: ‘Image Query versus Text database’ (Image→Text) and ‘Text Query versus Image database’ (Text→Image).

Following the settings in [35,108,170], the whole training set for the Wiki dataset is utilized in generating the unified binary representation. While for the other two benchmarks, 5,000 instances are randomly sampled from their training sets to produce the binary code. For fair comparison, instead of using the original image features (e.g. SIFT) in previous papers, the 4096-d deep feature vectors are extracted from the *fc7* layer of the pre-trained AlexNet for those non-deep methods (e.g. SCMFH [108]) during the code learning. For the parameter settings in the semantic binary latent model, γ , β and λ are set to 5, 1 and 1, respectively. The maximum iteration t is set to 5 in updating the binary code. When building the deep hash functions, the original image pixels and their tagging vectors are kept fixed and employed as the inputs to those deep networks for two modalities, respectively. We adopt SGD optimizer in the network training with the basic learning rates 0.0001 and 0.01 for the image and text modality, respectively. The batch sizes are fixed as 512 and they take 10 epochs at most until the networks converge. In this work, we construct the deep architectures using *Caffe* [82]. The codes for the above prior arts are implemented by MATLAB 2014a on an Ubuntu 14.04 LTS machine, which is configured with Intel Core i7-6700k CPU, 64GB RAM and NVIDIA 1080i GPU.

For UDCMH, we generally follow the implementation steps as SSDMH. One minor difference is that the number of near and far neighbors are set to 50 and 200 in building the affinity graph. For fair comparison, seven unsupervised multimodal hashing methods, CVH [93], IMH [170], LCMH [251], CMFH [35], LSSH [249], RFDH [189] and DBRC [102] are selected based on their original papers against UDCMH in the experiments. Following the previous works, three popular metrics: mean Average Precision (mAP), Precision-Recall (PR) and Precision at Top N returned candidates (Precision@N) curves, are adopted in the performance evaluation accordingly.

⁷We adopt the supervised version of CMFH in the comparison.

⁸SePH_{km} adopts k-means sampling in selecting training data.

5.3.3 Results and Analysis

5.3.3.1 Architecture Investigation

In this section, we first investigate the impact of each loss term in the regularized binary latent model on the multimodal retrieval performance. Particularly, two different cases are analyzed: SSDMH_{brl} and $\text{SSDMH}_{brl+grl}$, where *brl* and *grl* are shorts for *binary reconstruction loss* and *graph regularization loss* respectively. Here, SSDMH_{brl} is realized by setting β as 0 during the optimization. We report the mAP results on three datasets at 128 bits in Table 5.3. As can be seen, the worst performance has been achieved by SSDMH_{brl} without any supervision information. With the graph regularization loss involved, $\text{SSDMH}_{brl+grl}$ improves the mAP values by approximately $3.4\% \sim 8.5\%$ on two retrieval tasks, implying the importance of preserving the neighborhood structure within the original data in the hash code learning.

Then the value variations of $\alpha_i (i = 1, 2)$, which controls the contributions from two modalities in the discrete optimization, at 128 bits during each learning iteration (t) on three datasets are provided, as shown in Table 5.4. As can be seen, the values of α_2 are larger than α_1 at the beginning of the optimization on all datasets. With the ongoing optimization steps, the gaps between the values of α_1 and α_2 reduce gradually when the binary reconstruction errors decrease and finally remain stable after $t = 3 \sim 5$.

5.3.3.2 Overall Comparisons with Baselines

To validate the superiority of the proposed SSDMH, we compare it with the state-of-the-arts and report the mAP results at various code lengths on three datasets, as shown in Table 5.5. Generally, the proposed SSDMH outperforms all baselines in

Table 5.3: mAP results at the code length of 128 when involving various loss terms deployed in the proposed method: SSDMH_{brl} and $\text{SSDMH}_{brl+grl}$.

Method	Task	Dataset		
		Wiki [146]	MIRFlickr [77]	NUS-WIDE [25]
SSDMH_{brl}	Image→Text	0.408	0.745	0.774
	Text→Image	0.703	0.767	0.802
$\text{SSDMH}_{brl+grl}$	Image→Text	0.451	0.823	0.834
	Text→Image	0.745	0.852	0.836

Table 5.4: The variations on $\alpha_i (i = 1, 2)$ during the optimization iteration at 128 bits on three datasets. α_i are initialized as 0.5.

Dataset	α_i	$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$
Wiki [146]	α_1	0.3857	0.4123	0.4123	0.4117	0.4116
	α_2	0.6143	0.5877	0.5877	0.5883	0.5884
MIRFlickr [77]	α_1	0.4419	0.4578	0.4585	0.4583	0.4584
	α_2	0.5581	0.5422	0.5415	0.5417	0.5416
NUS-WIDE [25]	α_1	0.4436	0.4660	0.4664	0.4664	0.4665
	α_2	0.5564	0.5340	0.5336	0.5336	0.5335

terms of mAP on two retrieval tasks. Specifically, regarding Image→Text tasks, the MAP values from SSDMH are 3.9%, 4.7% and 6.7% higher than those achieved by the most competitive baselines at the code length of 128 on Wiki, MIRFlickr and NUS-WIDE, respectively. While for Text→Image task, the gaps have increased to 4.3%, 9.3% and 8.7%. When dealing with the short codes (e.g. 16, 32 bits), SSDMH still achieves the best performance showing the great potential of SSDMH on wide deployment in the industrial applications. Moreover, we also plot the PR curves at 128 bits when using those methods on three datasets, as shown in Fig. 5.2. It can be seen the curves of the proposed SSDMH are always at the top of the figures, which comply with the results in Table 5.5.

5.3.3.3 Top-5 Retrieved Examples for SSDMH

In Fig. 5.3, the top-5 retrieved results of SSDMH regarding two different tasks: Image→Text and Text→Image, on Wiki dataset [146] are presented. Two categories are picked up: *sport* and *warfare*. As can be seen, the texts are quite noisy owing to the disordered and redundant descriptions, which interprets the worst performance on Wiki compared to that of the other two datasets.

Following the previous section, the top-5 returned candidates of SSDMH regarding two retrieval tasks: Image→Text and Text→Image, on MIRFlickr dataset [77] are plotted in Fig. 5.4. Different from the text descriptions in Wiki dataset, MIRFlickr provides a list of keywords for each image and it is a multi-labelled dataset, where the query and its retrieved candidates are considered to be similar if at least one of their labels matched. For example, in the Image→Text task, the query is labelled

Table 5.5: mAP results for Image \rightarrow Text and Text \rightarrow Image tasks on three datasets at various code lengths (bits) when using different methods. The best performance is shown in boldface.

Task	Method	Wiki [146]				MIRFlickr [77]				NUS-WIDE [25]			
		16	32	64	128	16	32	64	128	16	32	64	128
Image \rightarrow Text	IMH [170]	0.201	0.203	0.204	0.195	0.612	0.601	0.592	0.579	0.47	0.473	0.476	0.459
	DBRC [102]	0.253	0.265	0.269	0.288	0.617	0.619	0.62	0.621	0.424	0.459	0.447	0.447
	RDFH [189]	0.242	0.246	0.244	0.243	0.632	0.636	0.641	0.652	0.488	0.492	0.494	0.508
	DCMH [84]	0.264	0.269	0.279	0.284	0.732	0.747	0.748	0.752	0.584	0.603	0.612	0.623
	CAH [21]	0.242	0.248	0.253	0.261	0.688	0.705	0.708	0.715	0.509	0.542	0.567	0.582
	SCMFH [35]	0.284	0.294	0.299	0.305	0.651	0.654	0.655	0.664	0.495	0.499	0.506	0.624
	DisCMH [223]	0.375	0.394	0.395	0.392	0.72	0.727	0.721	0.732	0.683	0.758	0.775	0.764
	SePH _{km} [108]	0.399	0.405	0.408	0.412	0.763	0.769	0.773	0.776	0.739	0.75	0.761	0.767
	SSDMH	0.421	0.436	0.446	0.451	0.797	0.801	0.808	0.823	0.803	0.809	0.821	0.834
Text \rightarrow Image	IMH [170]	0.467	0.478	0.453	0.456	0.603	0.595	0.589	0.58	0.478	0.483	0.472	0.462
	DBRC [102]	0.574	0.588	0.598	0.599	0.618	0.626	0.626	0.628	0.455	0.459	0.468	0.473
	RDFH [189]	0.59	0.596	0.603	0.61	0.681	0.693	0.698	0.702	0.612	0.641	0.658	0.68
	DCMH [84]	0.621	0.628	0.648	0.658	0.733	0.745	0.749	0.753	0.639	0.656	0.661	0.678
	CAH [21]	0.373	0.386	0.393	0.402	0.661	0.674	0.694	0.722	0.514	0.545	0.584	0.608
	SCMFH [35]	0.635	0.641	0.656	0.664	0.682	0.703	0.716	0.726	0.569	0.612	0.657	0.684
	DisCMH [223]	0.676	0.662	0.663	0.654	0.747	0.758	0.75	0.759	0.652	0.736	0.75	0.749
	SePH _{km} [108]	0.664	0.696	0.695	0.702	0.727	0.731	0.748	0.743	0.686	0.695	0.709	0.711
	SSDMH	0.716	0.735	0.737	0.745	0.833	0.836	0.843	0.852	0.815	0.821	0.833	0.836

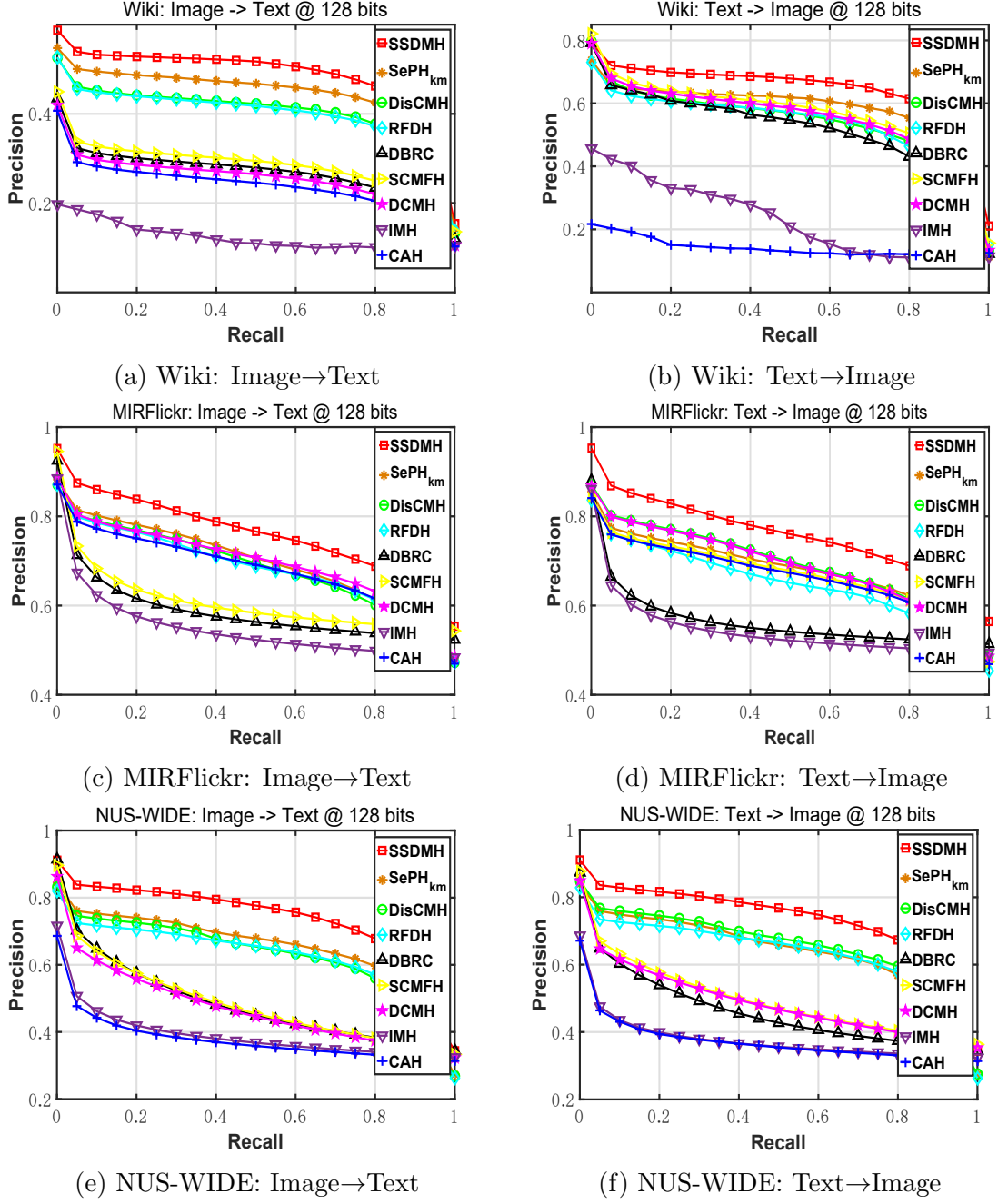



Figure 5.2: The Precision-Recall curves at 128 bits on three datasets.

with *sky* and *structure*, which indicates that the top-5 retrieved texts share at least one of these labels.

