Compact feature learning for multimedia retrieval

Liong, Venice Erin Baylon

2019

Liong, V. E. B. (2019). Compact feature learning for multimedia retrieval. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/105335>

<https://doi.org/10.32657/10220/48655>

Compact Feature Learning for Multimedia Retrieval

Venice Erin Baylon Liong

**Interdisciplinary Graduate School
Rapid-Rich Object SEarch (ROSE) Lab**

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2019

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

June 5, 2019

Date

Venice Erin Baylon Liong

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

6/6/19

Date

Prof. Tan Yap-Peng

Authorship Attribution Statement

This thesis contains material from 6 papers published in the following peer-reviewed journals / from papers accepted at conferences in which I am listed as an author.

- (i) Chapter 3 is published in “Liong, V.E., Lu, J., Tan, Y.P. and Zhou, J., 2016. *Deep coupled metric learning for cross-modal matching*. IEEE Transactions on Multimedia, 19(6), pp.1234-1244.”
- (ii) Chapter 4 is published in “Liong, V.E., Lu, J. and Tan, Y.P., 2018. *Cross-modal discrete hashing*. Pattern Recognition, 79, pp.114-129.” and “Liong, V.E., Lu, J. and Tan, Y.P., 2017, September. *Learning a cross-modal hashing network for multimedia search*. In 2017 IEEE International Conference on Image Processing (ICIP) (pp. 3700-3704). IEEE.”
- (iii) Chapter 5 is published in “Liong, V.E., Lu, J., Duan, L.Y. and Tan, Y., 2018. *Deep Variational and Structural Hashing*. IEEE transactions on pattern analysis and machine intelligence. doi: 10.1109/TPAMI.2018.2882816” and “Liong, V.E., Lu, J., Tan, Y.P. and Zhou, J., 2017, October. *Cross-modal deep variational hashing*. In 2017 IEEE International Conference on Computer Vision (ICCV) (pp. 4097-4105). IEEE.”
- (iv) Chapter 6 is published in “Liong, V.E., Lu, J., Tan, Y.P. and Zhou, J., 2016. *Deep video hashing*. IEEE Transactions on Multimedia, 19(6), pp.1209-1219.”

The contributions of the co-authors from the above publications are as follows:

- (i) Constant guidance on implementation details and experiment set-up.
- (ii) Proofreading of manuscripts.

June 5, 2019

Date

Venice Erin Baylon Liong

Acknowledgments

I would like to thank my supervisor, Professor Yap-Peng Tan, for his constant support, proper guidance and inspiring discussions. I have learned a lot from our meetings and I am driven to work harder to accomplish my goals as a researcher. Also to my co-supervisor and mentors, Dr. Lin Weisi and Dr. Junsong Yuan/Dr. Jiang Xudong for their support and guidance on my thesis. I would also like to thank Dr. Jiwen Lu for the interesting and relevant discussions in our research field and giving proper advice in several implementation details, technical writing, and research directions. I want to thank my colleagues, particularly Dr. Junlin Hu and Raman, for being approachable while I work on my thesis. I also thank the people in the Rapid-Rich Object Search (ROSE) Laboratory for providing me with a great environment so that I could perform my work easily and comfortably. I would also like to acknowledge the Interdisciplinary Graduate School (IGS), the ROSE Lab, and Nanyang Technological University, for granting me the NTU Research Scholarship to complete my Ph.D. study.

Lastly, I would like to thank my family and friends for being the best support system for many years during this period.

Contents

Acknowledgments	i
Summary	vii
List of Figures	x
List of Tables	xiii
1 Introduction	1
1.1 Compact Feature Learning for Nearest Neighbor Search	4
1.2 Multimedia Search	9
1.3 Motivation	11
1.4 Overview of Contributions and Thesis Organization	13
2 Literature Review	16
2.1 Single-Modal Search	16
2.1.1 Shallow Hashing methods	16
2.1.2 Deep Hashing methods	20
2.2 Cross-Modal Search	23
2.2.1 Cross-Modal Matching methods	23
2.2.2 Shallow Cross-Modal Hashing methods	24
2.2.3 Deep Cross-Modal Hashing methods	27
2.3 Video Search	30

2.4	Deep Learning	31
2.4.1	Deep Metric Learning	32
2.4.2	Supervised Deep learning for Video Analysis	32
2.4.3	Deep Generative Models	33
2.5	Chapter Summary	34
3	Deep Coupled Metric Learning	35
3.1	Introduction	35
3.2	Deep Coupled Metric Learning	37
3.2.1	Coupled Metric Learning	37
3.2.2	Deep Coupled Metric Learning	38
3.2.3	Implementation Details	44
3.3	Experiments	44
3.3.1	Datasets and Experimental Set-up	45
3.3.2	Experimental Results	47
3.3.3	Further Analysis	59
3.3.4	Discussion	60
3.4	Chapter Summary	62
4	Cross-Modal Discrete Hashing	63
4.1	Introduction	64
4.2	Cross-Modal Discrete Hashing	65
4.2.1	Shallow CMDH	65
4.2.2	Shallow CMDH extensions	73
4.2.3	Deep CMDH	74

4.3	Experiments	80
4.3.1	Datasets and Experimental Set-up	80
4.3.2	Experimental Results for Shallow CMDH	83
4.3.3	Experimental Results for Deep CMDH	97
4.3.4	Discussion	101
4.4	Chapter Summary	102
5	Deep Variational and Structural Hashing	103
5.1	Introduction	104
5.2	Deep Variational and Structural Hashing	107
5.3	Cross-Modal Deep Variational and Structural Hashing (CM-DVStH)	114
5.3.1	Cross-Modal Fusion Network	116
5.3.2	Modality-Specific Networks	117
5.4	Experiments	120
5.4.1	Datasets and Experimental Set-up	120
5.4.2	Experimental Results	125
5.4.3	Further Analysis	134
5.4.4	Discussion	140
5.5	Chapter Summary	141
6	Deep Video Hashing	143
6.1	Introduction	143
6.2	Deep Video Hashing	145
6.2.1	Learning-based Video Hashing	145
6.2.2	Deep Video Hashing	146

6.2.3	Implementation Details	150
6.3	Experiments	151
6.3.1	Datasets and Experimental Set-up	152
6.3.2	Experimental Results	154
6.3.3	Further Analysis	159
6.3.4	Discussion	160
6.4	Chapter Summary	160
7	Conclusion and Future Work	168
7.1	Conclusion	168
7.2	Future Work	171
	Author's Publications	175
	Bibliography	177

Summary

Efficient multimedia search has attracted much attention in recent years due to the exponential growth of data available. Extracting representative compact features for nearest neighbor (NN) search and approximate nearest neighbor (ANN) search has played an important but challenging role in multimedia retrieval systems. This is because efficient compact features require two essential properties: (1) It should be able to capture high quality information from raw data; (2) It needs to be of small dimensionality to support fast search with low memory costs. Motivated by this, this thesis is dedicated to developing several compact feature learning algorithms using machine learning tools applied in different multimedia search systems or combined into a complete multimedia search system for any type of visual data. While several approaches have been proposed in the literature, a number of issues still need to be addressed. We present the limitations of existing work and propose our corresponding solutions below.

First, most compact feature learning methods for cross-modal retrieval perform shallow linear/nonlinear projection from high-dimensional features to reduce the modality gap between different data modalities. However, this may not be representative enough to capture the high-level semantics from nonlinear data. Hence, we propose a deep coupled metric learning (DCML) method to perform nonlinear hierarchical transformations trained using a discriminative loss to simultaneously reduce the modality gap and learn the compact features.

Second, most compact binary code learning (also coined as *hashing*) methods for cross-modality retrieval relax the binary constraint during training which leads to accumulated approximation loss that may limit the information exploited during learning. To address this, we propose a cross-modal discrete hashing (CMDH) method that performs discrete optimization to learn a unified binary code and learn discriminative hashing functions for each modality alternatively. We propose the CMDH framework in both shallow and deep frameworks.

Third, most deep hashing methods perform training using pairwise and triplet-wise loss functions. This may not exploit the label information fully as the learning process would depend largely on generating random pairs and triplets during training. Moreover, most deep hashing models use a series of convolution, pooling and fully connected-layers to obtain discriminative features; this may not be general enough as it is prone to biases from the training set. Hence, we propose a deep variational and structural hashing method for single-modality and cross-modality retrieval (DVStH and CM-DVStH). Our proposed approach learns a deep hashing method with a probabilistic framework and a structured hash layer to obtain more general and discriminative codes. In addition, the network is trained using a point-wise classification loss for better learning.

Lastly, compact binary code learning for video retrieval has not been heavily explored and most only perform frame sampling and use conventional hashing methods to learning the binary codes. Exploiting the temporal information and learn discriminative binary codes are much desirable. Hence, we propose a deep video hashing (DVH) method that performs binary code learning using a deep framework with frame pooling to exploit the spatio-temporal information from videos trained under discriminative and quantization losses.

Experimental results in benchmark retrieval datasets and comparisons with popular feature learning and hashing methods have demonstrated the effectiveness of our proposed methods.

List of Figures

1.1	General procedure of a content-based retrieval system	2
1.2	Binary code learning in linear and nonlinear form	6
1.3	Computational complexity of hamming distance	7
1.4	Approximate Nearest Neighbor Search in the Hamming space.	8
1.5	Cross-Modal Retrieval Problem.	9
1.6	Video Search Problem.	11
3.1	Basic idea of the proposed DCML method	36
3.2	Basic idea of the large-margin loss of our proposed DCML method.	39
3.3	Sample images from the CASIA VIS-NIR (version 2.0) dataset.	47
3.4	Precision-scope curve of different coupled metric learning methods from the Wiki experiment using SIFT features	48
3.5	Sample results by DCML method for text-to-image and image-to-text in the Wiki dataset	50
3.6	Precision-scope curve of different coupled metric learning methods from the PASCAL VOC 2007 using BOVW-GIST-COLOR image features.	52
3.7	Sample results by DCML method for text-to-image and image-to-text in the Pascal VOC 2007 dataset	53

3.8	Performance curves of different coupled metric learning methods on the CASIA VIS-NIR (version 2) dataset	55
3.9	Convergence curve of DCML on the CASIA VIS-NIR (version 2) dataset. .	60
4.1	Basic idea of our proposed shallow CMDH approach	66
4.2	Basic idea of our proposed deep CMDH approach.	76
4.3	Precision-Recall (PR) curves and Precision-N curves on the Wiki dataset for Image-to-Text and Text-to-Image tasks	86
4.4	Precision-Recall (PR) curves and Precision-N curves on the MIRFlickr dataset for Image-to-Text and Text-to-Image tasks.	88
4.5	Precision-Recall (PR) curves and Precision-N curves on the NUS-WIDE dataset for Image-to-Text and Text-to-Image tasks	90
4.6	mAP performance of our CMDH versus different values of η	96
4.7	Training time of different cross-modal hashing methods	97
4.8	Objective function values of our CMDH-kernel and CMDH-linear on the NUS-WIDE dataset.	99
4.9	NDCG performance of different cross-modal hashing methods for the IAPRTC12 and NUSWIDE database.	100
5.1	Different hashing network architectures	104
5.2	Bottleneck layer length performance analysis	106
5.3	Basic idea of our DVStH method for image retrieval	107
5.4	Visual representation of how the discriminative loss for multi-label target data is obtained	111
5.5	Binary code encoding in DVStH	114

5.6	Basic idea of our CM-DVStH for cross-modality multimedia retrieval. . . .	115
5.7	NDCG and ACG performance of different cross-modal hashing methods . .	131
5.8	mAP performance of different variants of our DVStH	134
5.9	mAP performance of varying values of η of DVStH	135
5.10	Mean variance of hashing methods in the NUS-WIDE	138
5.11	The encoding time for each image on the ImageNet database. The black line denotes the feature extraction time of the base network.	140
5.12	The retrieval time of a query sample on a database.	140
6.1	Basic idea of our DVH approach for video retrieval	162
6.2	Different DVH architectures with frame fusion	163
6.3	Extracting binary codes for a single video from DVH	164
6.4	Sample videos for the Columbia Consumer Video (CCV) database.	164
6.5	Sample videos for the Joint-annotated HMDB (JHMDB) database.	164
6.6	Deep baseline architectures based on frame-by-frame training for video hashing	165
6.7	DVH Retrieval time on the CCV and JHDMB dataset	165
6.8	Precision-Recall (PR) curves on the CCV dataset	166
6.9	Precision-N curves on the CCV dataset	166
6.10	Precision-Recall (PR) curves on the JHMDB dataset	166
6.11	Precision-N curves on the JHMDB dataset	167
6.12	mAP performance of our DVH method at varying λ	167

List of Tables

2.1	General Overview of Single-Modal Hashing Methods.	23
2.2	General Overview of Cross-Modal Search Methods.	30
2.3	General Overview of Video Hashing Methods.	31
3.1	mAP (%) performance comparison of our DCML and state-of-the-art cross-modal matching methods on the Wiki dataset using SIFT features.	49
3.2	mAP (%) performance comparison of our DCML and state-of-the-art cross-modal matching methods on the Wiki dataset using CNN features.	49
3.3	mAP (%) performance comparison of our DCML and state-of-the-art cross-modal matching methods on the PASCAL VOC 2007 dataset using hand-crafted image features.	51
3.4	mAP (%) performance comparison of our DCML and state-of-the-art cross-modal matching methods on the PASCAL VOC 2007 dataset using CNN features.	52
3.5	mAP (%) and top 20% performance on the NUS-WIDE dataset.	55
3.6	Sample Tag Annotations obtained from the NUS-WIDE Dataset.	56
3.7	Recognition performance comparison of our DCML method and state-of-the-art coupled metric learning methods on the CASIA VIS-NIR (version 2.0) dataset.	57

3.8	Rank-one Recognition rate comparison of our DCML method and state-of-the-art heterogeneous face recognition methods on the CASIA VIS-NIR (version 2.0) dataset.	58
3.9	Rank-one recognition rate on the CASIA VIS-NIR (version 2.0) dataset and average mAP using variants of our DCML method	59
3.10	Computational time (seconds) used by different coupled metric learning methods on the CASIA VIS-NIR (version 2) dataset.	61
4.1	mAP performance comparison of our shallow CMDH methods and different cross-modal hashing methods, where images were used as query	84
4.2	mAP performance comparison of our shallow CMDH methods and different cross-modal hashing methods, where texts/tags were used as query . . .	85
4.3	mAP performance comparison of our shallow CMDH methods and different cross-modal hashing methods on different datasets where shared binary codes were used as gallery samples.	87
4.4	mAP performance comparison of our MMDH and other multi-modal hashing method on the Wiki dataset where shared binary codes were used as gallery samples.	91
4.5	mAP performance comparison of our MMDH and other multi-modal hashing method on the NUS-WIDE dataset where out-of-sample binary codes were used as gallery samples.	92
4.6	Average mAP performance of our CMDH on Wiki and MIRFlickr datasets when different learning strategies were employed.	95
4.7	Computational time of different cross-modal hashing methods	97

4.8	mAP performance of our deep CMDH and different cross-modal hashing methods on the IAPRTC12 dataset.	98
4.9	mAP performance comparison of our deep CMDH and different cross-modal hashing methods on the NUS-WIDE dataset.	98
5.1	mAP performance comparison of our DVStH and different deep hashing models for the single-domain category retrieval experiment on the CIFAR10 dataset	126
5.2	mAP performance comparison of our DVStH and other quantization/hashing-based models for the cross-domain category retrieval experiment.	127
5.3	mAP performance comparison of our DVStH and deep hashing methods for the multi-label single-modality retrieval experiment.	128
5.4	NDCG@100 and ACG@100 performance comparison of our DVStH and different deep hashing methods on the NUS-WIDE dataset.	128
5.5	mAP performance comparison of our CM-DVStH and different cross-modal hashing methods.	130
5.6	mAP performance comparison of our CM-DVStH and different deep cross-modal hashing methods on the cross-modal retrieval experiments.	132
5.7	mAP performance of different variants of our CM-DVStH method on the MIRFLICKR25k and IAPRTC12 dataset.	137
6.1	DVH architectural implementations in the experiments.	152
6.2	Performance results comparison of our DVH with the baseline deep architecture on the CCV dataset.	155
6.3	Performance results comparison of our DVH with the baseline deep architecture on the JHMDB dataset.	155

6.4	Performance results comparison of our DVH and different learning-based hashing methods on the CCV dataset.	156
6.5	Performance results comparison of our DVH and different learning-based hashing methods on the JHDMB dataset.	156
6.6	Hamming ranking (mAP, %) results in comparison with other video hashing algorithms on the CCV dataset.	158
6.7	Hamming ranking (mAP, %) results of our DVH with different values of p and <i>stride</i> on the CCV dataset.	158
6.8	Hamming ranking (mAP, %) results of our DVH in different values of p and <i>stride</i> on the JHDMB dataset.	159

Chapter 1

Introduction

Multimedia data in the form of images, text, and videos have grown rapidly due to the emergence of web technologies in this digital age. Particularly, the photo-sharing website, Flickr is reported to have more than one million photos shared per day,¹ while the video-sharing website, YouTube, has 300 hours of videos uploaded every minute². With the large amount of data available, efficient large-scale content-based retrieval has a critical role more than ever for search engines and multimedia systems.