In this section, the top-5 returned candidates of SSDMH regarding two retrieval tasks: Image \rightarrow Text and Text \rightarrow Image, on NUS-WIDE dataset [25] are presented in Fig. 5.5. Similar to MIRFlickr, the text descriptions are formed as a list of keywords for each image and each data instance is labelled with at least one of the 10 most

Query



Top-5 Retrieved Texts

Boston College won the ceremonial pre-game coin toss to determine first possession and elected to kick off to begin the game, ensuring that the Eagles would receive the ball to begin the second half. Virginia

.....

On the first play after the penalty, Davis was sacked for a loss of two yards. Harris regained the lost yardage and more with a six-yard run, but after Davis threw an incomplete pass on third down, the Eagles were forced to punt again. Before they could kick the ball, however, the quarter came to an end. With three quarters remaining in the game, Virginia Tech held a 7-0 lead.ESPN.com, Boston College Eagles vs. Virginia Tech Hokies Play-by-Play, ESPN.com, December 6, 2008. Accessed December 6, 2008.

The Virginia Tech offense was led by quarterback Marcus Vick, brother of former Tech all-star Michael Vick. Coming off a season-long suspension in 2004, Vick threw for 1,855 yards, 14 touchdowns and nine interceptions in the 2005 season leading up to the ACC Championship

.....

(PDF) Virginia Tech Sports Information, Hokiesports.com. "All Eyes on Vick", Page 13. E01, November 27, 2005. Accessed December 24, 2007. Imoh, meanwhile, was limited by an ankle injury suffered during the course of the season. Heading into the conference championship game, he had rushed for 415 yards and three touchdowns. (PDF) Virginia Tech Sports Information, Hokiesports.com. "Finding Imoh", Page 12. Accessed March 5, 2008.

The 2007 ACC Championship Game kicked off at 13:10 EST in Jacksonville, Florida. ESPN.com, December 1, 2007. Accessed December 10, 2007. At kickoff, the weather was partly cloudy, with winds from the northeast at . The air temperature was The pre-game coin toss involved two members of

.....

rehabilitation of wounded American combat veterans returning to the United States from fighting overseas. One soldier from Virginia and another from Massachusetts were chosen to throw the ceremonial coin that would determine the game's starting possession. The Atlantic Coast Conference, December 1, 2007. Accessed December 10, 2007. Supervising the coin toss was referee Jack Childress, who had also officiated the inaugural ACC Championship Game.

Virginia Tech began the fourth quarter in possession of the ball and with a first down, but trailing by 24 points and virtually out of the game. The first two plays of the fourth quarter were similar to what the Tech offense had shown all game: an incomplete pass and a Having recovered the ball, and

.....

stop the clock, Florida State was able to run out the remaining time in the game and secure a 27–22 victory. Towards the end of the game, players on each team acted with hostility towards each other, and several received personal foul penalties. The penalties had no effect on the final outcome of the game, and Florida State won the ACC Championship Game and an automatic bid to the 2006 Orange Bowl. ESPN.com, December 3, 2005. Accessed December 23, 2007.

USC continued Pac-10 play by hosting the struggling Stanford Cardinal, under first-year coach Jim Harbaugh. In a major upset, USC stumbled at home to the 41 point underdog, losing 24-23. Harbaugh made headlines prior to the season by claiming 2007 would be Carroll's last year

.....

"Los Angeles Times", October 8, 2007. Accessed July 3, 2008. The upset landed the Trojans in ESPN.com's Bottom 10. David Duffey, ESPN.com, October 9, 2007. Accessed August 4, 2008. In an interview the following month, Carroll assessed the mistakes that led to the loss as his own: At the end of the regular season, "Sports Illustrated" chose Stanford's upset of USC as the second "Biggest Upset of 2007" after Division I FCS Appalachian State's upset of No. 5 Michigan.

(a) Wiki: Image→Text

Query

Ultimately, a mixed solution named "Tecnología Santa Bárbara-Bazán" (Santa Bárbara-Bazán Technology) (or TSB) was chosen. Mazarrasa, "Carro de Combate AMX-30E", p. 80 The improvement of the tank's mobility entailed replacing the HS-110 diesel engine with an MTU 833 Ka-501 diesel engine, producing 850 metric horsepower (625.17 kW), and the transmission with a German ZF LSG-3000, compatible with engines of up to 1,500 metric horsepower (1103.25 kW). The first 30 engines were to have 50% of the engine manufactured in Spain; the rest, 73% were to be produced indigenously. Pérez-Guerra, "Spanish AMX-30 MBT upgrade program", p. 500 This new engine gave the modernized tank a power ratio of 23 metric horsepower to tonne (21.13 hp/S/T). The new engine was coupled with the AMX-30B2's improved torsion-bar suspension, which used larger diameter torsion-bars and new shocks. Mazarrasa,

.....

wind velocity, gun elevation and vehicle inclination. The fire control system also allowed for the future upgrade to a more sophisticated stabilization system for the tank's main gun. Survivability improvements included the addition of new steel side-skirts, a new smoke generating system linked to the engine and a new fire suppression system. Mazarrasa, "Carro de Combate AMX-30", pp. 81-83 One hundred fifty AMX-30Es received this modernization package and were designated AMX-30EM2s. The program began in 1989 and ended in 1993. Mazarrasa, "Carro de Combate AMX-30E", p. 85 Ultimately, Spain's AMX-30EM2s were replaced by brand-new Centauro anti-tank vehicles, which were partially manufactured in Spain, in the early 21st century. "Defensa firma un contrato de 200 millones de euros con Finmeccanica", El País

Top-5 Retrieved Images

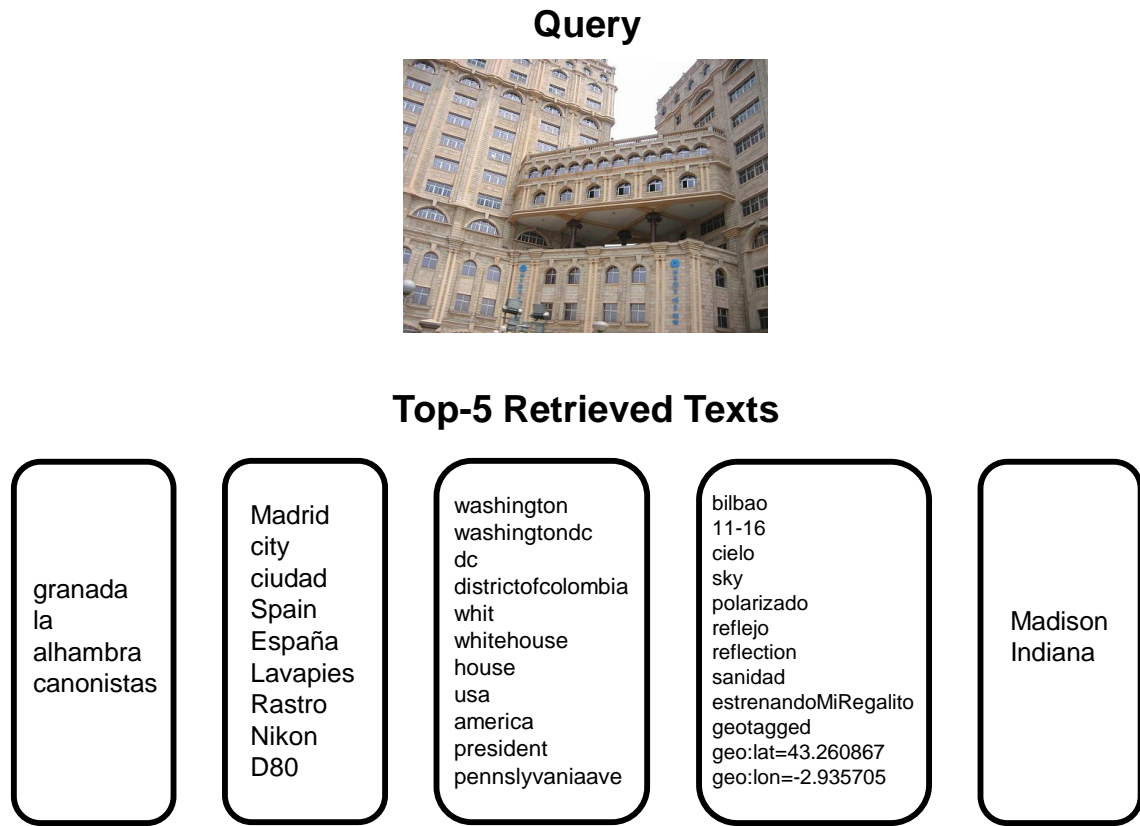
(b) Wiki: Text→Image

Figure 5.3: The top-5 retrieved results at 128 bits on Wiki dataset.

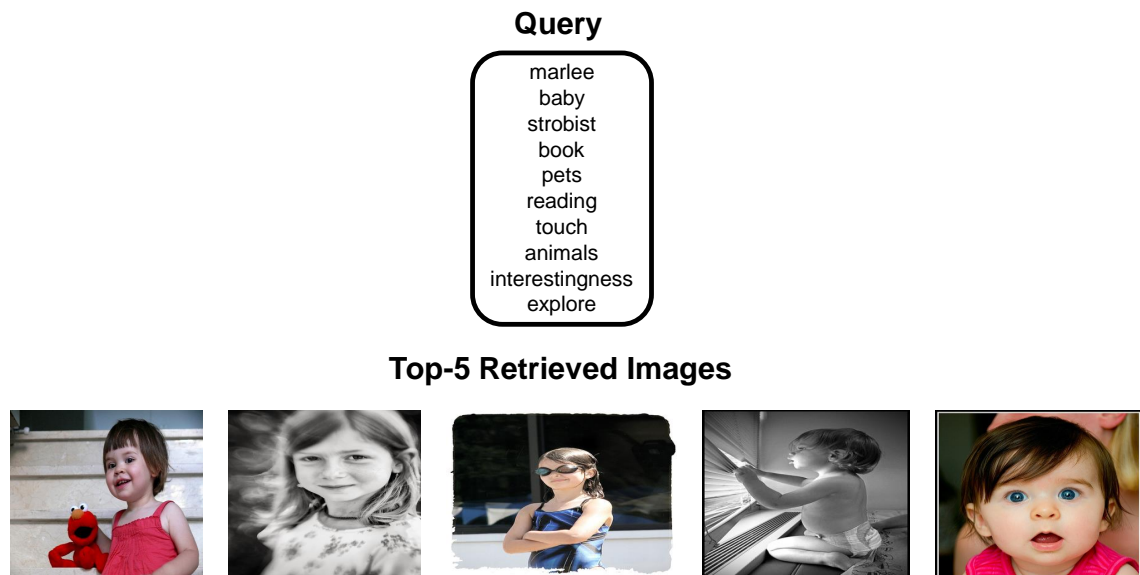
frequent concepts in NUS-WIDE dataset, where the query and its retrieved candidates are considered to be similar if at least one of their labels matched.

5.3.3.4 Effect of Training Data Size

Moreover, the variations on the mAP results are evaluated when using different amount of data points in the code learning, as shown in Table 5.9. Specifically,



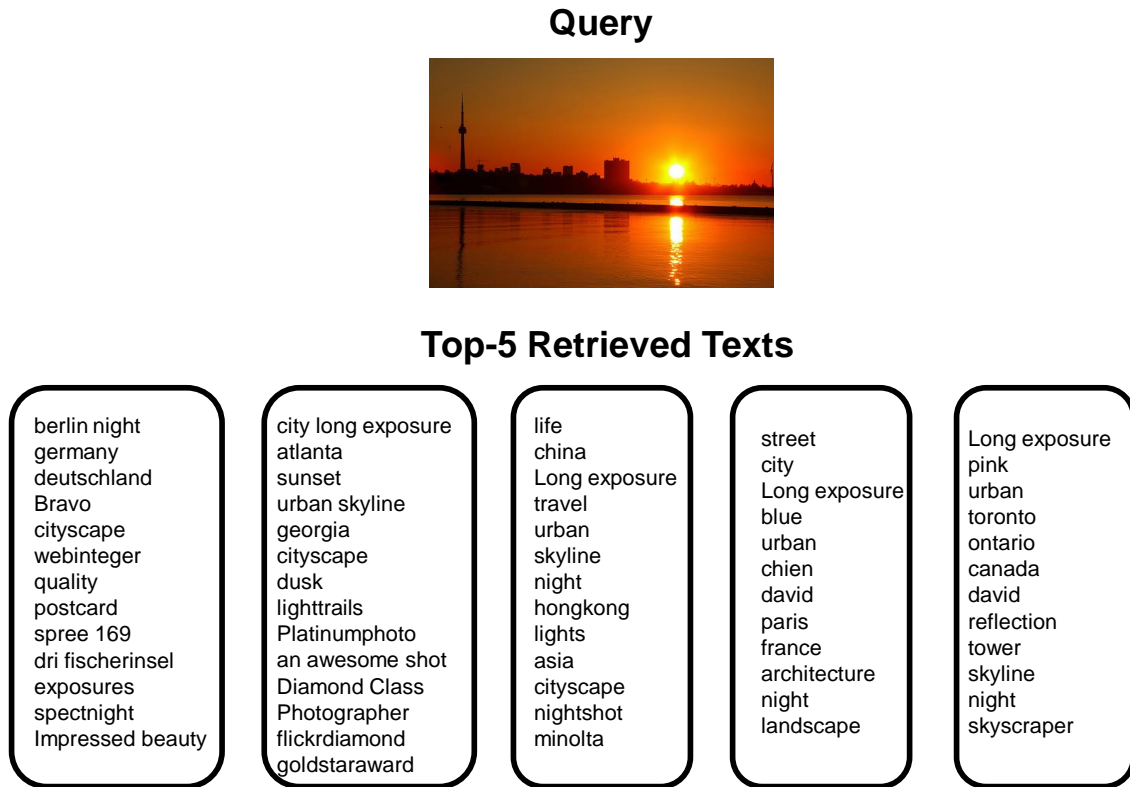
(a) MIRFlickr: Image→Text



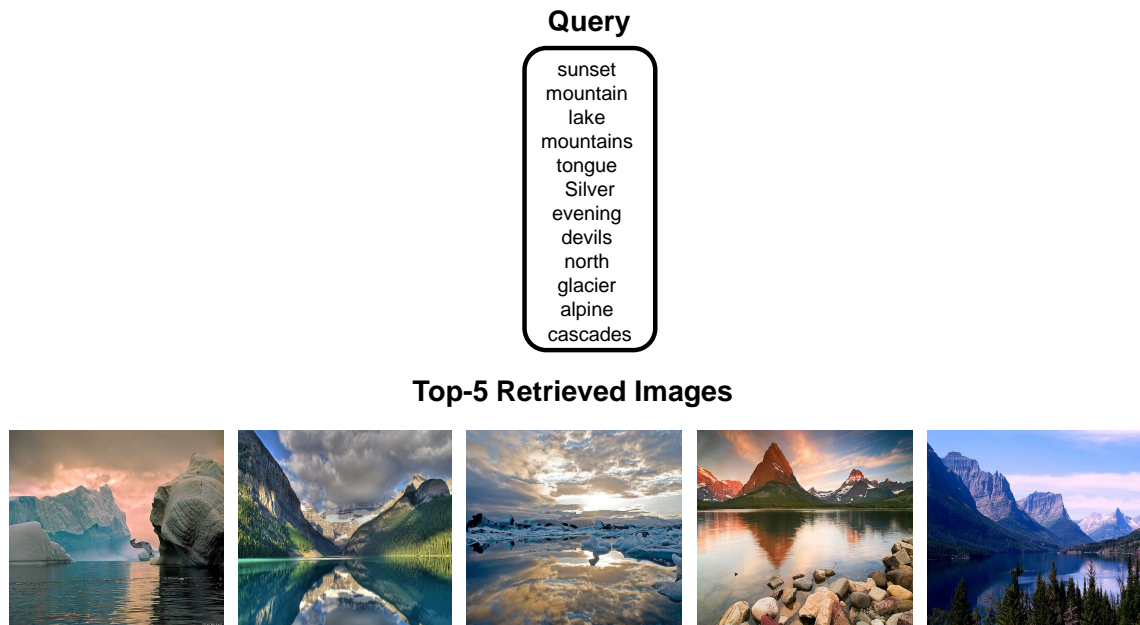
(b) MIRFlickr: Text→Image

Figure 5.4: The top-5 retrieved results at 128 bits on MIRFlickr dataset.

we report the results on MIRFlickr and NUS-WIDE at 64 bits on two retrieval tasks.



(a) NUS-WIDE: Image→Text



(b) NUS-WIDE: Text→Image

Figure 5.5: The top-5 retrieved results at 128 bits on NUS-WIDE dataset.

Table 5.6: Effect of training data size on MIRFlickr and NUS-WIDE at the code length of 64.

Dataset	Task	Training Data Size				
		1k	2k	5k	10k	15k
MIRFlickr	Image→Text	0.761	0.769	0.808	0.815	0.821
	Text→Image	0.793	0.811	0.843	0.854	0.861
NUS-WIDE	Image→Text	0.761	0.783	0.821	0.834	0.837
	Text→Image	0.781	0.803	0.833	0.842	0.85

As can be seen, the mAP values keep increasing with more data points employed in the initial stage and tend to converge after the size of 10,000. It is worth pointing out that SSDMH still achieves competitive results when limited data points (e.g., 5,000) available.

5.3.3.5 Parameter Sensitivity Analysis

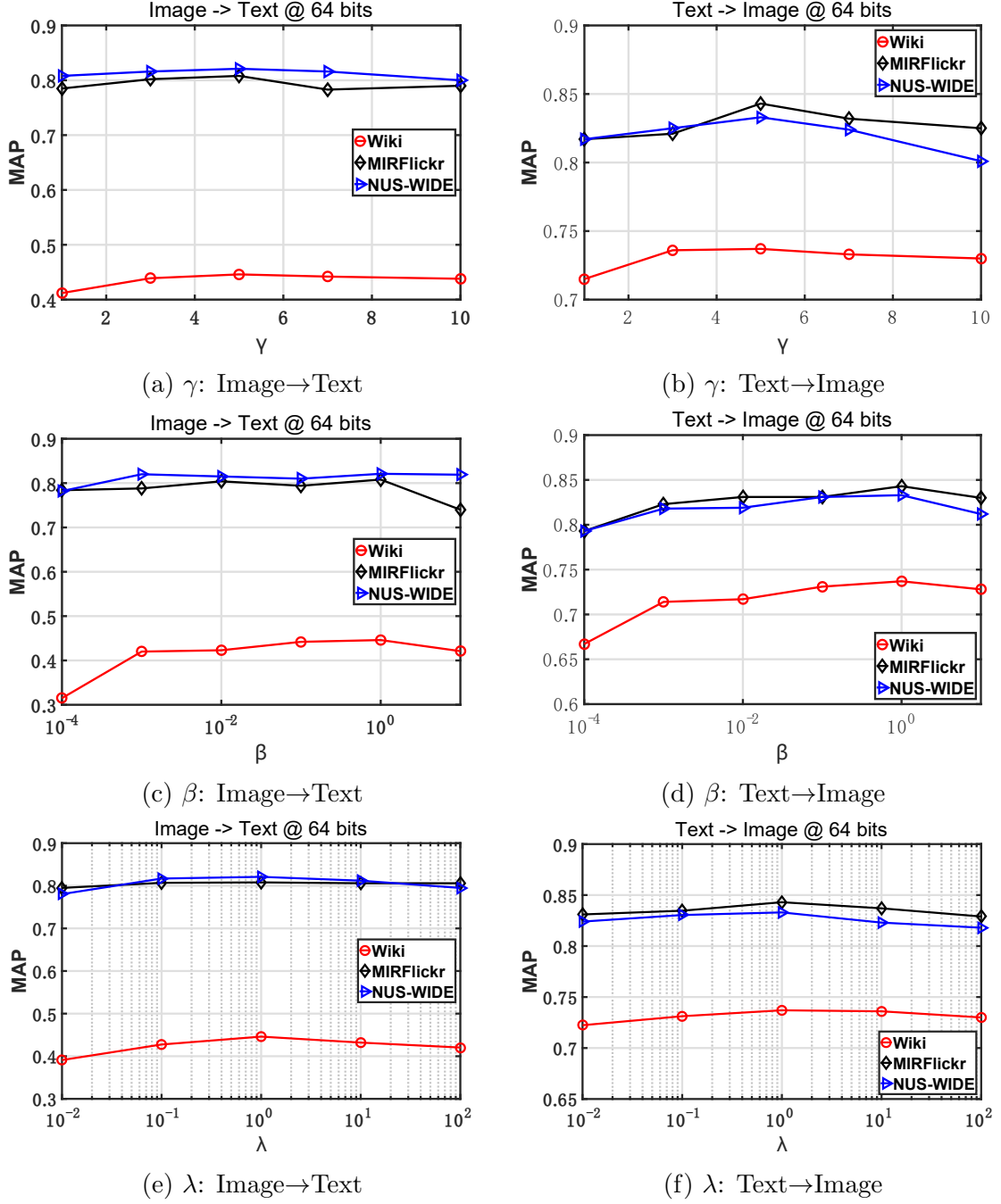
We further analyze the retrieval performance variations from adjusting the hyper-parameters in the code learning. By fixing the code length to 64 bits in the experiments, we plot the mAP variations in Fig. 5.6 when altering γ , β and λ . As can be seen, the mAP results have minor changes when varying the parameters, which indicates that SSDMH is not very sensitive to the hyper-parameters. Moreover, the empirical values of γ , β and λ are close to the optimal settings in the figures and they can make a great contribution in yielding superior retrieval performance.

5.3.3.6 Convergence Study

In Fig. 5.7, we plot to the estimation of the convergence rates in solving the unified binary code and learning the deep hash functions at 128 bits. As can be seen, the objective function values converge very fast within 5 iterations in the code learning, while the deep hash functions for two modalities can be built efficiently after $3 \sim 5$ iterations.

5.3.3.7 BGD versus One Entry

We further compare the efficiency of the proposed BGD and One Entry (namely flipping one entry per time) [153], where the latter one denotes the most representative method in the discrete optimization [126, 156]. As can be observed from Fig. 5.8,

Figure 5.6: mAP versus γ , β and λ at 64 bits on three datasets.

the proposed BGD costs much shorter time, averagely over 80%, against One Entry in solving one row of the unified binary code at 128 bits, thus accelerating the code optimization dramatically. Although some recent papers [54,126] make minor changes during the discrete optimization, they all utilize the same entry flipping strategy as One Entry. There is no evidence showing that such efficiency issue could be alleviated

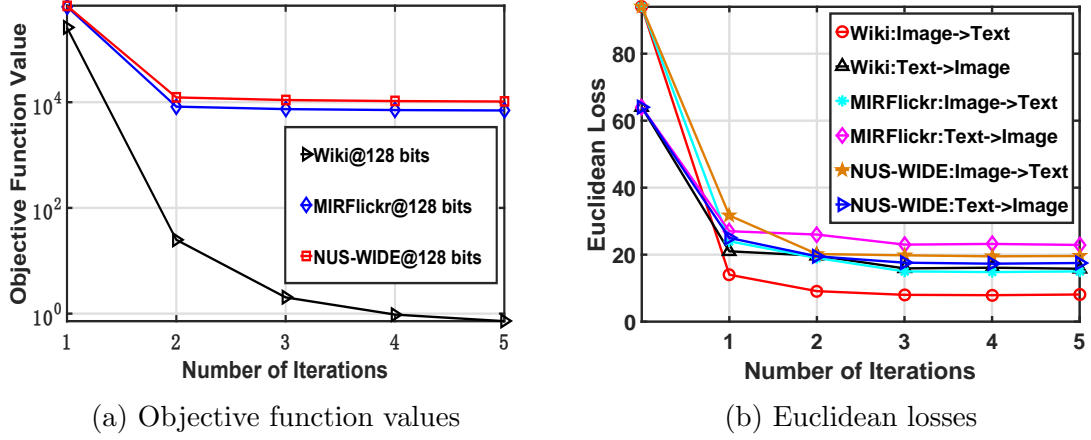


Figure 5.7: (a) Objective function values after each iteration (t) when solving the unified binary code at 128 bits; (b) Euclidean losses after every iteration (T) when learning the deep hash functions at 128 bits.

Table 5.7: Time costs (in seconds) in the training processes of SSDMH on three datasets at 128 bits for one loop (T).

Dataset	Code Learning	Network Training	
		Image	Text
Wiki	117.3	123.2	15.7
MIRFlickr	614.4	362.3	27.3
NUS-WIDE	582.1	413.1	38.3

in their methods.

5.3.3.8 Training Efficiency

Finally, the training costs of the proposed SSDMH at 128 bits on three datasets are reported in Table 5.7. There are two main sub processes: code learning and network training, during the optimization in each loop. As can be seen, the optimization for each loop can be done within 18 minutes for most cases. Considering the proposed SSDMH usually converges within $T = 5$ loops for one code length, the total optimization costs less than 1.5 hours while the values for other cases of short codes are far below.

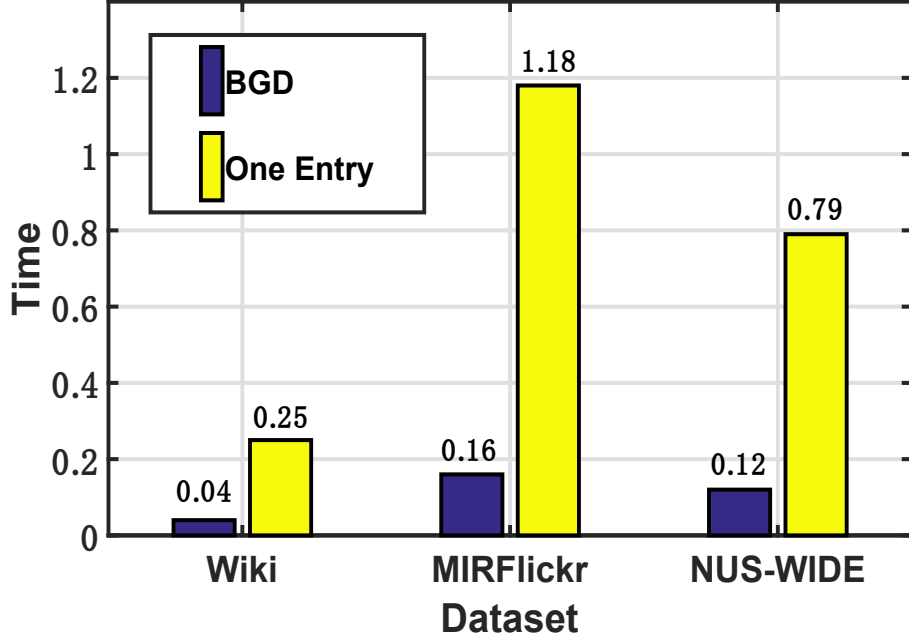


Figure 5.8: Time costs (in seconds) in optimizing one row of the unified binary code at 128 bits on three datasets when using BGD and One Entry [153] separately.