We present Figure 1.1 to illustrate the general procedure for content-based retrieval. While our scope is generally for multi-modal data, we use image data as an example for simplicity. Given a query sample, a content-based retrieval system aims to retrieve the most relevant semantic content from a large gallery set accurately and efficiently. It generally contains three main parts: *Feature extraction* extracts representative features from raw multimedia data. *Encoding/Indexing* encodes and/or indexes the features into compact representations suitable for retrieval. *Retrieval* selects and outputs the most relevant data samples from the gallery based on the encoding. The retrieval system comprises of an off-line and on-line phase. The former is where the **gallery** set features are pre-computed and

¹<https://techcrunch.com/2014/02/10/flickr-at-10-1m-photos-shared-per-day-170-increase-since-making-1tb-free/>

²<https://fortunelords.com/youtube-statistics/>

Figure 1.1: General procedure of a content-based retrieval system with images as the modality of the data samples. In our work, we focus on the feature extraction and encoding through compact feature learning.

indexes are stored, and the latter is where **query-dependent** computations are performed. The efficiency of a retrieval system usually depends more on the computational complexity of the on-line phase and should be real-time as much as possible.

The naive approach for content-based retrieval is the exhaustive Nearest Neighbor (NN) Search. Here, we first perform feature extraction, then compute the similarity between a query and all gallery samples using a distance function, and finally retrieve the gallery samples having the smallest distances. However, this is computationally intensive when the gallery set is large and the features are high-dimensional. Hence, Approximate Nearest Neighbor (ANN) search techniques have been explored where the nearest neighbors are selected based on probabilities. Representative methods for ANN search can be categorized into Tree-based [1–3], Quantization-based [4–6] and Hashing-based methods [7, 8].

Tree-based techniques such as KD-Trees [1] and Randomized KD-Trees [3] encode the data samples through hierarchically subdividing the feature space for faster search. However, it has been known that the performance of this approach degrades when dealing with high-dimensional data [2].

Quantization-based methods such as Product Quantization [4] involve dividing the feature space into several disjoint subspaces or bins. Each subspace is further partitioned into several clusters. Each data point is annotated with an index for each subspace based on the distance of the samples to a cluster centroid. From this, the similarity of a query and gallery sample is the summation of the distances of the indexed cluster centers for all subspaces. This method has presented high retrieval accuracy due to strong representation power and the potential for fast search by using a pre-computed look-up table. However, the existence of cluster centers for computing the similarity may still be a bottleneck in some applications speed and memory-wise especially if data are of high-dimension.

Lastly, Hashing-based methods aim to encode high-dimensional real-valued data into compact binary codes while retaining much semantic content as possible. Given a hashing algorithm which encodes data points into binary codes with K -bits, a feature space would be partitioned into 2^K parts coined as *hash buckets*. During retrieval, the nearest neighbor is selected if the gallery binary code falls into the same or nearby bucket as the query code. This encoding and indexing mechanism allows efficient retrieval as it provides faster search speed due to simple bit-wise operation for Hamming distance computation and great memory storage due to bit size compact codes.

From these search frameworks, it is easy to see that extracting efficient compact features is integral in achieving accurate search in large-scale databases. Hence, how to represent raw data to compact features such that distinct semantics are captured and be suitable for similarity search in the feature space is much desirable. However, extracting compact features present many challenges since multimedia data, such as images and videos, consists of low-level pixels and are high dimensional. This presents a semantic gap between raw data seen by machines and high-level concepts grasped by the human brain. *Compact feature learning* can bridge this gap through machine learning techniques which use

statistical inference on data to learn complex feature models. While feature engineering based on prior domain knowledge (also named as *hand-crafted features*) have been widely used in multimedia, it may not be representative in certain tasks and would be troublesome to engineer for each task. Differently, feature learning can directly learn from data available. Furthermore, such *hand-crafted features* can be enhanced through feature learning to achieve task-specific compact features.

Hence, in this thesis, we focus on compact feature learning for large-scale multimedia retrieval using machine learning techniques. Particularly, we perform accurate and efficient multimedia search through real-valued (Chapter 3) and binary code (Chapters 4, 5, and 6) compact feature learning; using metric learning (Chapters 3 and 6) and deep learning techniques (All Chapters). Our proposed methods address different forms of retrieval (single-modality, cross-modality, and video) but can be combined and used as a **rich multimedia information retrieval system** such that a user can choose to exploit different modalities of data, be it image, text or video, to search for information from a large-scale multi-modal database. This is useful for applications that have databases which rely on content to extract similar data of relevant information as opposed to current search systems which only rely on text or tags for queries.

In the following subsections, we provide an introduction to compact feature learning for nearest neighbor search, examine existing challenges on multimedia search, and discuss the motivation, contributions, and organization of this thesis.

1.1 Compact Feature Learning for Nearest Neighbor Search

Finding the nearest neighbor (NN) of a query data sample in a high-dimensional space for a single modality is formulated here. Given a query feature vector $\mathbf{x}_q \in \mathbb{R}^d$ and a gallery

feature set $\mathcal{X} \subset \mathbb{R}^d$, Nearest Neighbor search aims to identify the elements, $NN(\mathbf{x}_q) \in \mathcal{X}$, which is closest to $\mathbf{x}_q$ based on a certain distance metric, $d(\cdot)$:

$$NN(\mathbf{x}_q) = \arg \min_{\mathbf{x} \in \mathcal{X}} d(\mathbf{x}_q, \mathbf{x}) \quad (1.1)$$

where $d(\cdot)$ is commonly the Euclidean distance. This exhaustive search would require computing exact distances between all pairs of vectors to determine the closest samples. This requires a linear search time of $O(Nd)$, where N is the number of samples in $\mathcal{X}$, which is a bottleneck for many retrieval systems. Hence a suitable feature extractor, f , that maps high-dimensional multimedia data into a compact feature vector is much desired:

$$f : \mathbb{R}^d \rightarrow \mathbb{R}^p \quad \text{where} \quad d \gg p, \quad (1.2)$$

An approach for learning compact features would be projection-based where we map the high-dimensional feature linearly or nonlinearly through a projection matrix/matrices learned from statistical inference. The most basic implementation is to use a linear projection matrix $\mathbf{W} \in \mathbb{R}^{d \times p}$:

$$\tilde{\mathbf{x}} = f(\mathbf{x}) = \mathbf{W}^\top \mathbf{x} \quad (1.3)$$

where $\tilde{\mathbf{x}} \in \mathbb{R}^p$ is the compact feature vector. We can also learn compact binary codes suitable for hashing-based search in the hamming space. Binary code learning or *hashing*, aims to transform high-dimensional features such that compact binary codes are extracted as follows:

$$h : \mathbb{R}^d \rightarrow \{0, 1\}^K, \quad (1.4)$$

Similarly, the most basic way to learn compact binary codes, $\mathbf{b} \in \{0, 1\}^K$, is to learn a linear projection matrix to reduce the dimension of the feature and then perform hard

Figure 1.2: Binary code learning in a linear and nonlinear form. Given some data distribution where a color represents a class, a nonlinear model can better discriminate the data as compared to a linear model.

thresholding using a $\text{sgn}(\cdot)$ function:

$$\mathbf{b} = h_{lin}(\mathbf{x}) = \text{sgn}(\mathbf{W}^\top \mathbf{x}) \quad (1.5)$$

where $\text{sgn}(v)$ returns 1 if $v > 0$ and 0 otherwise. To be more representative, a feature can also be mapped in a nonlinear manner as follows:

$$\mathbf{b} = h_{nonlin}(\mathbf{x}) = \text{sgn}(\phi(\mathbf{x})) \quad (1.6)$$

where $\phi(\cdot) \in \mathbb{R}^K$ is a nonlinear function. This way, the hashing function can adapt well with the nonlinearities of real-world data as shown in Figure 1.2. Nonlinear functions can be in the form of kernels, boosting models, or neural networks.

While real-valued compact features are more representative, binary compact features present advantages for search systems. Compact binary codes provide fast distance approximation and sorting speed due to simple bit-wise XOR operation for Hamming distance computation and great memory storage due to bit size compact codes [9]. Figure 1.3(a) shows the difference between the computation speed using Hamming distance for binary

Figure 1.3: Computational complexity of using hamming distance when retrieving from 1 million samples given a query vector. In (b), an empty retrieval time means it is not applicable as the look-up table would be too large.

codes, and cosine/euclidean distance for real-valued codes of similar length. As can be seen, the Hamming distance provides faster computation most notably at larger bits.

Because exhaustive search leads to a linear search time, it may still be costly for applications using a large dataset even with using compact features. Hence, Approximate Nearest Neighbor (ANN) methods are preferred.

ANN search via hashing is through *Hash Table Lookup* such that multiple hash buckets are populated off-line with gallery points using their pre-computed compact binary codes. The gallery points are assigned with binary codes or hash keys for indexing it to its respective hash bucket. The retrieval scheme using Hash table lookup is represented in Figure 1.4. Given a query vector, hashing is applied to obtain the query binary code/hash key and retrieve the gallery points that fall into the same hash bucket or within a given Hamming ball radius, r , as the query [10]. A hash key is considered within the Hamming ball if the query code differs from the hash key index in r bits or less.

ANN search via Tree-based and Quantization-based methods also require representative compact features to be effective and efficient [2, 4]. While Quantization-based search

Figure 1.4: Approximate Nearest Neighbor Search in the Hamming space. Given a query code, we select the samples that fall under a Hamming Ball Radius of $r = 1$.

methods have provided promising performance, they use an Asymmetric distance computation (ADC) metric [4] to compute distance between query and gallery samples. Given feature vector subdivided into M subspaces $\mathbf{x} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(M)}]$ where $\mathbf{x}^{(M)} \in \mathbb{R}^{p/M}$ and a codebook of K clusters for each subspace, $\mathbf{C}^{(m)} \in \mathbb{R}^{p/M}$, the ADC is computed as follows:

$$ADC(\mathbf{x}_q, \mathbf{x}_i) = \sum_{m=1}^M d(\mathbf{x}_q^{(m)}, \mathbf{C}^{(m)}[u(\mathbf{x}_i)]) \quad (1.7)$$

where $\mathbf{C}^{(m)}[j]$ is the j th cluster in the codebook of the m th subspace, $u(\mathbf{x})$ determines the closest cluster vector from x , and distance metric is a Euclidean distance. To speed-up computation for large datasets, a pre-computed $M \times K$ look-up table for every query is used to immediately identify the distances for each subspace. However, this requires additional storage on look-up tables for every query sample and codebook floating point vectors which may not be suitable for some retrieval applications. Figure 1.3(b) shows the comparison in retrieval time of the ADC metric compared to Hamming distance. As can be seen, the computation speed improved compared to euclidean and cosine from 32 to 128 bits, but the Hamming metric is still the fastest.

Figure 1.5: Cross-Modal Retrieval Problem. Features from image and text data are extracted using Convolutional Neural Networks (CNN) and Bag-of-Words (BoW) model, respectively. However, these image and text features present an inherent modality gap between them which makes it difficult for similarity search. How to model functions G and F such that the features are projected into a common space suitable for search is desired.

In this section, we have shown the problem definition of compact feature learning and analyze how retrieval is performed for relevant exhaustive NN and ANN search methods. We also show some advantages of Hashing-based methods and compact binary code learning over Quantization-based methods and compact real-valued learning. Note that in our thesis, our focus is on *learning representative compact features* and are not concerned with optimizing the search mechanism in the feature space. We measure the performance of our compact features through experimental validation based on accuracy and precision on standard benchmark datasets for various multimedia search problems. We present detailed challenges on different multimedia search problems in the following section.

1.2 Multimedia Search

Nearest neighbor (NN) search is most commonly used for single modalities where the query and gallery data are from the same modality. A common NN search problem for single modality is visual search, or also termed as content-based image retrieval (CBIR). It

is the problem of searching for relevant images from a large database given a query image. Such systems do not have meta-data or annotations to aid in indexing and search. As already mentioned, this presents challenges as images are initially interpreted by machines as low-level pixels of high dimension. Hence, these images need to be represented such that semantic content is preserved as much as possible.

There are also scenarios where a variety of multimedia data exist in describing the same concept. These are termed as cross-modal data. For example, images uploaded in websites also include text descriptions illustrating the common content; Face representations can be captured from two different environments (e.g. visible images vs. near-infrared images) which would impose large variations in illumination and color balance. Considering these two examples, there may be instances that only one type of modality is available as a query to retrieve relevant data of another type in the database (e.g. a user can use a text to retrieve relevant pictures, a user can capture a photo to retrieve relevant text information). Hence, it is desirable to develop compact feature learning algorithms suitable for cross-modal retrieval. However, because different modalities would produce very different representations, how to effectively reduce the modality gap and exploit correlation across modalities so that the similarities of samples from different modalities can be accurately measured is a challenging task. Figure 1.5 visualizes the problem of cross-modal search.

Multimedia search may also involve time-series data which are more challenging as it adds another degree of information. A common type of time-series data would be videos where context is represented by successive images called *frames*. Figure 1.6 shows examples of videos that present challenges in compact feature learning. Particularly, videos provide diverse and complex visual patterns consisting of low-level visual content in each frame as well as high-level structured content across frames. Moreover, each video may have a number of image frames that may be redundant, resulting in exhaustive computation

Figure 1.6: Video Search Problem. The first video consists of several semantically relevant scenes but is found in different frames. The second video consists of redundant content across frames. How to extract reliable features that exploit the discriminative information of the video suitable for efficient video search is desired.

between frames, which is impractical. Hence, how to develop a framework for video search where discriminative information from videos can be well exploited efficiently remains an important problem need to be solved.

In this thesis, we propose compact feature learning techniques specific to image search, cross-modal search, and video search.

1.3 Motivation

While many compact feature learning methods have been introduced and have obtained reasonable success in image search, cross-modal search, and video search, there are still several issues that need to be addressed for better efficiency.

- (i) Most cross-modal retrieval methods in literature perform direct projections from the original feature representations, which usually cannot truly capture the high-level semantics from nonlinear real-world data. While there are studies that provide nonlinear transformations based on kernels [11, 12], these models are still limited as they

cannot learn explicit nonlinear mapping functions. While more recent deep learning models have provided nonlinear hierarchical transformations for discriminant feature representations, only a few of them have been implemented specifically for cross-modal retrieval [13–15].