5.3.4 Quantitative Results for UDCMH

In this section, we briefly discuss the results from UDCMH to provide further insights on the advantages of the proposed learning strategy.

5.3.4.1 Comparison With State-of-The-Arts

We compare the proposed UDCMH with the baselines and report the mAP results at various code lengths on three datasets in Table 5.8. As can be seen, the proposed UDCMH outperforms significantly the state-of-the-arts baselines in terms of mAP at all bit sizes. Generally, the mAP values on Wiki are slightly lower than those on NUS-WIDE and MIRFlickr accordingly. The main reason is that the former dataset contains much fewer instances compared to the others, which limits the learning capability of deep neural networks. Specifically, for the tasks of Image→Text, the mAP values are 34.6%, 55.8% and 71.7% on Wiki, NUS-WIDE and MIRFlickr separately at 128 bits, which are at least 5% higher than those of the baselines. While for another task of Text→Image, the advantages of the proposed UDCMH decline slightly to about 3%, which are still superior and highly competitive. Compared to those baselines using matrix factorization such as CMFH and RFDH, the proposed framework integrates deep learning with collaborative binary latent representation model,

Table 5.8: mAP results for Image \rightarrow Text and Text \rightarrow Image tasks on three datasets at various code lengths (bits) when using different unsupervised methods and UDCMH. The Best Performance is shown in boldface.

Task	Method	Wiki				MIRFlickr				NUS-WIDE			
		16	32	64	128	16	32	64	128	16	32	64	128
Image \rightarrow Text	CVH [93]	0.179	0.162	0.153	0.149	0.606	0.599	0.596	0.598	0.372	0.362	0.406	0.39
	IMH [170]	0.201	0.203	0.204	0.195	0.612	0.601	0.592	0.579	0.47	0.473	0.476	0.459
	LCMH [251]	0.115	0.124	0.134	0.149	0.354	0.361	0.389	0.383	0.559	0.569	0.585	0.593
	CMFH [35]	0.251	0.253	0.259	0.263	0.621	0.624	0.625	0.627	0.455	0.459	0.465	0.467
	LSSH [249]	0.197	0.208	0.199	0.195	0.584	0.599	0.602	0.614	0.481	0.489	0.507	0.507
	DBRC [102]	0.253	0.265	0.269	0.288	0.617	0.619	0.62	0.621	0.424	0.459	0.447	0.447
Text \rightarrow Image	RDFH [189]	0.242	0.246	0.244	0.243	0.632	0.636	0.641	0.652	0.488	0.492	0.494	0.508
	UDCMH	0.309	0.318	0.329	0.346	0.689	0.698	0.714	0.717	0.511	0.519	0.524	0.558
	CVH [93]	0.252	0.235	0.171	0.154	0.591	0.583	0.576	0.576	0.401	0.384	0.442	0.432
	IMH [170]	0.467	0.478	0.453	0.456	0.603	0.595	0.589	0.58	0.478	0.483	0.472	0.462
	LCMH [251]	0.115	0.124	0.134	0.149	0.559	0.569	0.585	0.593	0.354	0.361	0.389	0.383
	CMFH [35]	0.595	0.601	0.616	0.622	0.642	0.662	0.676	0.685	0.529	0.577	0.614	0.645
Text \rightarrow Image	LSSH [249]	0.569	0.593	0.593	0.595	0.637	0.659	0.659	0.672	0.577	0.617	0.642	0.663
	DBRC [102]	0.574	0.588	0.598	0.599	0.618	0.626	0.626	0.628	0.455	0.459	0.468	0.473
	RDFH [189]	0.59	0.596	0.603	0.61	0.681	0.693	0.698	0.702	0.612	0.641	0.658	0.68
	UDCMH	0.622	0.633	0.645	0.658	0.692	0.704	0.718	0.733	0.637	0.653	0.695	0.716

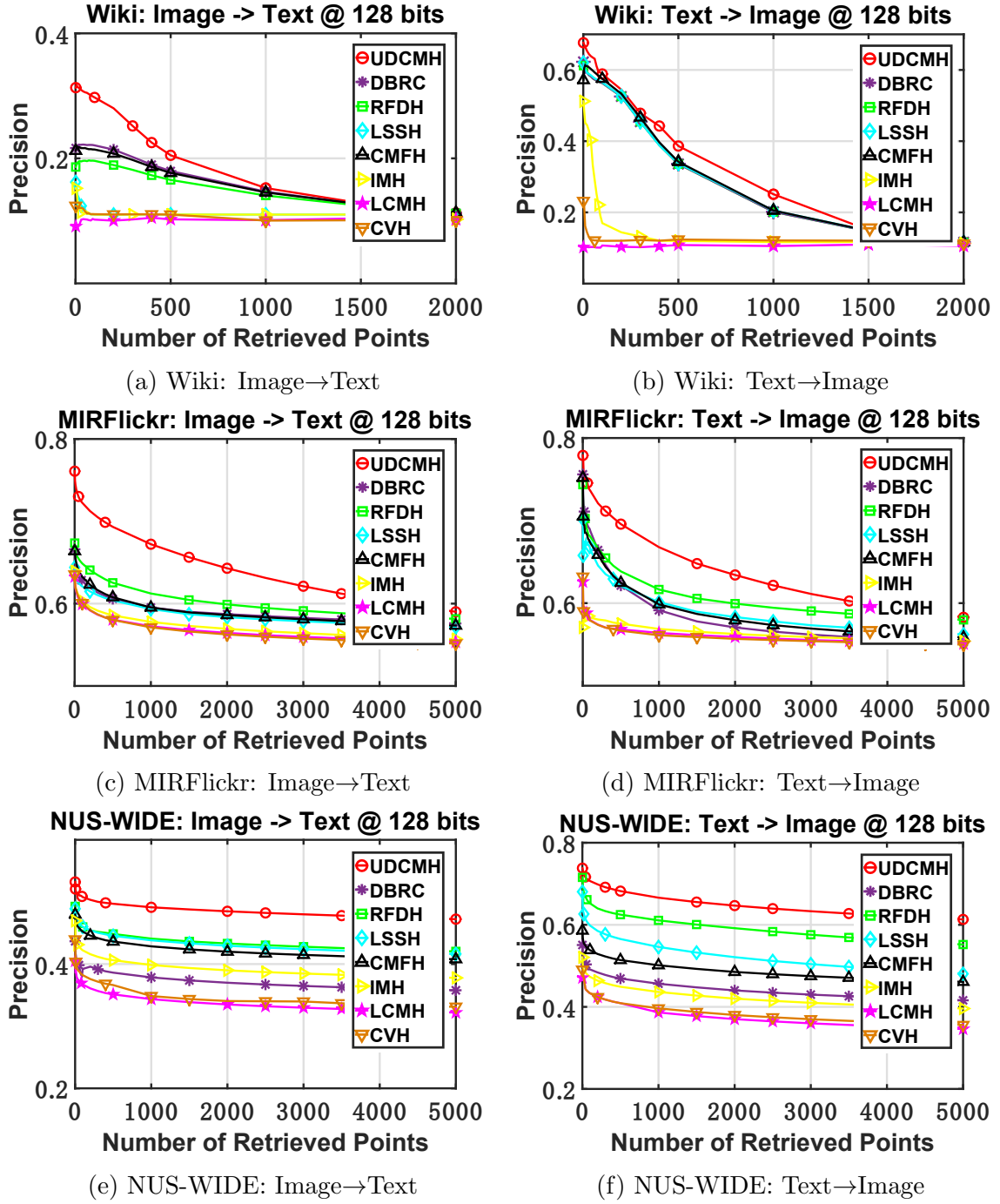


Figure 5.9: The precision@N curves at 128 bits on all datasets.

where unified binary codes can be optimized directly without relaxation, thus improving the hash code quality significantly. For comprehensive investigation, we also plot the Precision@N curves of various tasks at 128 bits on all datasets in Fig. 5.9. As observed from those figures, the best performance is still achieved by the proposed UDCMH and the claimed superiority can be further validated.

Table 5.9: Effect of training data size on NUS-WIDE at 64 bits. k indicates 1,000.

Dataset	Training Data Size				
	2k	5k	10k	15k	20k
Image→Text	0.515	0.524	0.538	0.549	0.557
Text→Image	0.668	0.695	0.698	0.717	0.724

5.3.4.2 Training Data Size

We further analyze the effects on mAP results when varying the training data size, as shown in Table 5.9. For the limited space, only the results on NUS-WIDE at 64 bits are reported. The mAP values keep increasing when utilizing more training data. It is worth noting that competitive results still can be achieved when using only 5,000 data points by UDCMH, which indicates its powerful ability in producing effective hash codes with the limited data size.

5.4 Chapter Summary

This section has provided an industrial solution for fast large-scale cross-media retrieval. Specifically, a novel self-supervised deep multimodal hashing method, namely SSDMH, is presented, where the deep feature learning and the semantic binary code learning are integrated into a unified framework. Notably, by solving the discrete constrained objective function in an alternating manner, the unified binary code can be generated directly without relaxation. Moreover, the semantic affinity matrix is utilized in the code learning with the neighborhood structure of the original data preserved. Besides, Binary Gradient Descent is proposed to *accelerate* the discrete optimization. The proposed algorithm is also extended to an unsupervised version termed UDCMH, which can be deployed in the applications lacking supervision information. Extensive experiments on three datasets demonstrate the superiority of the proposed methods over several state-of-the-art baselines.

The proposed method simulates the unsupervised multi-view embedding process via matrix factorization, as in Chapter 3. In other words, Chapter 5 can be viewed as an extension application of Chapter 3 in the research field of cross-modality retrieval but differs in many details. In the next chapter, we will summarize this thesis and discuss several possible research directions in the future.

Chapter 6

Conclusions and Future Work

6.1 Thesis Summary

This thesis mainly focuses on learning binary representations for efficient multimedia similarity retrieval using the hashing algorithm. Particularly, three different challenging tasks are explored: binary local descriptor (Chapter 3), unsupervised deep video hashing (Chapter 4) and deep cross-modal hashing (Chapter 5). In each chapter, a novel learning framework is proposed to handle the target application and the superior performance has been achieved against state-of-the-art baselines. Each chapter is briefly concluded and discussed in the following subsections.

6.1.1 Unsupervised Deep Binary Descriptor

In this chapter, we propose a novel learning-based feature descriptor, namely Unsupervised Deep Binary Descriptor (UDBD), to learn transformation invariant binary descriptors for patch-level recognition tasks. With the dedicated binary embedding model and weak bit scheme, the proposed binary descriptor outperforms most baselines significantly in terms of matching and retrieval accuracy on three public datasets.

To simplify the experiment, only rotation is performed on image patches to simulate the affine transformation. However, in real applications, other transformations like scaling and occlusion would impact the search performance from binary descriptors. This could be further investigated in the future work to evaluate the effectiveness of the proposed methodology when dealing with complicated cases. Moreover, the employed weak bit scheme could be enhanced by optimizing the mutual information between bits to make more wise selections on those unreliable bits, thus improving the matching performance further.

6.1.2 Unsupervised Deep Video Hashing

In this chapter, a novel video hashing framework named Unsupervised Deep Video Hashing (UDVH) is proposed to learn compact yet effective binary codes for large-scale video similarity search tasks. The major contribution of this work is that a smart rotation is applied to the video-specific features to balance the variance of dimensions, thus improving the retrieval performance with better code quality. Superior performance achieved by UDVH on three large-scale video datasets consolidates the claimed contribution.

In this work, a general video-to-video similarity retrieval is performed and discussed. However, the research on video retrieval could be further extended and investigated to tackle more realistic applications like use single frame to search a specific scene in long movies. Moreover, the overall loss function of UDVH is solved by iteratively optimizing two sub-objective functions. Performing the batch-level balanced rotation during the deep hash function learning might be a feasible solution to unify those optimization processes, thus yielding a more optimal solution mathematically. Last but not the least, more advanced video representation learning techniques, more than LSTM and TSN, could be incorporated in the video hashing framework to improve the overall retrieval performance.

6.1.3 Deep Cross-Modal Hashing

To deal with the cross-modal similarity retrieval problem, a novel method termed Self-Supervised Deep Multimodal Hashing (SSDMH) is proposed for large-scale cross-media search. Particularly, the hashing system based on the *binary* latent factor models can generate unified binary codes by solving a discrete-constrained objective function directly with no need for relaxation. Moreover, a new discrete optimization solution termed *Binary Gradient Descent*, which aims at improving the optimization efficiency towards the real-time operation. We also propose an unsupervised deep cross-modal hashing (UDCMH) at the end of this algorithm, which makes it more flexible in dealing with different scenarios. Extensive experiments on three benchmark datasets demonstrate the superiority of our methods over state-of-the-art cross-modal hashing approaches.