- (ii) Existing cross-modal hashing methods perform relaxation on the binary constraint of the optimization objective which may lead to an accumulated quantization error and possible information loss [8, 16]. Moreover, some cross-modal hashing models [17–19] learn a separate binary code for each modality with similar semantics, which may lead to inconsistent binary codes.
- (iii) While several deep hashing methods have also been proposed [20–22], most perform pairwise and triplet-wise network learning using similarity matrices or metric learning approaches. However, these approaches may not fully utilize the label information and rely heavily on the quality of training samples generated during training. In addition, most deep hashing models have a straightforward architecture of a base network with series of convolution, pooling and fully-connected layers, which includes a hash layer at the top to represent the binary codes. However, this type of architecture is too precise on the training data, which may be not be general enough, inaccurate due to biases, and is prone to over-fitting.
- (iv) Most video hashing methods [23–25] treat each video frame as an image and use conventional image-based hashing [26–28] techniques that are typically shallow and perform linear mapping. While these frameworks are straight-forward, they cannot exploit the spatio-temporal information of videos in the learning stage. Furthermore, hashing image frames individually and comparing binary codes frame-by-frame is computationally expensive.

1.4 Overview of Contributions and Thesis Organization

In this thesis, we address the limitations aforementioned and propose different compact feature learning and compact binary code learning methods for similarity search through the use of various machine learning techniques. As mentioned earlier, our proposed methods contribute to building a **rich multimedia information retrieval system**. This is because our proposed methods, if combined, would be flexible as they are able to tackle single-modality and cross-modality retrieval, for temporal and non-temporal data. In this way, a user can extract relevant multi-modal information by just providing query information with no restriction on the data type. In addition, our work contributes in an algorithmic level of content-based retrieval since our methods were evaluated using images, texts, and videos, but can be actually used for other data types as long as they are encoded first as features. Other examples of data types that can be used are point cloud data which are common outputs from LIDAR used in self-driving cars and 3D data from medical scans.

The specific contributions of this thesis are summarized as follows: We review the related work on current research on compact feature learning for image search, cross-modal search, and video search; as well as the limitations of the existing works. We then introduce literature on the machine learning concepts applied in our proposed methods to address these limitations. We present this in Chapter 2.

To handle nonlinearities of data of different modalities while reducing the modality gap for cross-modal search, a model which can extract high-level semantic representations efficiently from nonlinear relationships across different modalities through deep metric learning is proposed. Particularly, we propose a deep coupled metric learning (DCML) method for cross-modal search. Our DCML designs two neural networks to learn two sets of hierarchical nonlinear transformations (one set for each modality) to nonlinearly map data

samples into a shared feature subspace, under which the intra-class variation is minimized and the inter-class variation is maximized, and the difference of each sample pair captured from two modalities of the same class is minimized. Our DCML method for cross-modality search is described in detail in Chapter 3 and the work has been published in [29].

To address the accumulated approximation error caused by performing relaxation on the binary constraint during cross-modal binary code learning, we propose a new cross-modal discrete hashing (CMDH) method for cross-modality search. We jointly learn binary codes and a series of hashing functions alternately and iteratively. In our approach, we do not relax the binary constraint to avoid approximated solutions. Instead, we perform discrete optimization so that our learned hash codes are more representative. In addition, our CMDH method learns a shared set of binary code for samples from different modalities based on their semantic affinities such that the modality gap is implicitly reduced. Our CMDH method for cross-modality search is described in detail in Chapter 4 and the work has been published in [30] and [31].

To explore more discriminative and general binary codes for multimedia retrieval, we propose a deep variational and structural hashing (DVStH) method. Unlike most existing deep hashing methods which use a series of convolution and fully-connected layers to learn binary features, we develop a probabilistic framework to infer latent feature representations inside the network. We then design a struct layer, instead of a bottleneck hash layer, to obtain the binary codes through a simple encoding procedure. We also extend our DVStH to cross-modal retrieval (CM-DVStH) which first learns a unified binary code through a fusion network; which is then used to learn the modality-specific networks. Our DVStH and CM-DVStH for single-modality and cross-modality search are described in detail in Chapter 5. The work in this chapter has been published in [32] and [33].

To perform better video hashing such that we do not need to learn binary codes for every video frame, we propose a deep video hashing (DVH) method which learns strong features by performing frame fusion in the network space. Particularly, we build an end-to-end deep framework which utilizes spatio-temporal information after the stacked convolutional-pooling layers. We discriminatively train our model with a Siamese network by maximizing the inter-class distance and minimizing the intra-class distance of the video feature pairs, as well as minimizing the quantization loss between real-valued codes and binary codes. Our DVH method for video search is described in detail in Chapter 6 and this work has been published in [34].

The final chapter, Chapter 7, summarizes this thesis and provide some future research directions.

Chapter 2

Literature Review

The related works are divided into two parts. The first three sections discuss prior work used for multimedia search, particularly for single-modal search, cross-modal search, and video search. The last section discusses deep learning techniques introduced in different visual domains which are related to our proposed algorithms for multimedia search.

2.1 Single-Modal Search

Approximate Nearest Neighbor (ANN) search methods play an important role for efficient single-modal retrieval and most commonly, for visual retrieval problem. Representative methods include tree-based methods [3, 35, 36], quantization-based methods [4, 5] and hashing-based methods [26, 37, 38]. Hashing-based methods are popular ANN search methods known for its efficiency in memory and speed requirements. We shall review shallow and deep hashing-based methods on the following subsections.

2.1.1 Shallow Hashing methods

Existing hashing-based methods can be generally classified into either data-independent or data-dependent. Data-independent hashing methods use random projections to create the hashing functions such as the Locality-Sensitive Hashing (LSH) method [37]. Specifically,

the LSH is the most popular and has been extended in different ways [39, 40]. However, the LSH can only provide promising performance from longer codes which may be difficult for applications that require more compact representation. Data-dependent hashing or learning-based hashing methods utilize statistical learning to learn efficient hashing functions and map data samples into compact binary features. Representative methods include subspace models [38, 41], kernel models [42], clustering models [43] and boosting models [44]. We shall briefly elaborate representative data-dependent hashing-based methods below.

Spectral Hashing (SH) [26]: Different from LSH that relies on random projections for hashing, SH proposed to learn compact binary codes to be dependent on training data. Particularly, SH learns binary codes, $\{\mathbf{b}_i\}_{i=1}^N$ from high-dimensional feature $\{\mathbf{x}_i\}_{i=1}^N$, by optimizing the following equation:

$$\begin{aligned} \min \quad & \sum_{i,j} W_{i,j} \|\mathbf{b}_i - \mathbf{b}_j\|_2^2 \\ \text{s.t.} \quad & \mathbf{b}_i \in \{-1, 1\}^k \\ & \sum_i \mathbf{b}_i = 0 \\ & \frac{1}{n} \sum_i \mathbf{b}_i \mathbf{b}_i^\top = \mathbf{I} \end{aligned}$$

where $W_{i,j} = \exp(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\epsilon^2})$. The loss term preserves the similarity from real-valued codes while the constraints mathematically define a good binary code suitable for search. The first constraint requires the binary codes to only have a value of -1 and +1¹. The second constraint requires balanced bits, where for every k th bit, it is expected to have a 50% chance to have a value of -1 and 50% be +1. The last constraint requires bits to be uncorrelated. The whole formulation can be solved by relaxing the binary and independence

¹During formulation it is more common to represent binary codes as $\{-1, 1\}$ than $\{0, 1\}$ to make it zero-centered

constraint and computing the eigenvectors of a graph Laplacian.

Iterative Quantization (ITQ) [41]: ITQ first performs dimensionality reduction by solving for $\mathbf{W}$ in $\mathbf{B} = \text{sgn}(\mathbf{W}^\top \mathbf{X})$. This is done by maximizing the variance of each bit which leads to uncorrelated compact codes:

$$\max \sum_k \text{var}(\mathbf{h}_k(\mathbf{x})), \quad \frac{1}{n} \mathbf{B}^\top \mathbf{B} = \mathbf{I} \quad (2.1)$$

Relaxation on the binary constraint into a signed magnitude is then applied to simplify the formulation which can then be solved through applying Principal Component Analysis (PCA) and retrieving the top K eigenvectors. Quantization loss is then minimized by alternatively optimizing the binary code $\mathbf{B}$ and an orthogonal rotation matrix $\mathbf{R} \in \mathbb{R}^{K \times K}$ in the formulation below:

$$\min_{\mathbf{B}, \mathbf{R}} \|\mathbf{B} - \mathbf{R}(\mathbf{W}^\top \mathbf{X})\|_F^2 \quad (2.2)$$

The ITQ can be extended in a supervised manner by using Canonical Correlation Analysis (CCA) instead of PCA during dimension reduction.

Kernel Supervised Hashing (KSH) [42]: KSH uses a kernel function, $k : \mathbb{R}^d \rightarrow \mathbb{R}^r$, to define a nonlinear transformation for $\mathbf{x}$. The kernel function follows the KLSH [39] which exploits anchor samples such that $k(\mathbf{x}) = [\kappa(\mathbf{x}, \mathbf{a}_1) - \mu_1, \dots, \kappa(\mathbf{x}, \mathbf{a}_g) - \mu_g]$ where $\mathbf{a}_m$ is the g th anchor vector and $\kappa(\cdot, \cdot)$ would be any kernel mapping function like the Radial Basis Function (RBF) kernel. A linear projection matrix, $\mathbf{P} \in \mathbb{R}^{r \times K}$, is then learned to map the samples from kernel space into a K -dimensional space suitable to threshold binary codes as follows:

$$\min_{\mathbf{P}} \left\| \frac{1}{K} \text{sgn}(\mathbf{P}^\top \mathcal{K}) \text{sgn}(\mathbf{P}^\top \mathcal{K})^\top - \mathbf{S} \right\|_F^2 \quad (2.3)$$

where $\mathcal{K} = [k(\mathbf{x}_1), \dots, k(\mathbf{x}_n)]$. The loss function exploits code inner products to preserve

the similarity between samples. A greedy optimization is performed to solve the formulation which involves relaxing the $\text{sgn}(\cdot)$ into a sigmoid-shaped function and optimizing the loss by each bit sequentially such that a residue matrix is minimized at every iteration.

Supervised Discrete Hashing (SDH) [8]: SDH performs discrete binary code learning by treating $\mathbf{B}$ as a separate variable independent from the hashing function $h(\cdot)$ such that it optimizes a classification loss and a quantization loss as follows:

$$\min_{\mathbf{B}, \mathbf{W}, h(\cdot)} = \sum_{i=1}^N \|\mathbf{y}_i - \mathbf{W}^\top \mathbf{b}_i\|_2^2 + \lambda \|\mathbf{W}\|_F^2 + \eta \sum_{i=1}^N \|\mathbf{b}_i - h(\mathbf{x}_i)\|_2^2 \quad (2.4)$$

where $\mathbf{y}_i = [y_{i,1}, y_{i,2}, \dots, y_{i,C}]$ is the label vector where $y_{i,c} = 1$ if $\mathbf{x}_i$ belongs to class c , and $h(\cdot)$ can be any linear or nonlinear hashing function. The formulation can be solved through an alternating optimization procedure such that other variables are fixed each time one variable is solved. During optimization, $\mathbf{B}$ is not relaxed to a signed-magnitude and is instead learned through a cyclic coordinate descent method.

Aside from these models, there are also recent works in this area. Particularly, Hu *et al.* [45] learned binary codes by minimizing the reconstruction error between the cosine similarities of the original real-valued features and the output binary codes. It is similar to the Binary Reconstructive Embedding work [46] except they are dealing with cosine similarities for reconstruction; hence, their method is coined as Angular Reconstructive Embeddings (ARE). Another recent work is from Luo *et al.* [47] which applies kernel feature mapping and exploits a label map embedding instead of a similarity matrix to exploit multi-label information. Similar to SDH, they used an iterative and alternating optimization procedure for their loss function. Finally, Cakir *et al.* [48] proposed to learn binary codes by a novel neighborhood preservation loss suitable for weighted Hamming distance retrieval of binary codes. Once binary codes are learned on training data, they can learn

any hash function by mapping the inferred binary codes. In their work, they used a deep network as their hashing function and optimized through a hinge-loss criterion.

2.1.2 Deep Hashing methods

While the aforementioned learning-based hashing methods have achieved promising performance through statistical linear and nonlinear projections, these hashing models require pre-computed feature vectors also known as *hand-crafted features*. This greatly limits the ability to exploit and process all the information we can obtain from raw images during training of hashing functions. Hence, many deep learning-based hashing models have been proposed to learn representative binary codes from raw images for image search, and they can be categorized into unsupervised [49, 50] and supervised methods [51–56].

Semantic hashing [57] is the first work on applying deep learning techniques to learn hashing functions for search. They used a deep generative model in the form of a stacked Restricted Boltzmann Machine (RBM) to learn compact binary codes for document search. Srivastava *et al.* [58] proposed a deep Boltzmann machine approach for multi-modal retrieval. However, their method ignored the binary constraints in learning hashing functions. Recently, Masci *et al.* [51] proposed a coupled Siamese neural network to jointly maximize the intra-similarity and inter-similarity metric for single-modality and cross-modality retrieval tasks. Xia *et al.* [52] proposed a deep convolutional network to learn the hash codes by exploiting a pairwise similarity matrix. Zhao *et al.* [53] trained a network with triplet-wise supervision such that a metric loss based on a rank-criterion is optimized. Lin *et al.* [54] learned binary codes by using a deep architecture which utilizes the hidden layers to represent the latent concepts dominated by the class labels. Liu *et al.* [56] proposed a network that maximizes the discriminability of samples in the Hamming space using pairwise supervision and imposes regularization on the real-valued codes to approximate

discrete binary values. More recently, Shen *et al.* [50] proposed an unsupervised deep hashing approach that preserves an anchor-based similarity graph through discrete optimization. Dizaji *et al.* [59] also proposed an unsupervised deep hashing network learned by coupling generative adversarial networks (GAN). Cao *et al.* [60] designed deep networks with smoothed activations that evolve during training to optimize the learning of binary codes. Cao *et al.* [61] used pairwise labels trained using cross-entropy loss based on Cauchy distribution.

These recent deep hashing models for image search usually consist of a series of convolution, pooling and fully connected layers with a hash layer that outputs a real-valued output of length K . The binary codes are then learned by thresholding the output of the hash layer. These networks are trained using different loss functions for discriminability while encouraging efficient binary codes to be learned. Example loss functions of representative supervised deep hashing-based methods are presented below.

Deep Supervised Hashing (DSH) [56]: Given $\mathbf{b}_i = \phi(\mathbf{x}_i, \Theta)$ as the output of a deep network with input $\mathbf{x}_i$ and network parameter Θ , the DSH is trained with a loss function given as

$$\begin{aligned} \mathcal{L} = & \sum_i^N \sum_j^N \left\{ \frac{1}{2} (1 - S_{i,j}) d_h(\mathbf{b}_i, \mathbf{b}_j) + \frac{S_{i,j}}{2} \max(\tau - d_h(\mathbf{b}_i, \mathbf{b}_j), 0) \right. \\ & \left. + \alpha (\|\mathbf{b}_i\| - 1)_1 + \|\mathbf{b}_j\| - 1)_1 \right\} \end{aligned} \quad (2.5)$$

where d_h is the hamming distance, $S_{i,j}$ is 1 if $\mathbf{x}_i$ and $\mathbf{x}_j$ have similar labels, $|\cdot|$ is the element-wise absolute value, and $\tau > 0$ is a margin threshold. The first term is a contrastive loss to ensure similar (dissimilar) pairs are close (far) from each other and the second term is a regularizer to impose discrete values for the network's real-valued output.

Convolution Neural Network Hashing (CNNH) [52]: This work also uses pairwise

samples and minimizes a reconstruction loss as follows:

$$\mathcal{L} = \sum_i^N \sum_j^N (S_{i,j} - \frac{1}{K} \mathbf{b}_i \mathbf{b}_j^\top) = \left\| \mathbf{S} - \frac{1}{K} \mathbf{B} \mathbf{B}^\top \right\|_F^2 \quad (2.6)$$

The loss term preserves the similarity between pairs by minimizing the difference between code inner product and similarity matrix. A coordinate descent algorithm is proposed to solve the non-convex term $\mathbf{B} \mathbf{B}^\top$.

Deep Semantic Ranking Hashing (DSRH) [53]: DSRH preserves the multi-level similarity information of samples by exploiting set of triplets and optimizing the network's loss function:

$$\mathcal{L} = \sum_i^N \sum_{j,k \neq i}^N L_{triplet}(\mathbf{b}_i, \mathbf{b}_j, \mathbf{b}_k) + \frac{\alpha}{2} \|\text{mean}(\mathbf{b}_i)\|_2^2 + \frac{\beta}{2} \|\mathbf{W}\|_F^2 \quad (2.7)$$

$$L_{triplet}(\mathbf{b}, \mathbf{b}_1, \mathbf{b}_2) = \omega \max(d_h(\mathbf{b}, \mathbf{b}_1) - d_h(\mathbf{b}, \mathbf{b}_2) + \tau, 0) \quad (2.8)$$

where ω is an adaptive weight function based on the label similarity of i, j and i, k . The first term is triplet loss function which controls the margin between the distance of two pairs. The second term ensures balanced bits by encouraging each bit to have a mean of zero over the training data. The last term is a regularizer for the weight parameters in the network.

The general overview of single-modal hashing can be found in Table 2.1. In this thesis, we contribute to single-modal hashing for ANN search in Chapter 5. Particularly, we propose a deep hashing method with a deep architecture with a variational block and struct layer to improve the learning of compact binary codes. Here, representative latent features from our network is regularized from a prior distribution to encourage a probabilistic approach in generating binary codes. In addition, different from most deep hashing methods that use pairs or triplets of data as input, our supervised model is learned in a point-wise

Table 2.1: General Overview of Single-Modal Hashing Methods.

Category			Algorithm
Shallow Hashing methods	unsupervised	data-independent subspace modeling anchor-based cluster-based	LSH [37] PCAH [38], PCA-ITQ [41], Spectral Hashing [26] AGH [62], DGH [16] K-means Hashing [43]
	supervised	subspace modeling kernel-based boosting metric-based	CCA-ITQ [41], SPLH [38] KSH [42], SDH [8], FSSH [47] FastHash [44] BRE [46], ARE [45], HBMP [48]
Deep Hashing methods	unsupervised		Semantic Hashing [57], DH [49], SADH [50], HashGAN [59]
	supervised	pair-wise	DPSH [63], DSH [56], CNNH [52], BDNN [64], HashNet [60], ADSH [65], DCH [61]
		triplet-wise	DTSH [66], DSRH [53], DRSC [67], DNNH [55]
		point-wise	SuBIC [68], DLBHC [54]

manner (data are fed element by element) where the binary codes are learned based on a cross-entropy loss.

In this thesis, we also contribute to the extensions of single-modal search problem: cross-modal and video search. The succeeding sections tackle their literature review.

2.2 Cross-Modal Search

Cross-modal search aims to transform data samples of different modalities but of similar semantics to have similar compact features as possible. We briefly introduce several methods on cross-modal matching, shallow cross-modal hashing, and deep cross-modal hashing methods the succeeding subsections.

2.2.1 Cross-Modal Matching methods

Existing cross-modal matching methods can be categorized into two types: homogenous data synthesis [69, 70] and cross-modal invariant feature learning [71–75]. In our thesis, we focus on the latter. For the first type, data from one modality are synthesized into another

modality so that the modality gap is reduced. For the second type, data instances from different modalities are mapped into a common latent space so that the modality difference is reduced and they can be measured directly. Most existing cross-modal matching methods learn a single latent space through a pair of linear transformations to reduce the modality gap. However, real-world data are typically nonlinear in nature, hence these methods may not be fully effective for reducing the modality gap during cross-modal matching. There are several kernel-based methods that provide nonlinear feature representations which are invariant for cross-modalities. For example, Haroon *et al.* [11] used kernel canonical correlation analysis (KCCA) to learn a common semantic space for cross-modal retrieval. Gong *et al.* [76] performed nonlinear kernel embedding followed by a linear dimensionality reduction and CCA for content-based retrieval and tag-image search. While these methods may provide representative features, they may not be scalable when new training data become available due to the kernel function computation. In addition, most kernel-based methods do not exploit label information which makes them less discriminative.

In recent years, several coupled metric learning algorithms have been proposed for cross-modal matching such as cross-modal metric learning (CMML) [77], maximum-margin coupled mappings (MMCM) [78], and coupled marginal fisher analysis (CMFA) [79]. However, these methods only learn and employ modal-specific linear projections to transform the cross-modal samples into a shared latent feature space. This may not be effective enough to discover the nonlinear relationship of real-world samples. Hence, in this thesis, we explore deep metric learning for cross-modal matching to capture the nonlinearities of data effectively for retrieval.

2.2.2 Shallow Cross-Modal Hashing methods

Cross-modal hashing methods can be classified into *unsupervised* [80,81] and *supervised* [67, 82–84]. Unsupervised methods utilize co-occurrence information such that only the image-

text pairs which occurred in the same article are known to be of similar semantic. Differently, supervised methods utilize semantic labels to enhance the correlation of cross-modal data. Examples of representative cross-modal hashing methods are the following, Bronstein *et al.* [17] proposed to learn hashing functions by preserving the intra-class similarity and performing boosting and eigen-decomposition operations. Kumar *et al.* [19] extended the spectral hashing method from single-view to cross-view by minimizing the cross-modal similarity. Rastegari *et al.* [18] performed dual-view mapping to learn the binary codes such that data points which are near in the original space are also near in the hash space. Ding *et al.* [81] learned a unified binary code for the two modalities in the training stage by performing matrix factorization with a latent factor model. Zhang *et al.* [67] utilized semantic labels to maximize the semantic correlation between cross-modal data in an iterative and sequential manner. Lin *et al.* [82] learned unified binary codes for cross-modal data by transforming the hamming distance and affinity matrix in two probability distributions and minimizing their Kullback-Leibler divergence. Zheng *et al.* [85] represented homogenous data into submanifolds to create a hetero-manifold for multi-modal data, where the manifold is regularized through SVM according to specific criterions that preserve the geometric structure and global information. We briefly elaborate some of these methods below.

Predictable Dual-View Hashing (PDH) [18]: Suppose $\mathbf{X}_u$ and $\mathbf{X}_v$ be data matrices from different modalities with similar context, PDH learns linear projections $\mathbf{W}_u$ and $\mathbf{W}_v$ as follows:

$$\begin{aligned} \min_{\mathbf{W}_u, \mathbf{W}_v} \quad & \|\mathbf{W}_v^\top \mathbf{X}_v - \mathbf{B}_u\|_F^2 + \|\mathbf{W}_u^\top \mathbf{X}_u - \mathbf{B}_v\|_F^2 + \|\mathbf{B}_u \mathbf{B}_u^\top - \mathbf{I}\|_F^2 + \|\mathbf{B}_v \mathbf{B}_v^\top - \mathbf{I}\|_F^2 \\ \text{s.t.} \quad & \mathbf{B}_u = \text{sgn}(\mathbf{W}_u^\top \mathbf{X}_u) \quad \mathbf{B}_v = \text{sgn}(\mathbf{W}_v^\top \mathbf{X}_v) \end{aligned} \quad (2.9)$$

Here, the first and second terms implicitly reduce the modality gap by minimizing the difference between real-valued projected vectors from modality u and learned binary codes

from modality v , and vice versa. The third and fourth terms impose low correlation of bits. The formulation is optimized by regularizing the loss function with max-margin constraints and applying Expectation Maximization (EM) algorithm to solve the multiple Support Vector Machines (SVM) problem.

Collective Matrix Factorization Hashing (CMFH) [81]: Here, the feature vector for each modality is represented by matrix factorization: $\mathbf{X}_* = \mathbf{P}_* \mathbf{V}$ using basis matrix $\mathbf{P} \in \mathbb{R}^{d \times K}$ and a common latent representation matrix $\mathbf{V} \in \mathbb{R}^{K \times N}$. Suppose the binary codes for each modality is represented as hard-threshold linearly projected feature vectors, the optimization procedure to learn these transform matrices are as follows:

$$\begin{aligned}
 \min_{\mathbf{W}_v, \mathbf{W}_u, \mathbf{P}_v, \mathbf{P}_u, \mathbf{V}} \quad & \mu(\|\mathbf{V} - \text{sgn}(\mathbf{W}_u^\top \mathbf{X}_u)\|_F^2 + \|\mathbf{V} - \text{sgn}(\mathbf{W}_v^\top \mathbf{X}_v)\|_F^2) \\
 & + \lambda \|\mathbf{X}_u - \mathbf{P}_u \mathbf{V}\|_F^2 + (1 - \lambda) \|\mathbf{X}_v - \mathbf{P}_v \mathbf{V}\|_F^2 \\
 & + \gamma R(\mathbf{W}_u, \mathbf{W}_v, \mathbf{P}_u, \mathbf{P}_v, \mathbf{V})
 \end{aligned} \tag{2.10}$$

where $R(\cdot)$ is a regularizer to avoid overfitting. The first two terms enforce the binary codes to be as similar as possible to a common latent semantic representation. The next two terms minimize the reconstruction error between the latent representation and the original real-valued feature vector. The formulation can be optimized by relaxing the $\text{sgn}(\cdot)$ function and solving each variable alternatively while fixing the other variables.

Semantic Preserving Hashing (SePH) [82]: SePH learns discriminative codes in a probabilistic manner where the affinity matrix is transformed into a probability distribution as follows:

$$p_{i,j} = \frac{S_{i,j}}{\sum_{i \neq j} S_{i,j}} \tag{2.11}$$

and the pairwise Hamming distances is also transformed in a probability distribution using

t-SNE distribution as follows:

$$q_{i,j} = \frac{(1 + d_h(\mathbf{b}_i, \mathbf{b}_j))^{-1}}{\sum_{k \neq m} (1 + d_h(\mathbf{b}_k, \mathbf{b}_m))^{-1}} \quad (2.12)$$

The hashing function is then learned by optimizing the following formulation:

$$\min_{\mathbf{B}} \sum_{i \neq j} p_{i,j} \log \frac{p_{i,j}}{q_{i,j}} + \alpha \|\mathbf{B} - \mathbf{I}\|_F^2 \quad (2.13)$$

where the first term minimizes the Kullback-Leibler divergence between the affinity probability and hamming distance probability, while the second term minimizes the quantization loss. This is solved by relaxing the $\text{sgn}(\cdot)$ and performing a gradient descent based optimization.

2.2.3 Deep Cross-Modal Hashing methods

Similar to single-modal hashing methods, cross-modal deep hashing methods have also been proposed recently. For example, Cao *et al.* [86] designed a stacked auto-encoder architecture to jointly maximize the feature and semantic correlation across modalities. However, this work does not perform end-to-end learning which may limit the discriminative representation of data samples, particularly in images. Jiang *et al.* [22] performed an end-to-end deep learning framework with a negative log-likelihood criterion to preserve the similarity between real-valued representations having the same class. However, their training model performs similarity preservation on real-valued codes and not binary codes. Another work from Cao *et al.* [20] learned a visual semantic fusion network with cosine hinge loss to obtain the binary codes, and trained modality-specific deep networks to obtain the hashing functions. Cao *et al.* [87] implemented a cross-modal network which minimizes a collective quantization loss and maximizes a cross-modal correlation between modality training pairs. Shen *et al.* [88] exploited region proposals for the image network

when trying to preserve the semantic similarity of hash code pairs from the image and text network. Li *et al.* [89] designed a pair of adversarial networks to maximize the semantic correlation between modalities and supervised through a separate self-supervised semantic network which uses multi-label annotations. Cao *et al.* [90] proposed a deep framework that is suitable for hamming space retrieval by penalizing using a pairwise focal loss and per-modal quantization loss all based on an exponential distribution. Representative cross-modal deep hashing methods are further elaborated below.

Deep Cross-Modal Hashing (DCMH) [22]: DCMH is an end-to-end coupled network, one for each modality, which learns from an objective function defined as follows:

$$\begin{aligned} \mathcal{L} = & - \sum_{i,j=1}^N (S_{i,j} \Pi_{i,j} - \log(1 + \exp(\Pi_{i,j}))) \\ & + \gamma(\|\mathbf{B} - \mathbf{F}\|_F^2 + \|\mathbf{B} - \mathbf{G}\|_F^2) + \eta(\|\mathbf{F}\mathbf{1}\|_2^2 + \|\mathbf{G}\mathbf{1}\|_2^2) \\ \text{s.t.} \quad & \mathbf{B} \in \{-1, 1\}^{K \times N}, \quad \mathbf{F} = \phi_u(\mathbf{X}_u, \Theta_u), \quad \mathbf{G} = \phi_v(\mathbf{X}_v, \Theta_v) \end{aligned} \quad (2.14)$$

where $S_{i,j}$ is 1 if $\mathbf{x}_{u,i}$ and $\mathbf{x}_{v,j}$ have similar labels, and $\Pi_{i,j} = \frac{1}{2} \mathbf{f}_i^\top \mathbf{g}_j \mathbf{b}_i$ such that $\mathbf{f}_i, \mathbf{g}_j$ is the i th, j th vector of $\mathbf{F}, \mathbf{G}$. The first term is the negative log likelihood loss which maximizes the likelihood that the similarity from the inner product of $\mathbf{F}$ and $\mathbf{G}$ is large if $S_{i,j} = 1$, and small if $S_{i,j} = 0$. The second term minimizes the quantization loss between a unified binary code and real-valued hash code for each modality. The third term promotes balanced bits. The network is trained in an alternative manner such that $\mathbf{B}$ and the network parameters are solved individually while fixing the other variables at every iteration.

Correlation Hashing Network (CHN) [21]: CHN is a coupled network which solves

an objective function defined below:

$$\begin{aligned} \mathcal{L} = & \sum_{i,j=1}^N \max \left(\delta - S_{i,j} \frac{\langle \mathbf{b}_{u,i}, \mathbf{b}_{v,j} \rangle}{\|\mathbf{b}_{u,i}\| \|\mathbf{b}_{v,j}\|}, 0 \right)^2 \\ & + \eta \sum_{i=1}^N \left(\max \left(\delta - \frac{\langle \|\mathbf{b}_{u,i}\|, \mathbf{1} \rangle}{\|\mathbf{b}_{u,i}\| \|\mathbf{1}\|}, 0 \right) + \max \left(\delta - \frac{\langle \|\mathbf{b}_{v,i}\|, \mathbf{1} \rangle}{\|\mathbf{b}_{v,i}\| \|\mathbf{1}\|}, 0 \right) \right) \end{aligned} \quad (2.15)$$

where $\langle \cdot \rangle$ is the inner product, $\|\cdot\|$ is the vector length, and $0 < \delta \leq 1$ is a threshold margin parameter. The first term is max-margin loss function using cosine distance which mitigates the diversity of vector lengths improving representation. The second term is another max-margin loss that minimizes the quantization loss by reducing the gap between hamming distance and cosine distance. The network is trained by a standard back-propagation procedure.