In the optimization of this work, the learning processes of binary code and hash function are jointly optimized. It implies that we have to update the hash function during each iteration of the discrete code learning, which leads to extra computational costs and reduces the efficiency. Moreover, the proposed method learns different hash

functions for two modalities (i.e., image and text), where an unified representation layer could be constructed and optimized in the learning process. Finally, data from more modalities (e.g., audio and video) can be employed to implement the multi-modal retrieval via adopting the proposed methodology for complicated recommendation systems.

6.2 Future Research Topics

In this section, we briefly discuss several possible research directions in the future, which are strictly related to the research topic in this thesis.

6.2.1 Hashing for Deep Binary Neural Network

In the past decade, deep Convolutional Neural Network (CNN) has been widely deployed in many computer vision tasks, which shows excellent performance in most cases. However, traditional CNN usually contains millions of network parameters that could occupy ample memory space. Moreover, large amounts of computational resources (e.g., high-level GPU) are required to run those CNNs successfully. Those two factors make it infeasible to deploy the complicated CNN directly on smart devices like mobile phones, camera, and pad. This motivates the research of network compression such that deep CNN could be deployed in such cases.

To this end, network binarization has been widely used in the network compression, where the network weights and activations are quantized into binary codes to reduce the memory space required for each parameter [29, 30, 147]. However, most of existing methods utilize quantization techniques or thresholding schemes in the binarization, which suffers from large quantization errors and yields unsatisfactory performance. In this case, the hashing algorithm could be an option to binarize the weights of the filters at each convolutional layer of the deep network, thus further reducing the network capacity with high discriminativeness.

6.2.2 Online Hashing

Recently, due to the urgent necessity of research for large-scale data, hashing has become a hot topic in the areas of machine learning and the data mining communities. However, there are rare online methods of hashing which have been proposed. Traditional hashing algorithms build the hash functions from the complicated training process and the performance is being tested with such fixed hash function for the

test data. There always exists an assumption that the train and test sets remain unchanged. However, in the real application scenarios, this assumption no longer holds, where the data will be streamed into the retrieval system. Nevertheless, traditional hashing algorithms have to learn the new hash function again, which is usually extremely time-consuming and computationally expensive.

Online hashing addresses this problem by only making minor changes in the learning objectives to update the learned hash function [76] when getting new data. That avoids the redundant retrain process and makes online hashing a promising solution in tackling large-scale online similarity retrieval problems. However, only a few efforts have been made on this research topic, such as Online Kernel-based Hashing (OKH) [76] and Online Sketching Hashing (OSH) [96]. More worriedly, the results from these shallow learning based methods are still far from satisfactory. Our proposed methods in this thesis could be used as backbones to achieve novel fine-grained online hashing algorithms.

6.2.3 Fine-Grained Retrieval with Weighted Hamming Distance

As discussed in Chapter 3, the candidates are returned to a specific query by directly measuring the Hamming distances between them in the test phase for most existing hashing algorithms. This can be treated as coarse-level retrieval. However, in many application cases, only one exact candidate is expected to be returned for a given query. Therefore, fine-grained retrieval is desirable in solving this problem, where weighted hamming distance (i.e., bit weighting) is a promising solution in this case [119, 167, 168].

Nevertheless, the binary code usually contains less information compared to the real-valued feature descriptor because of the quantization, which makes the fine-grained retrieval more challenging. In recent applications of bit weighting [119, 120], the dynamic weighting assignment scheme is adopted to further improve the ranking precision of binary descriptor in the matching and retrieval tasks, where large weights should be assigned to important bits in the distance measurement and vice versa [44, 80, 119]. The bit significance is usually determined by jointly utilizing the real-valued representations, as in Chapter 5. However, extra computations and memory space are required. It is more desirable to develop an advanced bit weighting scheme that obtains the weights based on the binary code directly, where the entropy and mutual information could be further utilized to generate the weight for each bit [15].

6.2.4 Fast Person Re-Identification

Pedestrian re-identification has become a popular topic in the research field of computer vision and it is widely adopted in applications such as human-computer interaction, security surveillance, and criminal detection [128].

The research goal of pedestrian re-identification is to find the same/similar pedestrian captured by different cameras or one camera at different times, which can be viewed as another promising application of instance-level (i.e., pedestrian) similarity retrieval. Traditional pedestrian re-identification works mainly focus on the feature learning, namely extract representative feature like global features, pedestrian image color, texture, edge, shape, for the next re-identification step [53, 110].

Even though the considerable efforts have been made, it is still an open problem due to the dramatic variations caused by different camera viewpoints and person pose changes [128, 250]. The multi-view embedding could be a possible solution in tackling these problems as discussed in Chapter 3 and the pedestrian search process can be accelerated significantly by using the binary descriptor.

Bibliography

- [1] A. Alahi, R. Ortiz, and P. Vandergheynst, “Freak: Fast retina keypoint,” in *Computer Vision and Pattern Recognition*, 2012, pp. 510–517.
- [2] A. Andoni and P. Indyk, “Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions,” in *Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 2006, pp. 459–468.
- [3] V. Balntas, K. Lenc, A. Vedaldi, and K. Mikolajczyk, “Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors,” in *Computer Vision and Pattern Recognition*, 2017, pp. 5173–5182.
- [4] V. Balntas, L. Tang, and K. Mikolajczyk, “Bold-binary online learned descriptor for efficient image matching,” in *Computer Vision and Pattern Recognition*, 2015, pp. 2367–2375.
- [5] S. Baluja and M. Covell, “Beyond ‘near duplicates’: Learning hash codes for efficient similar-image retrieval,” in *International Conference on Pattern Recognition*. IEEE, 2010, pp. 543–547.
- [6] H. Bay, T. Tuytelaars, and L. Van Gool, “Surf: Speeded up robust features,” in *European Conference on Computer Vision*. Springer, 2006, pp. 404–417.
- [7] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.
- [8] L. Bottou, “Large-scale machine learning with stochastic gradient descent,” in *Proceedings of International Conference on COMPUTATIONAL STATISTICS (COMPSTAT)*. Springer, 2010, pp. 177–186.
- [9] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. O’Reilly Media, Inc., 2008.

- [10] M. M. Bronstein, A. M. Bronstein, F. Michel, and N. Paragios, “Data fusion through cross-modality metric learning using similarity-sensitive hashing,” in *Computer Vision and Pattern Recognition*, 2010, pp. 3594–3601.
- [11] M. Brown, G. Hua, and S. Winder, “Discriminative learning of local image descriptors,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 43–57, 2010.
- [12] M. Brown and D. G. Lowe, “Unsupervised 3d object recognition and reconstruction in unordered datasets,” in *3DIM*, 2005, pp. 56–63.
- [13] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, “Spectral networks and locally connected networks on graphs,” *arXiv preprint arXiv:1312.6203*, 2013.
- [14] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. Carlos Niebles, “Activitynet: A large-scale video benchmark for human activity understanding,” in *Computer Vision and Pattern Recognition*, 2015, pp. 961–970.
- [15] F. Cakir, K. He, S. A. Bargal, and S. Sclaroff, “Hashing with mutual information,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 10, pp. 2424–2437, 2019.
- [16] F. Cakir and S. Sclaroff, “Adaptive hashing for fast similarity search,” in *International Conference on Computer Vision*, 2015, pp. 1044–1052.
- [17] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “Brief: Binary robust independent elementary features,” in *European Conference on Computer Vision*. Springer, 2010, pp. 778–792.
- [18] L. Cao, Z. Li, Y. Mu, and S.-F. Chang, “Submodular video hashing: a unified framework towards video pooling and indexing,” in *ACM Multimedia*. ACM, 2012, pp. 299–308.
- [19] Y. Cao, M. Long, J. Wang, and S. Liu, “Collective deep quantization for efficient cross-modal retrieval,” in *AAAI conference on Artificial Intelligence*, 2017, pp. 3974–3980.
- [20] Y. Cao, M. Long, J. Wang, Q. Yang, and P. S. Yu, “Deep visual-semantic hashing for cross-modal retrieval,” in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 1445–1454.

- [21] Y. Cao, M. Long, J. Wang, and H. Zhu, “Correlation autoencoder hashing for supervised cross-modal search,” in *Proceedings of the ACM on International Conference on Multimedia Retrieval*. ACM, 2016, pp. 197–204.
- [22] M. S. Charikar, “Similarity estimation techniques from rounding algorithms,” in *Proceedings of the Annual ACM Symposium on Theory of Computing*. ACM, 2002, pp. 380–388.
- [23] Z. Chen, J. Lu, J. Feng, and J. Zhou, “Nonlinear discrete hashing,” *IEEE Transactions on Multimedia*, vol. 19, no. 1, pp. 123–135, 2017.
- [24] —, “Nonlinear structural hashing for scalable video search,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 6, pp. 1421–1433, 2018.
- [25] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng, “Nus-wide: a real-world web image database from national university of singapore,” in *Proceedings of the ACM International Conference on Image and Video Retrieval*. ACM, 2009, pp. 1–9.
- [26] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” *arXiv preprint arXiv:1412.3555*, 2014.
- [27] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [28] B. Coskun, B. Sankur, and N. Memon, “Spatio-temporal transform based video hashing,” *IEEE Transactions on Multimedia*, vol. 8, no. 6, pp. 1190–1208, 2006.
- [29] M. Courbariaux, Y. Bengio, and J.-P. David, “Binaryconnect: Training deep neural networks with binary weights during propagations,” in *Advances in Neural Information Processing Systems*, 2015, pp. 3123–3131.
- [30] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, “Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1,” *arXiv preprint arXiv:1602.02830*, 2016.
- [31] J. Davis and M. Goadrich, “The relationship between precision-recall and roc curves,” in *International Conference on Machine Intelligence*. ACM, 2006, pp. 233–240.

- [32] C. Deng, Z. Chen, X. Liu, X. Gao, and D. Tao, “Triplet-based deep hashing network for cross-modal retrieval,” *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 3893–3903, 2018.
- [33] G. Ding, W. Chen, s. Zhao, J. Han, and Q. Liu, “Real-time scalable visual tracking via quadrangle kernelized correlation filters,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 140–150, 2018.
- [34] G. Ding, Y. Guo, and J. Zhou, “Collective matrix factorization hashing for multimodal data,” in *Computer Vision and Pattern Recognition*, 2014, pp. 2075–2082.
- [35] G. Ding, Y. Guo, J. Zhou, and Y. Gao, “Large-scale cross-modality search via collective matrix factorization hashing,” *IEEE Transactions on Image Processing*, vol. 25, no. 11, pp. 5427–5440, 2016.
- [36] X. Dong, L. Yu, Z. Wu, Y. Sun, L. Yuan, and F. Zhang, “A hybrid collaborative filtering model with deep structure for recommender systems,” in *AAAI Conference on Artificial Intelligence*, 2017, pp. 1309–1315.
- [37] A. Dosovitskiy, P. Fischer, J. T. Springenberg, M. Riedmiller, and T. Brox, “Discriminative unsupervised feature learning with exemplar convolutional neural networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 9, pp. 1734–1747, 2015.
- [38] M. Douze, H. Jégou, and C. Schmid, “An image-based approach to video copy detection with spatio-temporal post-filtering,” *IEEE Transactions on Multimedia*, vol. 12, no. 4, pp. 257–266, 2010.
- [39] Y. Duan, J. Lu, J. Feng, and J. Zhou, “Learning rotation-invariant local binary descriptor,” *IEEE Transactions on Image Processing*, vol. 26, no. 8, pp. 3636–3651, 2017.
- [40] —, “Context-aware local binary feature learning for face recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 5, pp. 1139–1153, 2018.
- [41] Y. Duan, J. Lu, Z. Wang, J. Feng, and J. Zhou, “Learning deep binary descriptor with multi-quantization,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 8, pp. 1924–1938, 2019.

- [42] Y. Duan, Z. Wang, J. Lu, X. Lin, and J. Zhou, “Graphbit: Bitwise interaction mining via deep reinforcement learning,” in *Computer Vision and Pattern Recognition*, 2018, pp. 8270–8279.
- [43] V. Erin Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, “Deep hashing for compact binary codes learning,” in *Computer Vision and Pattern Recognition*, 2015, pp. 2475–2483.
- [44] H. Fan, H.-M. Hu, R. Wang, and Y. Zhang, “Adaptive query re-ranking based on imagegraph for image retrieval,” in *IEEE International Conference on Big Data*. IEEE, 2018, pp. 4593–4599.
- [45] L. Fei, B. Zhang, Y. Xu, Z. Guo, J. Wen, and W. Jia, “Learning discriminant direction binary palmprint descriptor,” *IEEE Transactions on Image Processing*, vol. 28, no. 8, pp. 3808–3820, 2019.
- [46] J. H. Friedman, J. L. Bentley, and R. A. Finkel, “An algorithm for finding best matches in logarithmic expected time,” *ACM Transactions on Mathematical Software (TOMS)*, vol. 3, no. 3, pp. 209–226, 1977.
- [47] Y. Gao, M. Wang, R. Ji, X. Wu, and Q. Dai, “3-d object retrieval with hausdorff distance learning,” *IEEE Transactions on Industrial Electronics*, vol. 61, no. 4, pp. 2088–2098, 2014.
- [48] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with lstm,” *Neural Computation*, pp. 2451–2471, 2000.
- [49] K. Ghasedi Dizaji, F. Zheng, N. Sadoughi, Y. Yang, C. Deng, and H. Huang, “Unsupervised deep generative adversarial hashing network,” in *Computer Vision and Pattern Recognition*, 2018, pp. 3664–3673.
- [50] A. Gionis, P. Indyk, R. Motwani *et al.*, “Similarity search in high dimensions via hashing,” in *International Conference on Very Large Data Bases*, vol. 99, no. 6, 1999, pp. 518–529.
- [51] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, “Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 12, pp. 2916–2929, 2013.