The general overview of cross-modal search methods is found in Table 2.2. In this thesis, we contribute to cross-modal search and is found in three chapters (Chapters 3,4, and 5). First, we propose a real-valued feature learning approach through deep learning with supervised information. Different from the few deep learning strategies proposed for cross-modal matching, we propose a deep metric learning based approach to perform nonlinear transformations and reduce the modality gap between modalities. Second, most cross-modal hashing methods perform relaxation to the binary constraint in order to solve their respective formulation. Hence, we propose a discrete optimization technique for our cross-modal hashing model, both in a shallow and deep framework, to avoid the approximation loss caused by relaxations. Finally, most deep learning based cross-modal methods use a common network architecture with a hash layer with their respective loss functions to obtain similar binary codes. Hence, we propose a two-stage network which first fuses the two input modalities for binary code learning, and then learns modality-specific networks. Our network also introduces a variational block and struct layer.

Table 2.2: General Overview of Cross-Modal Search Methods.

Category			Algorithm
Real-Valued Feature Learning	unsupervised	subspace modeling deep learning	CCA [91], KCCA [11], PLS [72] Multimodal DBM [13], Multimodal Autoencoder [14]
	supervised	subspace modeling dictionary modeling deep learning	GMA [92], LCFS [73], LGCFL [74], JFSSL [93], MvDA [94] CDL [95] RE-DNN [96], LRBS [97]
Binary Code Learning (Hashing)	unsupervised	subspace modeling manifold modeling	CVH [19], PDH [18], CMFH [81], LSSH [80] HMR [85]
	supervised	subspace modeling kernel-based deep learning quantization-based	SCM [71] SePH [82] DCMH [22], CHN [21], CAH [86], TVDB [88], SSAH [89], CMMH [90]. CDQ [87]

2.3 Video Search

Previous works on video hashing were mostly used for near-duplicate video retrieval tasks and several of them focused on video feature representation rather than learning the hashing functions [23–25]. For example, Song *et al.* [23] introduced a multiple feature hashing method which utilizes multiple features and extracts different local structures to obtain efficient binary codes. Douze *et al.* [25] extracted representative spatial-temporal features from images and used conventional hashing methods to obtain binary codes. There are only a few video hashing methods proposed for content-based retrieval. For example, Cao *et al.* [98] proposed a submodular hashing framework which selects relevant frames from videos to learn the hashing functions for efficient video search. However, it did not really learn a video-based hashing function by using statistical knowledge. Ye *et al.* [99] proposed a supervised structural learning framework which exploits the temporal consistency to learn the linear hashing functions. However, it only learns a linear projection which may not truly capture the nonlinear nature of video data. Li *et al.* [100] proposed a hashing model across the Euclidean space and the Riemannian manifold, which learns hashing

Table 2.3: General Overview of Video Hashing Methods.

Category		Algorithm
Near-duplicate Retrieval	use single-modal hashing methods	Song <i>et al.</i> [23], Douze <i>et al.</i> [25], Coskun <i>et al.</i> [24]
	data-independent	Submodular Video Hashing [98]
Content-based Retrieval	subspace modeling	VHDT [99]
	covariance modeling	CVC [103]
	unsupervised deep modeling	SSVH [102], UDVH [101]

functions based on the kernel max-margin framework for face video retrieval. However, their work represented videos with a single feature representation in a form of a covariance matrix, which may not fully exploit the spatio-temporal information in videos. To our knowledge, there are only a few works that have investigated deep architectures for content-based video hashing. For example, Wi *et al.* [101] proposed a spatial and temporal deep model that learns the hash codes through steps that perform feature clustering, preserve a neighborhood structure, and use a rotation matrix for minimizing quantization error. Another example, Song *et al.* [102] designed a binary autoencoder that exploits temporal information in videos and learns binary codes through minimizing the reconstruction error and preserving neighborhood structure among frames. While these two recent deep models have explored temporal information in their frameworks, they are unsupervised and do not exploit label information during training.

The general overview of video hashing methods is found in Table 2.3. In this thesis, we contribute to video hashing for content-based retrieval and is found in Chapter 6. Particularly, we explore different deep architectures that exploit both temporal and discriminative information in videos.

2.4 Deep Learning

Since convolutional neural networks (CNN) have achieved big successes in extracting high-level semantics for recognition, several works have exploited deep CNN structures for their

computer vision models such as in object detection and recognition [104, 105], scene labelling [106, 107], and action recognition [28]. We present current research on deep learning relevant to this thesis in the succeeding subsections. Particularly, we present literature on deep metric learning, supervised deep learning for video analysis, and deep generative models.

2.4.1 Deep Metric Learning

Over the past decade, metric learning methods [108] have shown success in different machine learning applications. Most recently, deep metric learning methods have been proposed to perform explicit nonlinear mapping through a series of nonlinear transformations in a deep network which is trained using various metric-based loss functions to preserve similarity of data. The aim of metric-based learning is to learn a new embedding space such that the distance of transformed points of semantically similar samples are close, while the dissimilar samples are far. Common metric-based losses that showed success in deep metric learning are the contrastive loss [109] and triplet loss [110, 111]. These deep metric learning methods have shown promising performance for different visual analysis applications including face recognition [112], person re-identification [113], and visual tracking [114]. In this thesis, we apply deep metric learning for cross-modal search and video search found in Chapters 3 and 6, respectively.

2.4.2 Supervised Deep learning for Video Analysis

A number of deep learning methods have also been proposed for video analysis and most of them focused on video action recognition [28, 115], video classification [116, 117] and event detection [75]. For example, Ji *et al.* [28] introduced a 3D CNN approach which considers spatial and temporal information for action recognition. Karpathy *et al.* [116]

presented an extensive evaluation of different deep architectures for large-scale video classification where they introduced different spatio-temporal convolutions based on how the frames are fused in the network. Ng *et al.* [117] exploited different feature fusion methods after the stacked convolution and pooling layers and investigated the Long Short Term Memory (LSTM) networks for video classification. Xu *et al.* [75] explored pooling and encoding methods to combine frame-level features for event detection. Simonyan *et al.* [115] implemented a two-stream convolution network for action recognition in videos to model the spatial and temporal data individually, and fuse the scores together. In this thesis, we focus on various deep architectures for videos that are suitable for video hashing and search. It is found in Chapter 6.

2.4.3 Deep Generative Models

More recently, deep generative models, which aim to learn a data distribution from training samples in a deep framework, have gained a lot of attention in computer vision, especially variational autoencoders (VAEs) [118] which combine the strengths of both deep learning and probabilistic models. While VAEs have shown great progress and success in various visual applications [119, 120] only a few attempts have been made for multimedia hashing and retrieval. Specifically, Chaidaroon *et al.* [121] used variational encoder model for unsupervised and supervised text hashing. Hu *et al.* [122] also performed variational inference to learn representative latent factors but was not applied in a deep framework. In this thesis, we implement variational inference in a deep model for both single and cross-modality retrieval and present it in Chapter 5.

2.5 Chapter Summary

In this chapter, we have reviewed the literature on single-modality search, cross-modality search, and video search; which are the three multimedia applications that we tackle in this thesis. We also briefly reviewed the literature on deep learning concepts such as deep metric learning, deep learning for video analysis, and deep generative models; which we draw inspiration to solve the aforementioned multimedia search problems.

In the following chapters, we introduce our methods in more detail and show their performance on benchmark datasets. Particularly, Chapter 3 addresses the cross-modal retrieval problem using deep metric learning. Chapter 4 presents cross-modal hashing using simultaneous binary code learning and hashing function learning procedure optimized in a discrete manner. Chapter 5 presents single-modal and cross-modal hashing using hashing networks having probabilistic interpretation and structural output. Finally, Chapter 6 addresses the video hashing problem using deep metric learning and deep network with frame fusion suitable for video search.

Chapter 3

Deep Coupled Metric Learning

In this chapter, we propose a new deep coupled metric learning (DCML) method for cross-modal matching, which aims to match samples captured from two different modalities (e.g. texts vs. images, visible vs. near infrared images). Unlike existing cross-modal matching methods which learn a linear common space to reduce the modality gap, our DCML designs two feed-forward neural networks which learn two sets of hierarchical nonlinear transformations (one for each modality) to nonlinearly map samples from different modalities into a common latent feature space. We train the networks such that the intra-class (inter-class) variation is minimized (maximized) and the difference between the data pairs captured from two modalities of the same class is minimized. Experimental results on four different cross-modal matching datasets validate the efficacy of the proposed approach.

3.1 Introduction

A variety of cross-modal matching methods have been proposed in recent years, and the typical approach is to seek one common semantic space to reduce the modality gap. While several cross-modal matching methods [71–75] have achieved encouraging performance, most of them employ direct projections from the original feature representations, which usually cannot truly capture the high-level semantics from nonlinear real-world data. While

Figure 3.1: Basic idea of the proposed DCML method for cross-modal matching. For each pair of samples which were captured from two different modalities (e.g. modality U and modality V), we first represent them as two feature vectors $\mathbf{x}_u$ and $\mathbf{x}_v$. Then, we pass them into the designed two feed-forward neural networks to nonlinearly map them into a shared feature subspace, where each network contains a set of hierarchical nonlinear transformations such that the feature vectors are represented as $\mathbf{h}_u^{(2)}$ and $\mathbf{h}_v^{(2)}$, respectively. Finally, we perform discriminative loss at the top layers of the networks for training.

there are studies that provide nonlinear transformations based on kernels, these models do not explicitly learn discriminative nonlinear mapping. Even though more recent deep learning models have provided nonlinear hierarchical transformations for discriminant feature representations, only a few of them have been implemented particularly for cross-modal matching [13–15]. Hence, it is desirable to learn a model which can extract high-level semantic representations efficiently from nonlinear relationships across different modalities.

We propose a new deep coupled metric learning (DCML) method for cross-modal matching. Unlike most existing methods modal-invariant feature learning methods such

as CCA and PLS which learn a single linear latent space to reduce the modality gap, our DCML designs two neural networks to learn two sets of hierarchical nonlinear transformations (one for each modality) to nonlinearly map samples from different modalities into a common latent feature space where the intra-class (inter-class) variation is minimized (maximized), and the difference between the data pairs captured from two modalities of the same class is minimized. Figure 3.1 illustrates the basic idea of the proposed approach. Experimental results on four different cross-modal matching applications demonstrate the effectiveness of the proposed method.

3.2 Deep Coupled Metric Learning

In this section, we first briefly review the coupled metric learning approach. We then present the proposed DCML method and its implementation details.

3.2.1 Coupled Metric Learning

Let $\mathbf{X}_u = [\mathbf{x}_{u,1}, \mathbf{x}_{u,2}, \dots, \mathbf{x}_{u,N}] \in \mathbb{R}^{d_u \times N}$ and $\mathbf{X}_v = [\mathbf{x}_{v,1}, \mathbf{x}_{v,2}, \dots, \mathbf{x}_{v,N}] \in \mathbb{R}^{d_v \times N}$ be samples acquired from two different modalities, $\mathbf{x}_{u,i} \in \mathbb{R}^{d_u}$ and $\mathbf{x}_{v,i} \in \mathbb{R}^{d_v}$ are the i th corresponding pair, where $1 \leq i \leq N$, and d_u and d_v are the corresponding feature dimensions. Coupled metric learning approach aims to seek the following projection functions

$$g_u : \mathbb{R}^{d_u} \rightarrow \mathbb{R}^d, \quad g_v : \mathbb{R}^{d_v} \rightarrow \mathbb{R}^d \quad (3.1)$$

where $d \ll \min(d_u, d_v)$ is the feature dimension of the learned latent common space, where the similarity of these two sets is maximized so that the gap difference is reduced as much as possible.

Most existing coupled metric learning algorithms [91, 123] design g_u and g_v as linear

functions defined as

$$g_u(\mathbf{X}_u) = \mathbf{W}_u^\top \mathbf{X}_u, \quad g_v(\mathbf{X}_v) = \mathbf{W}_v^\top \mathbf{X}_v \quad (3.2)$$

where $\mathbf{W}_u \in \mathbb{R}^{d \times d_1}$ and $\mathbf{W}_v \in \mathbb{R}^{d \times d_2}$ are two linear projection matrices.

Many criterions can then be employed to measure the similarity of $\mathbf{X}_u$ and $\mathbf{X}_v$ in the learned metric space g_u and g_v , and the two most popular measures are canonical correlation maximization and Euclidean distance minimization [77, 92, 94].

3.2.2 Deep Coupled Metric Learning

As shown in Figure 3.1, we construct a coupled deep neural network to compute feature representations for cross-modal data, using one network for each modality. Given each pair of data instance from two modalities, we pass them into modality-specific deep networks having $L + 1$ layers each. Each network consists of multiple stacked layers of nonlinear transformations, where the l th layer ($1 \leq l \leq L$) for networks U and V have $d_u^{(l)}$ and $d_v^{(l)}$ units, respectively. Suppose $\mathbf{x}_u$ and $\mathbf{x}_v$ are the two data instances from the two modalities, the outputs of the first layer in these two networks are computed as $\mathbf{h}_u^{(1)} = s(\mathbf{W}_u^{(1)\top} \mathbf{x}_u + \mathbf{c}_u^{(1)})$ and $\mathbf{h}_v^{(1)} = s(\mathbf{W}_v^{(1)\top} \mathbf{x}_v + \mathbf{c}_v^{(1)})$. $\mathbf{W}_u^{(1)}$ and $\mathbf{W}_v^{(1)}$ are the projection matrices to be learned in the first layer of these networks, $\mathbf{c}_u^{(1)}$ and $\mathbf{c}_v^{(1)}$ are the bias vectors, and $s(\cdot)$ is a nonlinear activation function which is applied element-wise. The first layer outputs $\mathbf{h}_u^{(1)}$ and $\mathbf{h}_v^{(1)}$ are used as the inputs of the second layer. The outputs of the second layer are computed as $\mathbf{h}_u^{(2)} = s(\mathbf{W}_u^{(2)\top} \mathbf{h}_u^{(1)} + \mathbf{c}_u^{(2)})$ and $\mathbf{h}_v^{(2)} = s(\mathbf{W}_v^{(2)\top} \mathbf{h}_v^{(1)} + \mathbf{c}_v^{(2)})$, $\mathbf{W}_u^{(2)}$ and $\mathbf{W}_v^{(2)}$ are the projection matrices, and $\mathbf{c}_u^{(2)}$ and $\mathbf{c}_v^{(2)}$ are the bias vectors of the second layer. Similarly, the outputs for the l th layer are $\mathbf{h}_u^{(l)} = s(\mathbf{W}_u^{(l)\top} \mathbf{h}_u^{(l-1)} + \mathbf{c}_u^{(l)})$ and $\mathbf{h}_v^{(l)} = s(\mathbf{W}_v^{(l)\top} \mathbf{h}_v^{(l-1)} + \mathbf{c}_v^{(l)})$.

Figure 3.2: Basic idea of the large-margin loss of our proposed DCML method. There are eight samples captured from two different modalities (e.g. Modality U and V), and each modality has four samples. All samples come from three classes, denoted as circles, triangles, and squares. In the original feature space of each modality, the similarity of the samples from class 3 is less than that of the samples from two different classes (e.g. class 2 and class 3 in Modality U , and class 1 and class 3 in Modality V), which may reduce the matching performance. In addition, the original feature spaces for each modality are very much different. In the learned latent feature space by DCML, we expect that the intra-class similarity is increased and the inter-class similarity is reduced, and the similarity of samples from the same class across different modalities is increased, so that discriminative information can be exploited and modality gap can be reduced simultaneously. Having samples captured from two different modalities into the same semantic space, we can conduct matching for different applications.