- [52] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.
- [53] D. Gray, S. Brennan, and H. Tao, “Evaluating appearance models for recognition, reacquisition, and tracking,” in *IEEE International Workshop on Performance Evaluation for Tracking and Surveillance (PETS)*, vol. 3, no. 5. Citeseer, 2007, pp. 1–7.
- [54] J. Gui and P. Li, “R 2 sdh: Robust rotated supervised discrete hashing,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2018, pp. 1485–1493.
- [55] J. Gui, T. Liu, Z. Sun, D. Tao, and T. Tan, “Fast supervised discrete hashing,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 2, pp. 490–496, 2018.
- [56] Y. Guo, G. Ding, and J. Han, “Robust quantization for general similarity search,” *IEEE Transactions on Image Processing*, vol. 27, no. 2, pp. 949–963, 2018.
- [57] Y. Guo, G. Ding, J. Han, and X. Jin, “Robust iterative quantization for efficient l p-norm similarity search,” in *International Joint Conference on Artificial Intelligence*, 2016, pp. 3382–3388.
- [58] Y. Guo, G. Ding, L. Liu, J. Han, and L. Shao, “Learning to hash with optimized anchor embedding for scalable retrieval,” *IEEE Transactions on Image Processing*, vol. 26, no. 3, pp. 1344–1354, 2017.
- [59] J. Haitsma and T. Kalker, “A highly robust audio fingerprinting system.” in *Ismir*, vol. 2002, 2002, pp. 107–115.
- [60] J. Han and G. C. Langelaar, “Method and device for generating fingerprints of information signals,” 2018, uS Patent 10,248,723.
- [61] J. Han, E. Pauwels, and P. de Zeeuw, “Fast saliency-aware multi-modality image fusion,” *Neurocomputing*, vol. 111, pp. 70–80, 2013.
- [62] J. Han, E. Pauwels, P. de Zeeuw, and P. de With, “Employing a rgb-d sensor for real-time tracking of humans across multiple re-entries in a smart environment.” *IEEE Transactions on Consumer Electronics*, vol. 58, no. 2, pp. 255–263, 2012.

- [63] J. Han, R. Quan, D. Zhang, and F. Nie, “Robust object co-segmentation using background prior,” *IEEE Transactions on Image Processing*, vol. 27, no. 4, pp. 1639–1651, 2018.
- [64] J. Han, D. Zhang, G. Cheng, N. Liu, and D. Xu, “Advanced deep-learning techniques for salient and category-specific object detection: A survey,” *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 84–100, 2018.
- [65] Y. Hao, T. Mu, J. Y. Goulermas, J. Jiang, R. Hong, and M. Wang, “Unsupervised t-distributed video hashing and its deep hashing extension,” *IEEE Transactions on Image Processing*, vol. 26, no. 11, pp. 5531–5544, 2017.
- [66] Y. Hao, T. Mu, R. Hong, M. Wang, N. An, and J. Y. Goulermas, “Stochastic multiview hashing for large-scale near-duplicate video retrieval,” *IEEE Transactions on Multimedia*, vol. 19, no. 1, pp. 1–14, 2017.
- [67] D. R. Hardoon, S. Szedmak, and J. Shawe-Taylor, “Canonical correlation analysis: An overview with application to learning methods,” *Neural Computation*, vol. 16, no. 12, pp. 2639–2664, 2004.
- [68] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge University Press, 2003.
- [69] K. He, F. Wen, and J. Sun, “K-means hashing: An affinity-preserving quantization method for learning binary compact codes,” in *Computer Vision and Pattern Recognition*, 2013, pp. 2938–2945.
- [70] K. He, Y. Lu, and S. Sclaroff, “Local descriptors optimized for average precision,” in *Computer Vision and Pattern Recognition*, 2018, pp. 596–605.
- [71] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon, “Spherical hashing,” in *Computer Vision and Pattern Recognition*, 2012, pp. 2957–2964.
- [72] D. S. Hochba, “Approximation algorithms for np-hard problems,” *ACM Sigact News*, vol. 28, no. 2, pp. 40–52, 1997.
- [73] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [74] W. Hu, G.-P. Liu, and H. Zhou, “Web-based 3-d control laboratory for remote real-time experimentation,” *IEEE Transactions on Industrial Electronics*, vol. 60, no. 10, pp. 4673–4682, 2013.

-
- [75] Y. Hu, Z. Jin, H. Ren, D. Cai, and X. He, “Iterative multi-view hashing for cross media indexing,” in *ACM Multimedia*. ACM, 2014, pp. 527–536.
 - [76] L.-K. Huang, Q. Yang, and W.-S. Zheng, “Online hashing,” in *International Joint Conference on Artificial Intelligence*, 2013, pp. 1422–1428.
 - [77] M. J. Huiskes and M. S. Lew, “The mir flickr retrieval evaluation,” in *ACM Multimedia*. ACM, 2008, pp. 39–43.
 - [78] P. Indyk and R. Motwani, “Approximate nearest neighbors: towards removing the curse of dimensionality,” in *Proceedings of the Annual ACM Symposium on Theory of Computing*. ACM, 1998, pp. 604–613.
 - [79] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *International Conference on Machine Learning*, 2015, pp. 448–456.
 - [80] H. Jegou, M. Douze, and C. Schmid, “Product quantization for nearest neighbor search,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 117–128, 2010.
 - [81] H. Jégou, M. Douze, C. Schmid, and P. Pérez, “Aggregating local descriptors into a compact image representation,” in *Computer Vision and Pattern Recognition*, 2010, pp. 3304–3311.
 - [82] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional architecture for fast feature embedding,” in *ACM Multimedia*. ACM, 2014, pp. 675–678.
 - [83] Q.-Y. Jiang and W.-J. Li, “Discrete latent factor model for cross-modal hashing,” *IEEE Transactions on Image Processing*, vol. 28, no. 7, pp. 3490–3501, 2019.
 - [84] Q. Jiang and W. Li, “Deep cross-modal hashing,” *arXiv preprint arXiv:1602.02255*, 2016.
 - [85] W. Jiang, F. Nie, and H. Huang, “Robust dictionary learning with capped l1-norm,” in *International Joint Conference on Artificial Intelligence*, 2015, pp. 3590–3596.

-
- [86] A. Karpathy, J. Johnson, and L. Fei-Fei, “Visualizing and understanding recurrent networks,” *arXiv preprint arXiv:1506.02078*, 2015.
 - [87] W. Kong and W.-J. Li, “Double-bit quantization for hashing,” in *AAAI Conference on Artificial Intelligence*, 2012, pp. 634–640.
 - [88] —, “Isotropic hashing,” in *Advances in Neural Information Processing Systems*, 2012, pp. 1646–1654.
 - [89] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” Citeseer, Tech. Rep., 2009.
 - [90] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.
 - [91] B. Kulis and T. Darrell, “Learning to hash with binary reconstructive embeddings,” in *Advances in Neural Information Processing Systems*, 2009, pp. 1042–1050.
 - [92] B. Kulis and K. Grauman, “Kernelized locality-sensitive hashing for scalable image search,” in *International Conference on Computer Vision*, vol. 9, 2009, pp. 2130–2137.
 - [93] S. Kumar and R. Udupa, “Learning hash functions for cross-view similarity search,” in *International Joint Conference on Artificial Intelligence*, 2011, pp. 1360–1365.
 - [94] H. Lai, Y. Pan, Y. Liu, and S. Yan, “Simultaneous feature learning and hash coding with deep neural networks,” in *Computer Vision and Pattern Recognition*, 2015, pp. 3270–3278.
 - [95] Z. Lan, Y. Zhu, A. G. Hauptmann, and S. Newsam, “Deep local video feature for action recognition,” in *Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 2017, pp. 1219–1225.
 - [96] C. Leng, J. Wu, J. Cheng, X. Bai, and H. Lu, “Online sketching hashing,” in *Computer Vision and Pattern Recognition*, 2015, pp. 2503–2511.
 - [97] S. Leutenegger, M. Chli, and R. Y. Siegwart, “Brisk: Binary robust invariant scalable keypoints,” in *International Conference on Computer Vision*. IEEE, 2011, pp. 2548–2555.

-
- [98] C. Li, S. Gao, C. Deng, D. Xie, and W. Liu, “Cross-modal learning with adversarial samples,” in *Advances in Neural Information Processing Systems*, 2019, pp. 10 791–10 801.
 - [99] M. Li and V. Monga, “Robust video hashing via multilinear subspace projections,” *IEEE Transactions on Image Processing*, vol. 21, no. 10, pp. 4397–4409, 2012.
 - [100] S. Li, Z. Chen, J. Lu, X. Li, and J. Zhou, “Neighborhood preserving hashing for scalable video retrieval,” in *International Conference on Computer Vision*, 2019, pp. 8212–8221.
 - [101] X. Li, G. Lin, C. Shen, A. v. d. Hengel, and A. Dick, “Learning hash functions using column generation,” *arXiv preprint arXiv:1303.0339*, 2013.
 - [102] X. Li, D. Hu, and F. Nie, “Deep binary reconstruction for cross-modal hashing,” *arXiv preprint arXiv:1708.05127*, 2017.
 - [103] —, “Large graph hashing with spectral rotation,” in *AAAI Conference on Artificial Intelligence*, 2017, pp. 2203–2209.
 - [104] Y. Li, R. Wang, Z. Huang, S. Shan, and X. Chen, “Face video retrieval with image query via hashing across euclidean space and riemannian manifold,” in *Computer Vision and Pattern Recognition*, 2015, pp. 4758–4767.
 - [105] G. Lin, C. Shen, D. Suter, and A. Van Den Hengel, “A general two-step approach to learning-based hashing,” in *International Conference on Computer Vision*, 2013, pp. 2552–2559.
 - [106] K. Lin, J. Lu, C.-S. Chen, J. Zhou, and M.-T. Sun, “Unsupervised deep learning of compact binary descriptors,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 6, pp. 1501–1514, 2018.
 - [107] Z. Lin, G. Ding, J. Han, and J. Wang, “Cross-view retrieval via probability-based semantics-preserving hashing,” *IEEE Transactions on Cybernetics*, vol. 47, no. 12, pp. 4342–4355, 2017.
 - [108] Z. Lin, G. Ding, M. Hu, and J. Wang, “Semantics-preserving hashing for cross-view retrieval,” in *Computer Vision and Pattern Recognition*, 2015, pp. 3864–3872.

- [109] V. E. Liong, J. Lu, Y.-P. Tan, and J. Zhou, “Deep video hashing,” *IEEE Transactions on Multimedia*, vol. 19, no. 6, pp. 1209–1219, 2017.
- [110] C. Liu, S. Gong, C. C. Loy, and X. Lin, “Person re-identification: What features are important?” in *European Conference on Computer Vision*. Springer, 2012, pp. 391–401.
- [111] H. Liu, R. Wang, S. Shan, and X. Chen, “Deep supervised hashing for fast image retrieval,” in *Computer Vision and Pattern Recognition*, 2016, pp. 2064–2072.
- [112] J. Liu, S. Ji, and J. Ye, “Multi-task feature learning via efficient l_2 , l_1 -norm minimization,” in *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2009, pp. 339–348.
- [113] L. Liu, Z. Lin, L. Shao, F. Shen, G. Ding, and J. Han, “Sequential discrete hashing for scalable cross-modality similarity retrieval,” *IEEE Transactions on Image Processing*, vol. 26, no. 1, pp. 107–118, 2017.
- [114] L. Liu, L. Shao, F. Shen, and M. Yu, “Discretely coding semantic rank orders for supervised image hashing,” in *Computer Vision and Pattern Recognition*, 2017, pp. 1425–1434.
- [115] L. Liu, M. Yu, and L. Shao, “Latent structure preserving hashing,” *International Journal of Computer Vision*, vol. 122, no. 3, pp. 439–457, 2017.
- [116] W. Liu, C. Mu, S. Kumar, and S.-F. Chang, “Discrete graph hashing,” in *Advances in Neural Information Processing Systems*, 2014, pp. 3419–3427.
- [117] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, “Supervised hashing with kernels,” in *Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 2074–2081.
- [118] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, “Hashing with graphs,” 2011.
- [119] X. Liu, C. Deng, B. Lang, D. Tao, and X. Li, “Query-adaptive reciprocal hash tables for nearest neighbor search,” *IEEE Transactions on Image Processing*, vol. 25, no. 2, pp. 907–919, 2015.
- [120] X. Liu, J. He, B. Lang, and S.-F. Chang, “Hash bit selection: a unified solution for selection problems in hashing,” in *Computer Vision and Pattern Recognition*, 2013, pp. 1570–1577.

- [121] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.
- [122] J. Lu, V. E. Liong, and J. Zhou, “Deep hashing for scalable image search,” *IEEE Transactions on Image Processing*, vol. 26, no. 5, pp. 2352–2367, 2017.
- [123] —, “Simultaneous local binary feature learning and encoding for homogeneous and heterogeneous face recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 8, pp. 1979–1993, 2017.
- [124] J. Lu, V. E. Liong, X. Zhou, and J. Zhou, “Learning compact binary face descriptor for face recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 10, pp. 2041–2056, 2015.
- [125] M. Luo, X. Chang, Z. Li, L. Nie, A. G. Hauptmann, and Q. Zheng, “Simple to complex cross-modal learning to rank,” *Computer Vision and Image Understanding*, vol. 163, pp. 67–77, 2017.
- [126] Y. Luo, Y. Yang, F. Shen, Z. Huang, P. Zhou, and H. T. Shen, “Robust discrete code modeling for supervised hashing,” *Pattern Recognition*, vol. 75, pp. 128–135, 2018.
- [127] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, “Multi-probe lsh: efficient indexing for high-dimensional similarity search,” in *International Conference on Very Large Data Bases*. VLDB Endowment, 2007, pp. 950–961.
- [128] X.-Q. Ma, C.-C. Yu, X.-X. Chen, and L. Zhou, “Large-scale person re-identification based on deep hash learning,” *Entropy*, vol. 21, no. 5, p. 449, 2019.
- [129] K. Mikolajczyk and C. Schmid, “Scale & affine invariant interest point detectors,” *International Journal of Computer Vision*, vol. 60, no. 1, pp. 63–86, 2004.
- [130] A. Mishchuk, D. Mishkin, F. Radenovic, and J. Matas, “Working hard to know your neighbor’s margins: Local descriptor learning loss,” in *Advances in Neural Information Processing Systems*, 2017, pp. 4826–4837.
- [131] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, “Deepfool: a simple and accurate method to fool deep neural networks,” in *Computer Vision and Pattern Recognition*, 2016, pp. 2574–2582.

- [132] M. Muja and D. G. Lowe, “Scalable nearest neighbor algorithms for high dimensional data,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2227–2240, 2014.
- [133] E. R. Nascimento, G. Potje, R. Martins, F. Cadar, M. F. Campos, and R. Bajcsy, “Geobit: A geodesic-based binary descriptor invariant to non-rigid deformations for rgb-d images,” in *International Conference on Computer Vision*, 2019, pp. 10 004–10 012.
- [134] X. Nie, W. Jing, C. Cui, J. Zhang, L. Zhu, and Y. Yin, “Joint multi-view hashing for large-scale near-duplicate video retrieval,” *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2019.
- [135] J. Nocedal and S. J. Wright, *Sequential quadratic programming*. Springer, 2006.
- [136] M. Norouzi and D. M. Blei, “Minimal loss hashing for compact binary codes,” in *International Conference on Machine Learning*. Citeseer, 2011, pp. 353–360.
- [137] M. Norouzi, D. J. Fleet, and R. R. Salakhutdinov, “Hamming distance metric learning,” in *Advances in Neural Information Processing Systems*, 2012, pp. 1061–1069.
- [138] T. Ojala, M. Pietikainen, and T. Maenpaa, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 971–987, 2002.
- [139] J. Oostveen, T. Kalker, and J. Haitsma, “Feature extraction and a database strategy for video fingerprinting,” in *International Conference on Advances in Visual Information Systems*. Springer, 2002, pp. 117–128.
- [140] P. Over, J. Fiscus, G. Sanders, D. Joy, M. Michel, G. Awad, A. Smeaton, W. Kraaij, and G. Quénot, “Trecvid 2014—an overview of the goals, tasks, data, evaluation mechanisms and metrics,” in *Proceedings of TREC Video Retrieval Evaluation (TRECVID)*, 2014, pp. 1–58.
- [141] H. Peng, J. He, S. Chen, Y. Wang, and Y. Qiao, “Dual-supervised attention network for deep cross-modal hashing,” *Pattern Recognition Letters*, vol. 128, pp. 333–339, 2019.

- [142] Y. Peng, Y. Zhao, and J. Zhang, “Two-stream collaborative learning with spatial-temporal attention for video classification,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 3, pp. 773–786, 2018.
- [143] M. Qian and C. Zhai, “Robust unsupervised feature selection,” in *International Joint Conference on Artificial Intelligence*, 2013, pp. 1621–1627.
- [144] S. Qiao, R. Wang, S. Shan, and X. Chen, “Deep heterogeneous hashing for face video retrieval,” *IEEE Transactions on Image Processing*, vol. 29, pp. 1299–1312, 2019.
- [145] R. Rani, R. Kumar, and A. P. Singh, “Deep learning method based binary descriptor for object detection,” in *Proceedings of International Conference on Emerging Trends in Information Technology (ICETIT)*. Springer, 2020, pp. 364–371.
- [146] N. Rasiwasia, J. Costa Pereira, E. Coviello, G. Doyle, G. R. Lanckriet, R. Levy, and N. Vasconcelos, “A new approach to cross-modal multimedia retrieval,” in *ACM Multimedia*. ACM, 2010, pp. 251–260.
- [147] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, “Xnor-net: Imagenet classification using binary convolutional neural networks,” in *European Conference on Computer Vision*. Springer, 2016, pp. 525–542.
- [148] E. Rosten and T. Drummond, “Machine learning for high-speed corner detection,” in *European Conference on Computer Vision*. Springer, 2006, pp. 430–443.
- [149] N. Roussopoulos, S. Kelley, and F. Vincent, “Nearest neighbor queries,” in *ACM Sigmod Record*, vol. 24, no. 2. ACM, 1995, pp. 71–79.
- [150] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: An efficient alternative to sift or surf,” in *International Conference on Computer Vision*. IEEE, 2011, pp. 2564–2571.
- [151] H. Sak, A. Senior, and F. Beaufays, “Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition,” *arXiv preprint arXiv:1402.1128*, 2014.
- [152] R. Salakhutdinov and G. Hinton, “Semantic hashing,” *International Journal of Approximate Reasoning*, vol. 50, no. 7, pp. 969–978, 2009.

- [153] F. Shen, C. Shen, W. Liu, and H. Tao Shen, “Supervised discrete hashing,” in *Computer Vision and Pattern Recognition*, 2015, pp. 37–45.
- [154] F. Shen, C. Shen, Q. Shi, A. Van Den Hengel, and Z. Tang, “Inductive hashing on manifolds,” in *Computer Vision and Pattern Recognition*, 2013, pp. 1562–1569.
- [155] F. Shen, Y. Xu, L. Liu, Y. Yang, Z. Huang, and H. T. Shen, “Unsupervised deep hashing with similarity-adaptive and discrete optimization,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, pp. 3034–3044, 2018.
- [156] F. Shen, X. Zhou, Y. Yang, J. Song, H. T. Shen, and D. Tao, “A fast optimization method for general binary code learning,” *IEEE Transactions on Image Processing*, vol. 25, no. 12, pp. 5610–5621, 2016.
- [157] L. Shen, R. Hong, H. Zhang, X. Tian, and M. Wang, “Video retrieval with similarity-preserving deep temporal hashing,” *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 15, no. 4, pp. 1–16, 2019.
- [158] Y. Shen, L. Liu, L. Shao, and J. Song, “Deep binaries: Encoding semantic-rich cues for efficient textual-visual cross retrieval,” in *International Conference on Computer Vision*, 2017, pp. 4097–4106.
- [159] Y. Shi, X. You, F. Zheng, S. Wang, and Q. Peng, “Equally-guided discriminative hashing for cross-modal retrieval,” in *International Joint Conference on Artificial Intelligence*, 2019, pp. 4767–4773.
- [160] R. Shinde, A. Goel, P. Gupta, and D. Dutta, “Similarity search and locality sensitive hashing using ternary content addressable memories,” in *Proceedings of the ACM SIGMOD International Conference on Management of Data*. ACM, 2010, pp. 375–386.
- [161] H. Shu, W. Jiang, and R. Yu, “Study on weak bit in vote count and its application in k-nearest neighbors algorithm,” in *IEEE Conference on Industrial Electronics and Applications (ICIEA)*. IEEE, 2015, pp. 119–122.
- [162] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, “Discriminative learning of deep convolutional feature point descriptors,” in *International Conference on Computer Vision*, 2015, pp. 118–126.

- [163] K. Simonyan and A. Zisserman, “Two-stream convolutional networks for action recognition in videos,” in *Advances in Neural Information Processing Systems*, 2014, pp. 568–576.
- [164] —, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [165] A. W. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, “Content-based image retrieval at the end of the early years,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 12, pp. 1349–1380, 2000.
- [166] D. Song, W. Liu, R. Ji, D. A. Meyer, and J. R. Smith, “Top rank supervised binary coding for visual search,” in *International Conference on Computer Vision*, 2015, pp. 1922–1930.
- [167] J. Song, Y.-Z. Song, T. Xiang, and T. M. Hospedales, “Fine-grained image retrieval: the text/sketch input dilemma,” in *British Machine Vision Conference*, 2017, pp. 1–12.
- [168] J. Song, Q. Yu, Y.-Z. Song, T. Xiang, and T. M. Hospedales, “Deep spatial-semantic attention for fine-grained sketch-based image retrieval,” in *International Conference on Computer Vision*, 2017, pp. 5551–5560.
- [169] J. Song, T. He, L. Gao, X. Xu, A. Hanjalic, and H. T. Shen, “Unified binary generative adversarial network for image retrieval and compression,” *International Journal of Computer Vision*, pp. 1–22, 2020.
- [170] J. Song, Y. Yang, Y. Yang, Z. Huang, and H. T. Shen, “Inter-media hashing for large-scale retrieval from heterogeneous data sources,” in *Proceedings of the ACM SIGMOD International Conference on Management of Data*. ACM, 2013, pp. 785–796.
- [171] J. Song, Y. Yang, Z. Huang, H. T. Shen, and R. Hong, “Multiple feature hashing for real-time large scale near-duplicate video retrieval,” in *ACM Multimedia*. ACM, 2011, pp. 423–432.
- [172] J. Song, Y. Yang, Z. Huang, H. T. Shen, and J. Luo, “Effective multiple feature hashing for large-scale near-duplicate video retrieval,” *IEEE Transactions on Multimedia*, vol. 15, no. 8, pp. 1997–2008, 2013.

- [173] J. Song, H. Zhang, X. Li, L. Gao, M. Wang, and R. Hong, “Self-supervised video hashing with hierarchical binary auto-encoder,” *IEEE Transactions on Image Processing*, vol. 27, no. 7, pp. 3210–3221, 2018.
- [174] K. Soomro, A. R. Zamir, and M. Shah, “Ucf101: A dataset of 101 human actions classes from videos in the wild,” *arXiv preprint arXiv:1212.0402*, 2012.
- [175] N. Srivastava, E. Mansimov, and R. Salakhudinov, “Unsupervised learning of video representations using lstms,” in *International Conference on Machine Learning*, 2015, pp. 843–852.
- [176] Z. Stejic, Y. Takama, and K. Hirota, “Relevance feedback-based image retrieval interface incorporating region and feature saliency patterns as visualizable image similarity criteria,” *IEEE Transactions on Industrial Electronics*, vol. 50, no. 5, pp. 839–852, 2003.
- [177] C. Strecha, A. Bronstein, M. Bronstein, and P. Fua, “Ldhash: Improved matching with smaller descriptors,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 1, pp. 66–78, 2012.
- [178] S. Su, Z. Zhong, and C. Zhang, “Deep joint-semantics reconstructing hashing for large-scale unsupervised cross-modal retrieval,” in *International Conference on Computer Vision*, 2019, pp. 3027–3035.
- [179] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science and Business Media, 2010.
- [180] J. Tang, K. Wang, and L. Shao, “Supervised matrix factorization hashing for cross-modal retrieval,” *IEEE Transactions on Image Processing*, vol. 25, no. 7, pp. 3157–3166, 2016.
- [181] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li, “The new data and new challenges in multimedia research,” *arXiv preprint arXiv:1503.01817*, vol. 1, no. 8, 2015.
- [182] Y. Tian, B. Fan, and F. Wu, “L2-net: Deep learning of discriminative patch descriptor in euclidean space,” in *Computer Vision and Pattern Recognition*, 2017, pp. 661–669.