The outputs of the top layer are:

$$g_u(\mathbf{x}_u) = \mathbf{h}_u^{(L)} = s(\mathbf{W}_u^{(L)\top} \mathbf{h}_u^{(l-1)} + \mathbf{c}_u^{(L)}) \quad (3.3)$$

$$g_v(\mathbf{x}_v) = \mathbf{h}_v^{(L)} = s(\mathbf{W}_v^{(L)\top} \mathbf{h}_v^{(l-1)} + \mathbf{c}_v^{(L)}) \quad (3.4)$$

where $\mathbf{W}_u^{(L)}$ and $\mathbf{W}_v^{(L)}$ are the projection matrices and $\mathbf{c}_u^{(L)}$ and $\mathbf{c}_v^{(L)}$ are the bias vectors for the top layer.

To improve the cross-modal matching accuracy, we focus on the following objectives:

- We aim to exploit more discriminative information from training samples.

- We aim to reduce the modality gap of the data pair captured from different modalities.

Figure 3.2 illustrates the basic idea of our proposed DCML method. To achieve the first objective, we employ a large-margin criterion to minimize the intra-class variation and maximize the inter-class variation for feature representation at the top layer of these two networks simultaneously. Particularly, for each training pair $\{\mathbf{x}_{u,i}, \mathbf{x}_{v,j}\}$, we compute the squared distance of the top layer outputs of the coupled network $d(\mathbf{h}_{u,i}^{(L)}, \mathbf{h}_{v,j}^{(L)})$ as follows:

$$d(\mathbf{h}_{u,i}^{(L)}, \mathbf{h}_{v,j}^{(L)}) = \left\| \mathbf{h}_{u,i}^{(L)} - \mathbf{h}_{v,j}^{(L)} \right\|_2^2 \quad (3.5)$$

We expect that $d(\mathbf{h}_{u,i}^{(L)}, \mathbf{h}_{v,j}^{(L)})$ is as small as possible if $\mathbf{x}_{u,i}$ and $\mathbf{x}_{v,j}$ are of the same class, and as large as possible if they are from different classes. These are formulated by the following constraints:

$$d(\mathbf{h}_{u,i}^{(L)}, \mathbf{h}_{v,j}^{(L)}) \leq \tau_1, \quad \text{if } S_{u_i, v_j} = 1 \quad (3.6)$$

$$d(\mathbf{h}_{u,i}^{(L)}, \mathbf{h}_{v,j}^{(L)}) \geq \tau_2, \quad \text{if } S_{u_i, v_j} = -1 \quad (3.7)$$

where $S_{u_i, v_j} = 1$ means that $\mathbf{x}_{u,i}$ and $\mathbf{x}_{v,j}$ are of the same class, $S_{u_i, v_j} = -1$ indicates that they are from different classes, and τ_1 and τ_2 are the small and large thresholds, respectively. To reduce the number of parameters, we adopt the following large-margin objective function [124] to integrate these two constraints:

$$\min_{g_u, g_v} H_1 = 1 - S_{u_i, v_j} \left(\tau - d(\mathbf{h}_{u,i}^{(L)}, \mathbf{h}_{v,j}^{(L)}) \right) \quad (3.8)$$

where $\tau_1 = \tau - 1$ and $\tau_2 = \tau + 1$.

To achieve the second objective, we minimize the difference between each pair of data of the same class captured from different modalities over all layers as follows:

$$\min_{g_u, g_v} H_2 = \sum_{l=1}^{L-1} \sum_{i,j=1}^N \left\| \mathbf{h}_{u,i}^{(l)} - \mathbf{h}_{v,j}^{(l)} \right\|_2^2 \quad (3.9)$$

By combining the above two criteria, we write the following optimization formulation for our proposed DCML model as:

$$\begin{aligned}
 \arg \min_{g_u, g_v} H &= H_1 + \lambda_1 H_2 + \lambda_2 H_3 \\
 &= \frac{1}{2} \sum_{i,j=1}^N \zeta \left(1 - S_{u_i, v_j} \left(\tau - d \left(\mathbf{h}_{u,i}^{(L)}, \mathbf{h}_{v,j}^{(L)} \right) \right) \right) + \frac{\lambda_1}{2} \sum_{i,j=1}^N \sum_{l=1}^{L-1} \delta_{ij} \left\| \mathbf{h}_{u,i}^{(l)} - \mathbf{h}_{v,j}^{(l)} \right\|_2^2 \\
 &\quad + \frac{\lambda_2}{2} \sum_{l=1}^L \left(\left\| \mathbf{W}_u^{(l)} \right\|_F^2 + \left\| \mathbf{W}_v^{(l)} \right\|_F^2 + \left\| \mathbf{c}_u^{(l)} \right\|_2^2 + \left\| \mathbf{c}_v^{(l)} \right\|_2^2 \right)
 \end{aligned} \tag{3.10}$$

where H_1 exploits the discriminative information using a large-margin criterion, H_2 reduces the modality gap by preserving the similarity between each layer for similar training pairs, and H_3 represents the regularization of the parameters of the deep networks g_u and g_v . λ_1 and λ_2 are two parameters to balance the effect of the different terms, δ_{ij} is an indicator where it is 1 if $\mathbf{x}_{u,i}$ and $\mathbf{x}_{v,j}$ share the same class and 0 otherwise, and $\zeta(z)$ is a smooth approximation to the hinge loss function ($z = \max(z, 0)$) defined as a generalized logistic loss function as follows¹:

$$\zeta(z) = \frac{1}{\rho} \log(1 + \exp(\rho z)) \tag{3.11}$$

where ρ is the sharpness parameter.

We use stochastic gradient descent to solve the optimization problem in (3.10) to learn the network parameters $\left\{ \mathbf{W}_u^{(l)}, \mathbf{W}_v^{(l)}, \mathbf{c}_u^{(l)}, \mathbf{c}_v^{(l)} \right\} \Big|_{l=1}^L$. Particularly, the gradient of H with

¹We performed empirical tests and found that our method yielded better performance when a smooth approximation for the hinge loss function was used.

respect to these parameters can be computed as follows:

$$\frac{\partial H}{\partial \mathbf{W}_u^{(l)}} = \sum_{i,j=1}^N \left(\Phi_{u,i}^{(l)} \mathbf{h}_{u,i}^{(l-1)} + \lambda_1 \Psi_{u,i}^{(l)} \mathbf{h}_{u,i}^{(l-1)} \right) + \lambda_2 \mathbf{W}_u^{(l)} \quad (3.12)$$

$$\frac{\partial H}{\partial \mathbf{W}_v^{(l)}} = \sum_{i,j=1}^N \left(\Phi_{v,j}^{(l)} \mathbf{h}_{v,j}^{(l-1)} + \lambda_1 \Psi_{v,j}^{(l)} \mathbf{h}_{v,j}^{(l-1)} \right) + \lambda_2 \mathbf{W}_v^{(l)} \quad (3.13)$$

$$\frac{\partial H}{\partial \mathbf{c}_u^{(l)}} = \sum_{i,j=1}^N \left(\Phi_{u,i}^{(l)} + \lambda_1 \Psi_{u,i}^{(l)} \right) + \lambda_2 \mathbf{c}_u^{(l)} \quad (3.14)$$

$$\frac{\partial H}{\partial \mathbf{c}_v^{(l)}} = \sum_{i,j=1}^N \left(\Phi_{v,j}^{(l)} + \lambda_1 \Psi_{v,j}^{(l)} \right) + \lambda_2 \mathbf{c}_v^{(l)} \quad (3.15)$$

where Φ and Ψ are the update values. For the top layer these update values can be computed

as follows:

$$\Phi_{u,i}^{(L)} = \zeta'(z) \cdot S_{u_i, v_j} \cdot \left(\mathbf{h}_{u,i}^{(L)} - \mathbf{h}_{v,j}^{(L)} \right) \odot s' \left(\mathbf{z}_{u,i}^{(L)} \right)$$

$$\Phi_{v,j}^{(L)} = \zeta'(z) \cdot S_{u_i, v_j} \cdot \left(\mathbf{h}_{v,j}^{(L)} - \mathbf{h}_{u,i}^{(L)} \right) \odot s' \left(\mathbf{z}_{v,j}^{(L)} \right)$$

$$\Psi_{u,i}^{(L)} = \delta_{ij} \left(\mathbf{h}_{u,i}^{(L)} - \mathbf{h}_{v,j}^{(L)} \right) \odot s' \left(\mathbf{z}_{u,i}^{(L)} \right)$$

$$\Psi_{v,j}^{(L)} = \delta_{ij} \left(\mathbf{h}_{v,j}^{(L)} - \mathbf{h}_{u,i}^{(L)} \right) \odot s' \left(\mathbf{z}_{v,j}^{(L)} \right)$$

where

$$z \triangleq 1 - S_{u_i, v_j} \left(\tau - d \left(\mathbf{h}_{u,i}^{(L)}, \mathbf{h}_{v,j}^{(L)} \right) \right)$$

$$\mathbf{z}_{u,i}^{(l)} \triangleq \mathbf{W}_u^{(l)\top} \mathbf{h}_{u,i}^{(l-1)} + \mathbf{c}_u^{(l)}$$

$$\mathbf{z}_{v,j}^{(l)} \triangleq \mathbf{W}_v^{(l)\top} \mathbf{h}_{v,j}^{(l-1)} + \mathbf{c}_v^{(l)}$$

Algorithm 1: DCML

Input: Training set $\{\mathbf{X}_u, \mathbf{X}_v\}$, number of layers $L + 1$, learning rate η , threshold τ , number of iterations T , parameters λ_1 and λ_2 , and convergence error ε .

Output: Network weight and bias parameters $\{\mathbf{W}_u^{(l)}, \mathbf{c}_u^{(l)}\}_{l=1}^L$ and $\{\mathbf{W}_v^{(l)}, \mathbf{c}_v^{(l)}\}_{l=1}^L$.

Step 1 (Initialization):

Initialize $\{\mathbf{W}_u^{(l)}, \mathbf{c}_u^{(l)}\}_{l=1}^L$ and $\{\mathbf{W}_v^{(l)}, \mathbf{c}_v^{(l)}\}_{l=1}^L$.

Step 2 (Optimization by back propagation):

for $t = 1, 2, \dots, T$ **do**

 Randomly select a sample pair $(\mathbf{x}_{u,i}, \mathbf{x}_{v,j}; S_{u_i v_j})$ from the training set.

 Set $\mathbf{h}_{u,i}^0 = \mathbf{x}_{u,i}$ and $\mathbf{h}_{v,j}^0 = \mathbf{x}_{v,j}$, respectively.

for $l = 1, 2, \dots, L$ **do**

 Compute $\mathbf{h}_{u,i}^{(l)}$ and $\mathbf{h}_{v,j}^{(l)}$ using the coupled deep network.

end

for $l = L, L - 1, \dots, 1$ **do**

 Compute the gradients using (3.12)-(3.15).

end

for $l = 1, 2, \dots, L$ **do**

 Update $\mathbf{W}_u^{(l)}, \mathbf{W}_v^{(l)}, \mathbf{c}_u^{(l)}$ and $\mathbf{c}_v^{(l)}$ using (3.16)-(3.17).

end

 Calculate H_r using (3.10).

 If $t > 1$ and $|H_t - H_{t-1}| < \varepsilon$, go to **Return**.

end

Return: $\{\mathbf{W}_u^{(l)}, \mathbf{c}_u^{(l)}\}_{l=1}^L$ and $\{\mathbf{W}_v^{(l)}, \mathbf{c}_v^{(l)}\}_{l=1}^L$.

For the remaining layers, $1 \leq l \leq L - 1$, Φ and Ψ can be obtained as:

$$\Phi_{u,i}^{(l)} = (\mathbf{W}_u^{(l+1)})^\top \Phi_{u,i}^{(l+1)} \odot s'(\mathbf{z}_{u,i}^{(l)})$$

$$\Phi_{v,j}^{(l)} = (\mathbf{W}_v^{(l+1)})^\top \Phi_{v,j}^{(l+1)} \odot s'(\mathbf{z}_{v,j}^{(l)})$$

$$\Psi_{u,i}^{(l)} = \left((\mathbf{W}_u^{(l+1)})^\top \Psi_{u,i}^{(l+1)} + \delta_{i,j} (\mathbf{h}_{u,i}^{(l)} - \mathbf{h}_{v,j}^{(l)}) \right) \odot s'(\mathbf{z}_{u,i}^{(l)})$$

$$\Psi_{v,j}^{(l)} = \left((\mathbf{W}_v^{(l+1)})^\top \Psi_{v,j}^{(l+1)} + \delta_{i,j} (\mathbf{h}_{v,j}^{(l)} - \mathbf{h}_{u,i}^{(l)}) \right) \odot s'(\mathbf{z}_{v,j}^{(l)})$$

where the operation “ $\odot$ ” denotes the element-wise multiplication.

We then apply gradient descent to update the network parameters $\mathbf{W}_u^{(l)}, \mathbf{W}_v^{(l)}, \mathbf{c}_u^{(l)}$ and

$\mathbf{c}_v^{(l)}$ of our coupled deep network:

$$\mathbf{W}_u^{(l)} = \mathbf{W}_u^{(l)} - \eta \frac{\partial H}{\partial \mathbf{W}_u^{(l)}}, \quad \mathbf{W}_v^{(l)} = \mathbf{W}_v^{(l)} - \eta \frac{\partial H}{\partial \mathbf{W}_v^{(l)}} \quad (3.16)$$

$$\mathbf{c}_u^{(l)} = \mathbf{c}_u^{(l)} - \eta \frac{\partial H}{\partial \mathbf{c}_u^{(l)}}, \quad \mathbf{c}_v^{(l)} = \mathbf{c}_v^{(l)} - \eta \frac{\partial H}{\partial \mathbf{c}_v^{(l)}} \quad (3.17)$$

where η is the learning rate in our gradient descent algorithm. **Algorithm 1** summarizes the detailed procedure of the proposed DCML method.

3.2.3 Implementation Details

Our deep network consists of several fully-connected layers of different dimensions, where the learning rate η , parameters λ_1 and λ_2 are set as 0.0001, 0.01 and 0.0001, respectively, for all experiments². The parameters $\mathbf{W}_u^{(l)}$ and $\mathbf{W}_v^{(l)}$ are initialized as $\mathbf{I} \in \mathbb{R}^{d^{(l)} \times d^{(l-1)}}$, which represents an identity matrix. The bias vectors $\mathbf{c}_u^{(l)}$ and $\mathbf{c}_v^{(l)}$ are initialized as zero vectors. For the activation function, we use the hyperbolic tangent (*tanh*) function. We set the layers to $L = 2$ for all experiments to prevent over-fitting. We performed empirical tests which show that networks with $L > 2$ are comparable with $L = 2$. It is expected that if more training pairs are used, a deeper network would be more effective [125].

During training, we randomly choose sample pairs and iteratively feed them to the network. For each epoch, the positive and negative pairs are of equal quantity. The training ends when the objective function value does not change within a certain threshold $\epsilon = 0.0001$.

3.3 Experiments

We conducted experiments on three different cross-modal matching applications which include text-image matching, tag-image retrieval, and heterogeneous face recognition on four

²We obtained these parameters by using the 10-fold cross-validation strategy.

widely used benchmark datasets to evaluate our proposed DCML method. The following describes the details of the experiments and results.

3.3.1 Datasets and Experimental Set-up

Wiki dataset [71]³: The dataset consists of 2866 image-text pairs where each pair consists of an image and its corresponding text article. Each image-text pair is annotated with a label from 10 semantic classes (i.e. sport, music, warfare, etc).

We evaluated our DCML using two image descriptors. First, each image is represented by a 128-dimensional SIFT descriptor provided by [71]. Second, we extracted a convolutional neural network (CNN) feature from a VGG network [126] pre-trained in the ImageNet dataset [104]. We extracted the features from the fc7 layer of the network and employed PCA to reduce it to 512 dimension⁴. For each text, we used the 10-dimensional features obtained from the Latent Dirichlet allocation (LDA) model provided by [71]. We randomly used 1300 pairs in our experiments for training, 130 pairs per class, and used the remaining 1566 pairs for testing.

Pascal VOC 2007 [127]⁵: The dataset contains 5011 image-text pairs for training and 4952 image-text pairs for testing. Each image-text pair is annotated from 20 categories (i.e. aeroplane, bottle, horse, sofa). Unlike the Wiki dataset which utilizes text information from articles, the Pascal VOC 2007 dataset only makes use of tag information.

We also evaluated our DCML using two set of descriptors to represent the images for the Pascal dataset. The first image feature representation is the concatenation of bag-of-visual-words (BOVW), GIST features and color histograms provided by [128], which led to a 776-dimensional feature vector. The second is the CNN feature which is extracted from

³<http://www.svcl.ucsd.edu/projects/crossmodal/>

⁴<http://www.vlfeat.org/matconvnet/models/imagenet-vgg-f.mat>.

⁵<http://host.robots.ox.ac.uk/pascal/VOC/voc2007/>

a similar setting as the Wiki dataset. The text feature used is a 399-dimensional absolute rank feature similar to [128]. We used the original train and test split provided but removing image-text pairs that have multiple labels. This resulted to 2808 training pairs and 2841 test pairs.

NUS-WIDE dataset [129]⁶: This dataset contains approximately 270,000 images with labels from 81 categories. Following the same setting in [14], we selected 10 categories having the largest quantity and extracted 1000 image-tag pairs for each category. This led to a subset of 10,000 pairs. We randomly split this subset into 8000, 1000 and 1000 pairs for training, validation, and testing, respectively. Each set contains an equal number of samples for each category.

We also evaluated our DCML using two types of image features. First, we used the combined local feature provided in [129], which is represented as a 1134-dimension feature vector. Second, we used a CNN feature which is reduced to 512 dimensions by PCA.

Different from the PASCAL VOC 2007, tag information provided in the NUS-WIDE is more representative and used more words to describe an image. Each tag information is represented by a 1000-dimensional bag-of-words model same as that of [129]. We used the mAP and top 20% percentage measures as in [14] to evaluate and compare the performance of different methods.

CASIA VIS-NIR (version 2.0) [130]⁷: There are 275 subjects in the CASIA VIS-NIR (version 2.0) dataset. For each subject, there are 1-22 VIS and 5-50 NIR images. All face images in both the visual and near-infrared face sets were aligned and cropped into 128×128 pixels according to the eye coordinates provided. Figure 3.3 shows several cropped example images from the CASIA VIS-NIR (version 2.0) dataset.

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

⁷<http://www.cbsr.ia.ac.cn/english/NIR-VIS-2.0-Database.html>

Figure 3.3: Sample images from the CASIA VIS-NIR (version 2.0) dataset. Face images in the first and second rows are visible and near-infrared images, respectively. Face images in each column are the same person.

For each face image, we divided it into 16×16 non-overlapping blocks and extracted the SIFT feature from each block. The SIFT features from all blocks within the same image were then concatenated into a longer feature vector. Finally, we applied WPCA to learn a projection matrix to map each concatenated feature into a 2000-dimensional feature vector.

We followed the standard protocol of the CASIA VIS-NIR (version 2.0) dataset, where VIS images were used as the gallery set and the NIR images were used as the probe set.

3.3.2 Experimental Results

3.3.2.1 Text-Image Retrieval on the Wiki dataset

We compared our DCML with eight cross-modal matching methods: CCA [91], PLS [123], MvDA [94], GMLDA [92], GMMFA [92], SCM [71], KCCA [11] and LCFS [73]. CCA and PLS are cross-modal models which reduce the modality gap via subspace analysis using pairwise information. MvDA, GMLDA, and GMFA learn a single common discriminative representation for two modalities by using the Fisher criterion. KCCA uses kernels and finds a common space between two modalities based on the pairwise correlation information⁸. SCM obtains representations for two modalities to maximize their correlations

⁸In our experiments, we made use of Gaussian Kernels which provided the best results among different kernels.

Figure 3.4: Precision-scope curve for different coupled metric learning methods from the Wiki Experiment using SIFT image features, where scope is the number of retrieved instances.

and transform them in a semantic space. LCFS learns a coupled projection to transform each modal to a common subspace defined by class labels and low-rank constraint. Among these methods, only KCCA performs nonlinear representations while other methods perform linear regression or projection. The source codes of these methods are provided by the respective authors and we carefully tuned the parameters of different methods to obtain the best results for a fair comparison. We repeated our experiments 10 times and took the average as the final matching results.

In our DCML method, we trained our model using three layers ($L = 2$), where the feature dimensions for these layers were set as $[128 \rightarrow 50 \rightarrow 20]$ and $[10 \rightarrow 50 \rightarrow 20]$ for the SIFT image feature and text feature, respectively, and $[512 \rightarrow 100 \rightarrow 50]$ and $[10 \rightarrow 100 \rightarrow 50]$ when the CNN image feature is employed. For experiments using SIFT features, we also compared our method with CDL [95] which performs dictionary learning, LGCFL [74] which uses a variant of subspace learning method to reduce the modality gap, and JF-SSL [93] which performs linear regressions and multimodal graph regularization for feature selection. For experiments with the CNN features, we also compared our method

Table 3.1: mAP (%) performance comparison of our DCML and state-of-the-art cross-modal matching methods on the Wiki dataset using SIFT features.

Method	Image query	Text query	Average
PLS [123]	21.49	17.07	19.28
CCA [91]	24.60	19.18	21.89
GMMFA [92]	24.20	17.97	21.09
GMLDA [92]	18.08	13.83	15.96
MvDA [94]	15.99	13.26	14.63
SCM [71]	23.84	22.23	23.04
KCCA [11]	26.85	21.34	24.10
LCFS [73]	26.83	21.77	24.30
CDL* [95]	27.76	23.11	25.44
LGCFL* [74]	27.90	21.77	24.80
JFSSL* [93]	30.63	22.75	26.69
DCML	35.04	25.55	30.03

*The results are from the original papers.

with JFSSL [93], LRBS [97] and RE-DNN [96]. LRBS [97] uses bilinear transformation and RE-DNN uses deep networks for reducing the modality gap, respectively. Here, CDL and RE-DNN are nonlinear representations. Different from our deep network, RE-DNN performs multi-modal learning by using a Euclidean distance criterion with two stages of

Table 3.2: mAP (%) performance comparison of our DCML and state-of-the-art cross-modal matching methods on the Wiki dataset using CNN features.

Method	Image query	Text query	Average
PLS [123]	31.28	26.45	28.86
CCA [91]	35.42	32.50	33.96
GMMFA [92]	27.76	12.85	20.30
GMLDA [92]	18.01	14.09	16.05
MvDA [94]	16.17	12.00	14.09
SCM [71]	38.74	37.03	37.88
KCCA [11]	37.34	34.61	35.98
LCFS [73]	39.39	38.09	38.74
LRBS* [97]	44.41	37.70	41.06
RE-DNN* [96]	34.04	35.26	34.65
JFSSL* [93]	42.79	39.57	41.18
DCML	55.36	53.81	54.59

*The results are from the original papers.

Figure 3.5: Sample results by DCML method for (a) text to image and (b) image to text search, respectively, in the Wiki dataset. The first two columns in each figure are the ground truth text-image pairs provided by the dataset. The next four columns in each figure are the top retrieved samples.

intra-modal pre-training and inter-modal full training, while our method performed direct full training with cross-modal specific criterions. Tables 3.1 and 3.2 show the mean average precision (mAP) of our DCML and other cross-modal methods on the Wiki image-text dataset when the hand-crafted (SIFT) and CNN features were used, respectively. We observe that our DCML achieves the best performance for both CNN and SIFT features. It is important to note that it is difficult to perform an exact comparison due to certain experimental differences. Particularly, LRBS and RE-DNN employed 200 and 20 LDA topics as text features respectively, while LRBS and RE-DNN used different CNN models for their experiments. In addition, for LGCFL, LRBS, and Re-DNN, the number of the train

Table 3.3: mAP (%) performance comparison of our DCML and state-of-the-art cross-modal matching methods on the PASCAL VOC 2007 dataset using hand-crafted image features.

Method	Image query	Text query	Average
PLS [123]	17.40	14.01	15.71
CCA [91]	14.89	14.28	14.59
GMMFA [92]	29.65	26.52	28.08
GMLDA [92]	30.58	25.47	28.02
MvDA [94]	12.30	9.15	10.72
SCM [71]	23.08	19.38	21.23
KCCA [11]	29.24	24.66	26.95
LCFS [73]	41.37	33.48	37.43
CDL* [95]	37.41	29.44	33.42
LGCFL* [74]	40.10	32.00	36.00
JFSSL* [93]	36.07	28.01	32.04
DCML	44.49	36.26	40.38

*The results are from the original papers.

and test splits are also different from the one we followed in [73]. We see that our DCML outperforms other compared cross-modal matching methods by an average of 13% and 3% when the CNN and SIFT features were used, respectively. Figure 3.4 shows the precision-scope curves of different coupled metric learning methods with the SIFT feature on the Wiki image-text dataset. The scope refers to the number of top-ranked samples. Similarly, the precision-scope evaluation is consistent with the mAP for both the image and text query tasks, where our DCML outperformed the other compared methods significantly.

Figure 3.5 shows several cross-modal matching results by using our DCML method. As can be seen, given the query text (image), the retrieved images (texts) are correlated, which means the heterogeneous gap between image and text features can be effectively reduced in the learned semantic space.

Table 3.4: mAP (%) performance comparison of our DCML and state-of-the-art cross-modal matching methods on the PASCAL VOC 2007 dataset using CNN features.

Method	Image query	Text query	Average
PLS [123]	49.47	52.53	51.00
CCA [91]	30.49	31.28	30.89
GMMFA [92]	64.73	68.86	66.80
GMLDA [92]	67.19	72.47	69.83
MvDA [94]	70.17	71.71	70.94
SCM [71]	69.09	68.52	68.81
KCCA [11]	67.15	67.66	67.41
LCFS [73]	70.94	74.74	72.84
LRBS* [97]	65.10	68.69	66.90
DCML	73.77	75.01	74.39

*The results are from the original papers.

Figure 3.6: Precision-scope curve for different coupled metric learning methods from the PASCAL VOC 2007 experiment using BOVW-GIST-COLOR image features, where scope is the number of retrieved instances for the Pascal VOC 2007 dataset.

3.3.2.2 Tag-Image Retrieval on the Pascal VOC 2007 Dataset

In our DCML method, we trained our model using three layers on the training set and the feature dimension for these layers were set as $[776 \rightarrow 200 \rightarrow 100]$ and $[399 \rightarrow 200 \rightarrow 100]$ for the image and text network when the BOVW-GIST-COLOR image feature is used, respectively. When the CNN image feature is used, the network layers are set to $[512 \rightarrow 300 \rightarrow 50]$ and $[399 \rightarrow 300 \rightarrow 50]$ for image and text network, respectively. We also compared our

Figure 3.7: Sample results by DCML method for (a) text to image and (b) image to text search, respectively, in the Pascal VOC 2007 dataset. The first two columns in each figure are the ground truth text-image pairs provided by the dataset. The next four columns in each figure are the top retrieved samples.

method with JFSSL, CDL, LGCFL, and LRBS. Tables 3.3 and 3.4 show the mean average precision (mAP) of our DCML and other cross-modal methods on the Pascal dataset with image and text query, respectively. We can see that our DCML outperforms other compared cross-modal matching methods by approximately 2% and 4% when the CNN and hand-crafted features were used, respectively. As expected, the CNN features generally achieved better performance because of its strong representation. Figure 3.6 shows the precision-scope curves of different coupled metric learning methods using hand-crafted features. We see that our DCML still outperforms the other compared methods in both cross-modal matching tasks.

Figure 3.7 shows several cross-modal matching results by using our DCML method. Similarly, the query image (text) is consistent in semantic with its top retrieved texts (images). Some incorrectly retrieved data are due to similar tag words but contain two different meanings. For example, the second row of Figure 3.6(a) composes of glasses and cloth as tag features, which resulted in the retrieved images having eye-glasses and bed sheets instead of people eating with wine-glasses and tablecloth. In this regard, improvements in the text feature representation such that they are invariant to these types of conflict are desired and can be future work.

Similarly, we also show in Tables 3.3 and 3.4 the results of CDL, LGCFL, and LRBS taken from the authors' papers. As can be seen, our method still showed the best performance. But then again, it is still difficult to directly compare the results due to differences of feature representations. Unlike our hand-crafted DCML implementation, LRBS and CDL performed the experiment using only 512-dimensional GIST features. Also differently, the LRBS extracted CNN features from the DeCAF⁹ model

3.3.2.3 Tag-Image Retrieval on the NUS-WIDE Dataset

We trained our DCML model with three layers, where the feature dimensions for these layers were set as [512→200→100] for both image and text networks. We applied PCA to map each sample into a 512-dimensional feature vector.

We compared our DCML with eight state-of-the-art cross-modal retrieval methods. Table 3.5 shows the mean average precision (mAP) and top 20% criteria on the image-text and text-image experiments. We see that our DCML method outperforms all the other compared methods including deep models which were proposed for cross-modal retrieval [13, 14, 131]. Our method with CNN features is better by 0.5% and 2% in the mAP

⁹<http://daggerfs.com/>

Table 3.5: mAP (%) and top 20% performance on the NUS-WIDE dataset.

Method	mAP			Top 20%		
	Image query	Text query	Ave.	Image query	Text query	Ave.
CCA-AE [131]	32.6	26.8	23.4	37.0	33.8	35.4
CCA-Cross-AE [131]	13.7	34.4	27.2	29.0	47.7	38.4
CCA-Full-AE [131]	14.8	24.2	24.2	37.1	38.2	37.7
Bimodal AE [13]	25.0	29.7	27.4	30.2	35.4	32.8
Bimodal DBN [13]	17.3	20.3	18.8	25.3	27.0	26.2
Corr-AE [14]	31.9	37.5	34.7	47.1	53.5	50.3
Corr-Cross-AE [14]	34.9	34.8	34.9	53.1	59.7	56.4
Corr-Full-AE [14]	33.1	37.9	35.5	49.6	56.5	53.5
DCML	38.5	40.5	39.5	61.5	62.5	62.0
CCA-CNN	69.5	67.4	68.4	65.5	64.8	65.2
SCM-CNN	82.7	77.8	80.2	68.2	66.9	67.6
KCCA-CNN	81.7	80.6	81.1	71.1	70.5	70.8
LCFS-CNN	85.1	80.3	82.7	72.7	72.8	72.8
DCML-CNN	85.6	82.6	84.1	76.4	80.4	78.4

and 4% and 8% in the top20% evaluation metric for the image-text and text-image task, respectively.

Aside from retrieval, we can also perform tag annotation. Given an image, we can apply DCML's image network and acquire the most similar BoW tag features from a pre-

Figure 3.8: Performance curves of different coupled metric learning methods on the CASIA VIS-NIR (version 2) dataset .

Table 3.6: Sample Tag Annotations obtained from the NUS-WIDE Dataset.