- [183] T. Trzcinski, M. Christoudias, P. Fua, and V. Lepetit, “Boosting binary key-point descriptors,” in *Computer Vision and Pattern Recognition*, 2013, pp. 2874–2881.
- [184] T. Trzcinski and V. Lepetit, “Efficient discriminative projections for compact binary descriptors,” in *European Conference on Computer Vision*. Springer, 2012, pp. 228–242.
- [185] A. Turpin and F. Scholer, “User performance versus precision measures for simple search tasks,” in *Proceedings of the Annual International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 2006, pp. 11–18.
- [186] R. Veltkamp, H. Burkhardt, and H.-P. Kriegel, *State-of-the-art in content-based image and video retrieval*. Springer Science and Business Media, 2013, vol. 22.
- [187] A. Wang *et al.*, “An industrial strength audio search algorithm.” in *Ismir*, vol. 2003. Washington, DC, 2003, pp. 7–13.
- [188] D. Wang, X. Gao, X. Wang, L. He, and B. Yuan, “Multimodal discriminative binary embedding for large-scale cross-modal retrieval,” *IEEE Transactions on Image Processing*, vol. 25, no. 10, pp. 4540–4554, 2016.
- [189] D. Wang, Q. Wang, and X. Gao, “Robust and flexible discrete hashing for cross-modal similarity search,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 10, pp. 2703–2715, 2017.
- [190] ———, “Robust and flexible discrete hashing for cross-modal similarity search,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 10, pp. 2703–2715, 2018.
- [191] H. Wang and C. Schmid, “Action recognition with improved trajectories,” in *International Conference on Computer Vision*, 2013, pp. 3551–3558.
- [192] J. Wang, J. Wang, N. Yu, and S. Li, “Order preserving hashing for approximate nearest neighbor search,” in *ACM Multimedia*. ACM, 2013, pp. 133–142.
- [193] J. Wang, H. T. Shen, J. Song, and J. Ji, “Hashing for similarity search: A survey,” *arXiv preprint arXiv:1408.2927*, 2014.

- [194] J. Wang, T. Zhang, N. Sebe, H. T. Shen *et al.*, “A survey on learning to hash,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 769–790, 2017.
- [195] J. Wang, S. Kumar, and S.-F. Chang, “Semi-supervised hashing for scalable image retrieval,” in *Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 3424–3431.
- [196] ———, “Semi-supervised hashing for large-scale search,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 12, pp. 2393–2406, 2012.
- [197] J. Wang, W. Liu, S. Kumar, and S.-F. Chang, “Learning to hash for indexing big data: a survey,” *Proceedings of the IEEE*, vol. 104, no. 1, pp. 34–57, 2016.
- [198] J. Wang, W. Liu, A. X. Sun, and Y.-G. Jiang, “Learning hash codes with listwise supervision,” in *International Conference on Computer Vision*, 2013, pp. 3032–3039.
- [199] K. Wang, Q. Yin, W. Wang, S. Wu, and L. Wang, “A comprehensive survey on cross-modal retrieval,” *arXiv preprint arXiv:1607.06215*, 2016.
- [200] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, “Temporal segment networks: Towards good practices for deep action recognition,” in *European Conference on Computer Vision*. Springer, 2016, pp. 20–36.
- [201] Q. Wang, Z. Yuan, Q. Du, and X. Li, “Getnet: A general end-to-end 2-d cnn framework for hyperspectral image change detection,” *IEEE Transactions on Geoscience and Remote Sensing*, pp. 1–11, 2018.
- [202] Q. Wang, J. Gao, and Y. Yuan, “A joint convolutional neural networks and context transfer for street scenes labeling,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 5, pp. 1457–1470, 2017.
- [203] Q. Wang, J. Wan, and Y. Yuan, “Deep metric learning for crowdedness regression,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 10, pp. 2633–2643, 2017.
- [204] Q. Wang, G. Zhu, and Y. Yuan, “Statistical quantization for similarity search,” *Computer Vision and Image Understanding*, vol. 124, pp. 22–30, 2014.

- [205] T. Wang, L. Zhu, Z. Cheng, J. Li, and Z. Gao, “Unsupervised deep cross-modal hashing with virtual label regression,” *Neurocomputing*, 2019.
- [206] W. Wang, B. C. Ooi, X. Yang, D. Zhang, and Y. Zhuang, “Effective multi-modal retrieval based on stacked auto-encoders,” *VLDB Endowment*, vol. 7, no. 8, pp. 649–660, 2014.
- [207] X. Wang, L. Gao, P. Wang, X. Sun, and X. Liu, “Two-stream 3-d convnet fusion for action recognition in videos with arbitrary size and length,” *IEEE Transactions on Multimedia*, vol. 20, no. 3, pp. 634–644, 2018.
- [208] Y. Wang, X. Nie, Y. Shi, X. Zhou, and Y. Yin, “Attention-based video hashing for large-scale video retrieval,” *IEEE Transactions on Cognitive and Developmental Systems*, 2019.
- [209] X. Wei, Y. Zhang, Y. Gong, and N. Zheng, “Kernelized subspace pooling for deep local descriptors,” in *Computer Vision and Pattern Recognition*, 2018, pp. 1867–1875.
- [210] Y. Wei, Y. Zhao, C. Lu, S. Wei, L. Liu, Z. Zhu, and S. Yan, “Cross-modal retrieval with cnn visual features: A new baseline,” *IEEE Transactions on Cybernetics*, vol. 47, no. 2, pp. 449–460, 2017.
- [211] Y. Weiss, A. Torralba, and R. Fergus, “Spectral hashing,” in *Advances in Neural Information Processing Systems*, 2009, pp. 1753–1760.
- [212] Z. Wen and W. Yin, “A feasible method for optimization with orthogonality constraints,” *Mathematical Programming*, vol. 142, no. 1-2, pp. 397–434, 2013.
- [213] S. Wold, K. Esbensen, and P. Geladi, “Principal component analysis,” *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.
- [214] B. Wu, Q. Yang, W.-S. Zheng, Y. Wang, and J. Wang, “Quantized correlation hashing for fast cross-modal search,” in *International Joint Conference on Artificial Intelligence*, 2015, pp. 3946–3952.
- [215] F. Wu, Z. Yu, Y. Yang, S. Tang, Y. Zhang, and Y. Zhuang, “Sparse multi-modal hashing,” *IEEE Transactions on Multimedia*, vol. 16, no. 2, pp. 427–439, 2013.

- [216] G. Wu, Z. Lin, J. Han, L. Liu, G. Ding, B. Zhang, and J. Shen, “Unsupervised deep hashing via binary latent factor models for large-scale cross-modal retrieval.” in *International Joint Conference on Artificial Intelligence*, 2018, pp. 2854–2860.
- [217] G. Wu, L. Liu, Y. Guo, G. Ding, J. Han, J. Shen, and L. Shao, “Unsupervised deep video hashing with balanced rotation.” *International Joint Conference on Artificial Intelligence*, 2017, pp. 3076–3082.
- [218] L. Wu, Y. Wang, and L. Shao, “Cycle-consistent deep generative hashing for cross-modal retrieval,” *IEEE Transactions on Image Processing*, vol. 28, no. 4, pp. 1602–1612, 2018.
- [219] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, “Supervised hashing for image retrieval via image representation learning.” in *AAAI Conference on Artificial Intelligence*, 2014, pp. 2156–2162.
- [220] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, “Sun database: Large-scale scene recognition from abbey to zoo,” in *Computer Vision and Pattern Recognition*, 2010, pp. 3485–3492.
- [221] B. Xu, J. Bu, Y. Lin, C. Chen, X. He, and D. Cai, “Harmonious hashing.” in *International Joint Conference on Artificial Intelligence*, 2013, pp. 1820–1826.
- [222] R. Xu, C. Li, J. Yan, C. Deng, and X. Liu, “Graph convolutional network hashing for cross-modal retrieval,” in *International Joint Conference on Artificial Intelligence*, 2019, pp. 10–16.
- [223] X. Xu, F. Shen, Y. Yang, and H. T. Shen, “Discriminant cross-modal hashing,” in *Proceedings of the ACM on International Conference on Multimedia Retrieval*. ACM, 2016, pp. 305–308.
- [224] C. Yan, X. Bai, S. Wang, J. Zhou, and E. R. Hancock, “Cross-modal hashing with semantic deep embedding,” *Neurocomputing*, vol. 337, pp. 58–66, 2019.
- [225] C. Yan, H. Xie, S. Liu, J. Yin, Y. Zhang, and Q. Dai, “Effective uyghur language text detection in complex background images for traffic prompt identification,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 220–229, 2018.

- [226] C. Yan, H. Xie, D. Yang, J. Yin, Y. Zhang, and Q. Dai, “Supervised hash coding with deep neural network for environment perception of intelligent vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 284–295, 2018.
- [227] H. Yang, C. Huang, F. Wang, K. Song, and Z. Yin, “Robust semantic template matching using a superpixel region binary descriptor,” *IEEE Transactions on Image Processing*, vol. 28, no. 6, pp. 3061–3074, 2019.
- [228] R. Yang, “New results on some quadratic programming problems,” Ph.D. dissertation, University of Illinois at Urbana-Champaign, 2013.
- [229] Y. Yang, H. T. Shen, Z. Ma, Z. Huang, and X. Zhou, “L2, 1-norm regularized discriminative feature selection for unsupervised,” in *International Joint Conference on Artificial Intelligence*, 2011, pp. 1589–1594.
- [230] G. Ye, D. Liu, J. Wang, and S.-F. Chang, “Large-scale video hashing via structure learning,” in *International Conference on Computer Vision*, 2013, pp. 2272–2279.
- [231] J. Ye, S. Zhang, T. Huang, and Y. Rui, “Cdbin: Compact discriminative binary descriptor learned with efficient neural network,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 3, pp. 862–874.
- [232] P. K. R. Yelampalli, J. Nayak, and V. H. Gaidhane, “A novel binary feature descriptor to discriminate normal and abnormal chest ct images using dissimilarity measures,” *Pattern Analysis and Applications*, pp. 1–10, 2019.
- [233] X. Yu, Y. Tian, F. Porikli, R. Hartley, H. Li, H. Heijnen, and V. Balntas, “Unsupervised extraction of local image descriptors via relative distance ranking loss,” in *International Conference on Computer Vision Workshops*, 2019.
- [234] Z. Yu, F. Wu, Y. Yang, Q. Tian, J. Luo, and Y. Zhuang, “Discriminative coupled dictionary hashing for fast cross-media retrieval,” in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2014, pp. 395–404.
- [235] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, “Beyond short snippets: Deep networks for video classification,” in *Computer Vision and Pattern Recognition*, 2015, pp. 4694–4702.

- [236] C. Zach, T. Pock, and H. Bischof, “A duality based approach for realtime tv-l 1 optical flow,” *Pattern Recognition*, pp. 214–223, 2007.
- [237] S. Zagoruyko and N. Komodakis, “Learning to compare image patches via convolutional neural networks,” in *Computer Vision and Pattern Recognition*, 2015, pp. 4353–4361.
- [238] D. Zhang, J. Wang, D. Cai, and J. Lu, “Self-taught hashing for fast similarity search,” in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2010, pp. 18–25.
- [239] D. Zhang and W.-J. Li, “Large-scale supervised multimodal hashing with semantic correlation maximization,” in *AAAI Conference on Artificial Intelligence*, 2014, pp. 2177–2183.
- [240] H. Zhang, M. Wang, R. Hong, and T.-S. Chua, “Play and rewind: Optimizing binary representations of videos by self-supervised temporal hashing,” in *ACM Multimedia*. ACM, 2016, pp. 781–790.
- [241] H. Zhang, L. Liu, Y. Long, and L. Shao, “Unsupervised deep hashing with pseudo labels for scalable image retrieval,” *IEEE Transactions on Image Processing*, vol. 27, no. 4, pp. 1626–1638, 2018.
- [242] J. Zhang and Y. Peng, “Multi-pathway generative adversarial hashing for unsupervised cross-modal retrieval,” *IEEE Transactions on Multimedia*, vol. 22, no. 1, pp. 174–187, 2019.
- [243] K. Zhang, W.-L. Chao, F. Sha, and K. Grauman, “Video summarization with long short-term memory,” in *European Conference on Computer Vision*. Springer, 2016, pp. 766–782.
- [244] P. Zhang, W. Zhang, W.-J. Li, and M. Guo, “Supervised hashing with latent factor models,” in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2014, pp. 173–182.
- [245] Q. Zhang, Y. Liu, R. Blum, J. Han, and D. Tao, “Sparse representation based multi-sensor image fusion for multi-focus and multi-modality images: A review,” *Information Fusion*, vol. 40, pp. 57–75, 2018.

- [246] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang, “Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification,” *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 4766–4779, 2015.
- [247] Z. Zhang, L. Liu, F. Shen, H. T. Shen, and L. Shao, “Binary multi-view clustering,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 7, pp. 1774–1782, 2018.
- [248] F. Zhao, Y. Huang, L. Wang, and T. Tan, “Deep semantic ranking based hashing for multi-label image retrieval,” in *Computer Vision and Pattern Recognition*, 2015, pp. 1556–1564.
- [249] J. Zhou, G. Ding, and Y. Guo, “Latent semantic sparse hashing for cross-modal similarity search,” in *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2014, pp. 415–424.
- [250] F. Zhu, X. Kong, L. Zheng, H. Fu, and Q. Tian, “Part-based deep hashing for large-scale person re-identification,” *IEEE Transactions on Image Processing*, vol. 26, no. 10, pp. 4806–4817, 2017.
- [251] X. Zhu, Z. Huang, H. T. Shen, and X. Zhao, “Linear cross-modal hashing for efficient multimedia search,” in *ACM Multimedia*. ACM, 2013, pp. 143–152.
- [252] M. Zieba, P. Semberecki, T. El-Gaaly, and T. Trzcinski, “Bingan: Learning compact binary descriptors with a regularized gan,” in *Advances in Neural Information Processing Systems*, 2018, pp. 3608–3618.