Image	NUS-WIDE Groundtruth Tags	DCML Tags	LCFS Tags
morning trees red sky cloud sun mountain art nature beautiful beauty silhouette night clouds sunrise wonderful garden dark landscape dawn lights spring bush flora pretty shadows darkness native oz farm country sam-sung australia	'sky' 'sun-set' 'sun' 'freedom' 'balcony' 'dramatic'	'sky' 'blue' 'clouds' 'green' 'california' 'field' 'farm' 'hills' 'sunny' 'valley' 'sheep' 'land' 'meadow'
race racecar lola racing autoracing daredevil plm alms roadatlanta carracing americanlemansseries petitemans cytosport memogidley gregpickett klausgraf theesses	'race' 'racing'	'singapore' 'golf'

computed gallery set. Table 3.6 shows the tags suggestions of our DCML method in the NUS-WIDE experiment. We compared our DCML tags with the LCFS tags which showed the second best performance for the retrieval experiment. As can be seen, our suggested tags are more representative and are more similar with the human-annotated tags compared to the LCFS. Nevertheless, we could improve the performance for tag annotations task further by exploiting word vector representations and topic modeling for more representative text features but will leave this to future work [132–134].

3.3.2.4 Heterogeneous Face Recognition

We trained our deep model using three layers on the “View 1” subset and the dimensions for these layers were set as $[2000 \rightarrow 1000 \rightarrow 500]$, respectively. All other parameters of our model were the same as those used in the previous experiments.

Table 3.7: Recognition performance comparison of our DCML method and state-of-the-art coupled metric learning methods on the CASIA VIS-NIR (version 2.0) dataset.

Method	Rank-one	VR1	VR2
CCA [91]	76.08 ± 1.86	43.47	64.53
PLS [123]	33.90 ± 2.99	9.30	30.61
MvDA [94]	42.90 ± 3.60	14.99	39.30
GMLDA [92]	43.61 ± 3.06	13.57	39.17
GMMFA [92]	66.65 ± 2.03	30.32	62.83
KCCA [11]	67.20 ± 3.03	29.77	58.89
DCML	82.19 ± 1.11	46.71	66.85

Table 3.7 shows the recognition performance of different coupled metric learning methods on the CASIA VIS-NIR (version 2.0) dataset, where three different evaluation measures including the rank-one recognition rate, the verification rate at 0.1% false acceptance rate (VR1), and the verification rate at 1.0% false acceptance rate (VR2) were evaluated and compared. Figure 3.8 shows the receiver operating characteristic (ROC) curve and the cumulative match characteristic (CMC) curve of different coupled metric learning methods. We clearly see that our DCML consistently outperforms the current state-of-the-art coupled metric learning methods, and the minimal improvement is 6.11% in terms of the rank-one recognition rate.

We also compared our DCML with state-of-the-art heterogeneous face recognition methods. Table 3.8 shows the performance of different methods. FaceVACS is a commercial face recognition software which has achieved top performance in the National Institute of Standards and Technology (NIST) Multiple Biometrics Evaluation. [130], the baseline method for this dataset, performs PCA holistically on the raw image with augmentation from facial symmetry and a hetero-component analysis to remove the differences between NIR and VIS images. [136] extracts dense HOG features with dimensionality reduction using linear discriminant analysis and similarity measure using the cosine distance. [137] learns a mapping function using dictionary learning to reconstruct facial images between

Table 3.8: Rank-one Recognition rate comparison of our DCML method and state-of-the-art heterogeneous face recognition methods on the CASIA VIS-NIR (version 2.0) dataset.

Method	Rank-one	Year
FaceVACS [135]	58.56 ± 1.19	2012
PCA+Sym+HCA [130]	23.70 ± 1.89	2013
HOG+LDA+Cosine [136]	73.28 ± 1.10	2014
Reconstruction+UDP(DLBP) [137]	78.46 ± 1.67	2015
Gabor+RBM+Remove 11PCs [138]	86.16 ± 0.98	2015
CDFL (s=3) [139]	71.50 ± 1.40	2015
DCML	82.19 ± 1.11	

modalities. They use dense LBP as its handcrafted feature. [138] extracts local Gabor features *from facial points* and passes them through Restricted Boltzmann Machines (RBMs) to learn a shared representation reduced through PCA. Finally, CDFL [139] extracts its own local features from raw images and learns a shared representation in a feature-level using subspace analysis.

As observed, our method is very competitive and can achieve comparable performance with the state-of-the-art. While our method is worse than the method in [138] which yields the best performance, it extracts more powerful features for cross-modal face matching. Particularly, it requires a pre-processing step to locate the facial points before feature extraction. On the other hand, other methods (like ours) collect dense features by dividing the facial image into a grid. Since features from facial points are more representative, this gives [138] a boost in performance. However, dense feature extraction done by methods such as ours would be much faster. Additionally, [138] is too dataset-specific for it has to discard specifically 11 PCs (principal components) to attain optimal results. This heuristic step may not work for other datasets and is not a general solution. By not discarding PCs, [138] actually performs 2% less. It is also expected that extracting features from facial landmarks would lead to better performance for our DCML method which may achieve better or similar performance compared with [138].

Table 3.9: Rank-one recognition rate on the CASIA VIS-NIR (version 2.0) dataset and average mAP on the Wiki, PASCAL VOC 2007, NUS-WIDE dataset using different DCML methods.

Dataset	DCML1	DCML2	DCML
Wiki	26.99	29.52	30.03
Wiki (CNN)	45.21	53.14	54.59
PASCAL VOC 2007	18.97	39.08	40.38
PASCAL VOC 2007 (CNN)	29.39	72.27	74.39
NUS-WIDE	38.66	39.00	39.50
NUS-WIDE (CNN)	79.35	82.59	84.00
CASIA VIS-NIR	79.80	81.76	82.19

3.3.3 Further Analysis

3.3.3.1 Influence Analysis of Different Components in DCML

To investigate the contributions of different terms in our DCML model, we developed two variations of our method: DCML1 and DCML2. For DCML1, the discriminative part H_1 is removed. For DCML2, the correlation part H_2 is removed. The optimization procedure would be similar to our original DCML method which performs stochastic gradient descent. Table 3.9 shows the performance of different DCML methods on different datasets. We see that both the discriminative and correlation terms contribute to the final recognition rate. We also see the DCML2 consistently performs better than DCML1 which means the discriminative part H_1 contributes more to the overall recognition rate of our DCML method.

3.3.3.2 Convergence Analysis

We evaluated the convergence of our DCML method. Figure 3.9 shows the objective function value of our DCML at various iteration numbers on the CASIA VIS-NIR (version 2.0) dataset. We can observe that the proposed DCML method converges in $40 \sim 45$ iterations.

Figure 3.9: Convergence curve of DCML on the CASIA VIS-NIR (version 2) dataset.

3.3.3.3 Computation Time

Lastly, we investigated the computational time of our DCML and compared it with those of existing coupled metric learning methods. Our computer is configured with a 3.40 GHz CPU and 24.0 GB RAM. Table 3.10 shows the computational time for training and testing on the CASIA VIS-NIR (version 2.0) dataset. We can observe that the computational time of our method for training is relatively larger to other coupled metric learning methods, while the testing time is comparable.

3.3.4 Discussion

The above experimental results provide the following four key observations:

- (i) Our DCML method achieves strong performance and beats the state-of-the-art cross-modal methods in three image-text/tag retrieval experiments (Wiki, PASCAL VOC 2007, and NUS-WIDE). This is because our approach trained a deep network to

Table 3.10: Computational time (seconds) used by different coupled metric learning methods on the CASIA VIS-NIR (version 2) dataset.

Method	Training	Testing
CCA [91]	0.30	0.01
PLS [123]	208.50	0.08
MvDA [94]	0.27	0.03
GMLDA [92]	0.59	0.01
GMMFA [92]	0.41	0.01
KCCA [11]	14.10	6.24
DCML	22.17	0.02

model real-world data by exploiting both the discriminative and nonlinear information, simultaneously. Moreover, our DCML can also be extended for the tag annotation task.

- (ii) For text/tag-image retrieval experiments, we have used different types of image features in our experiments and observed that the CNN features provided a boost in the retrieval performance. This also shows that our DCML method is flexible for feature representations with varying dimensions.
- (iii) Our DCML method also achieves competitive performance with heterogeneous face recognition experiment, which shows that our model can be used for different cross-modal applications.
- (iv) We have investigated the contribution of each term in our DCML and have shown that both the discriminative and correlation terms contributed to the overall performance. Particularly, the discriminative part of our objective formulation has a larger contribution.

3.4 Chapter Summary

In this chapter, we have proposed a new deep coupled metric learning (DCML) method for cross-modal matching. Our method trains two deep neural networks to learn two sets of hierarchical nonlinear transformations to exploit both discriminative information and reduce the modality gap, which significantly improves the performance of different cross-modal matching applications. Experimental results on four cross-modal datasets have clearly demonstrated the effectiveness of the proposed method. The work presented in this chapter has been published in [29].

Chapter 4

Cross-Modal Discrete Hashing

In this chapter, we present a new cross-modal discrete hashing (CMDH) approach to learn compact binary codes for cross-modal multimedia search. Unlike most existing cross-modal hashing methods which usually relax the optimization objective function to obtain hash codes, we develop a discrete optimization framework to jointly learn binary codes and a series of hashing functions for each modality, so that the performance drop due to the inferior optimization techniques can be minimized. Specifically, we present two cross-modal hashing algorithms called shallow CMDH and deep CMDH under the proposed framework, which performs shallow linear/nonlinear and deep nonlinear mappings to learn binary codes, respectively. Unlike existing cross-modal hashing methods which maximize the corrections of hash codes from different modalities, our CMDH learns a set of shared binary codes for samples captured from different modalities, so that the modality gap can be effectively reduced in cross-modal multimedia retrieval. Experimental results on four benchmark datasets clearly show that our methods achieve results competitive with the state-of-the-art cross-modal hashing methods.

4.1 Introduction

Recently, several cross-modal hashing methods [18, 81, 82] have been proposed which have shown promising performance. While these cross-modal hashing methods have achieved encouraging performance, most of them relax the binary constraint in their optimization objective functions, which may lead to loss of information because of the accumulated quantization error [8, 16, 44, 84].

We propose a new cross-modal discrete hashing (CMDH) method for cross-modality search. Unlike several existing methods which relax the optimization objective function to address the NP-hard problem, we perform a discrete optimization method that jointly learns binary codes and a series of hashing functions. In our approach, we do not relax the binary constraint to avoid approximated solutions, so that our learned hash codes are more representative. In addition, unlike other methods which learn separate hash codes for each modality and maximize the correlations during training, our CMDH method learns a shared set of binary code for samples from different modalities based on their semantic affinities, so that the modality gap is implicitly reduced.

Specifically, we present two cross-modal hashing algorithms called shallow CMDH and deep CMDH. For shallow CMDH method, we present CMDH-linear and CMDH-kernel which perform linear and nonlinear mappings to learn binary codes, respectively. To improve the flexibility of our shallow CMDH for different scenarios, we further extend our shallow CMDH to unsupervised CMDH (unCMDH) and multi-modal discrete hashing (MMDH) which learn hash codes for training data without label information and with multi-modal labelled data, respectively. For our deep CMDH method, we design a deep neural network to learn multiple pairs of hierarchical nonlinear transformations, under which the nonlinear characteristics of samples can be well exploited and the modality gap is well reduced.

4.2 Cross-Modal Discrete Hashing

In this section, we first present the proposed CMDH framework in shallow form, particularly, CMDH-linear and CMDH-kernel which perform linear and nonlinear kernel mapping to learn the hashing functions, respectively. We then present two extensions called unsupervised CMDH (unCMDH) and multi-modal discrete hashing (MMDH) which learn hash codes for training data without label information and with multi-modal labelled data, respectively. Finally, we present our deep CMDH which learns cross-modal hashing functions using a deep framework.

4.2.1 Shallow CMDH

Figure 4.1 shows the basic idea of our proposed shallow CMDH approach which consists of two iterative steps to train the hashing functions and learn a unified binary code. The trained hashing functions are used to obtain the new binary codes of the query samples during the testing stage.

Let $\mathbf{X}_u = [\mathbf{x}_{u,1}, \dots, \mathbf{x}_{u,N}] \in \mathbb{R}^{d_u \times N}$ and $\mathbf{X}_v = [\mathbf{x}_{v,1}, \dots, \mathbf{x}_{v,N}] \in \mathbb{R}^{d_v \times N}$ be the training set of two different modalities U and V , respectively, d_u and d_v are its corresponding feature dimension (d_u needs not to be equal to d_v), and N is the number of training samples. Our shallow CMDH aims to learn a shared binary feature vector, $\mathbf{B} = [\mathbf{b}_1, \dots, \mathbf{b}_N] \in \{-1, 1\}^{K \times N}$ of length K and a series of hashing functions for each modality simultaneously. To achieve this, we formulate the following optimization objective:

$$\begin{aligned}
 \min_{\mathbf{b}, \mathbf{h}_u, \mathbf{h}_v} J &= J_1 + \eta J_2 \\
 &= \sum_{i,j=1, i \neq j}^N \|\mathbf{b}_i - \mathbf{b}_j\|_2^2 A_{ij} + \eta \sum_{i=1}^N (\|\mathbf{b}_i - \mathbf{h}_{u,i}\|_2^2 + \|\mathbf{b}_i - \mathbf{h}_{v,i}\|_2^2) \quad (4.1) \\
 \text{s.t.} \quad &\mathbf{b}_i \in \{-1, 1\}^K
 \end{aligned}$$

Figure 4.1: Basic idea of our proposed shallow CMDH approach. During training, our optimization objective consists of two steps which are performed iteratively. The first step is the H-step which learns the hashing functions with fixed binary codes, and the second step is the B-step which learns the unified binary code. During testing, we use the learned hashing functions to obtain the binary codes for large-scale cross-modal multimedia retrieval.

where $\mathbf{h}_{u,i}, \mathbf{h}_{v,i} \in \mathbb{R}^K$ is a continuous variable based on embedding function f with parameter θ_u and θ_v as follows:

$$\mathbf{h}_{u,i} = f(\mathbf{x}_{u,i}, \theta_u) \quad (4.2)$$

$$\mathbf{h}_{v,i} = f(\mathbf{x}_{v,i}, \theta_v) \quad (4.3)$$

and $\mathbf{b}_i$ and $\mathbf{b}_j$ are the shared binary vectors to be learned for the i th and j th cross-modal data, respectively. The embedding function, $f(\cdot)$, can be any linear or nonlinear mapping function.

In (4.1), J_1 ensures the hamming distance of binary code pairs that are semantically similar are minimized, J_2 ensures the quantization loss between binary codes and the corresponding learned real-valued codes in (4.2) and (4.3) are minimized, and η is a constant variable that balances J_1 and J_2 . A_{ij} is an affinity matrix representing the semantic relationship between $\mathbf{x}_i$ and $\mathbf{x}_j$, which can be obtained from the cosine similarity of the label vector with C classes as follows:

$$A_{ij} = \frac{\langle \mathbf{y}_i, \mathbf{y}_j \rangle}{\|\mathbf{y}_i\| \cdot \|\mathbf{y}_j\|} \quad (4.4)$$

where $\mathbf{y}_i = [y_{i,1}, y_{i,2}, \dots, y_{i,C}]$ is the label information for the i th where $y_{i,c} = 1$ if it belongs to the c th class.

Given $\mathbf{H}_* = [\mathbf{h}_{*,1}, \dots, \mathbf{h}_{*,N}] \in \mathbb{R}^{K \times N^1}$ and a normalized affinity matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, we re-write (4.1) into a matrix form as follows:

$$\begin{aligned} \min_{\mathbf{B}, \mathbf{H}_u, \mathbf{H}_v} J &= -\text{tr}(\mathbf{B}^\top \mathbf{A} \mathbf{B}) + \eta(\|\mathbf{B} - \mathbf{H}_u\|_F^2 + \|\mathbf{B} - \mathbf{H}_v\|_F^2) \\ \text{s.t.} \quad &\mathbf{B} \in \{-1, 1\}^{K \times N} \end{aligned} \quad (4.5)$$

Iterative Optimization: The optimization problem in (4.5) is non-convex due to the binary constraints, which makes it difficult to solve. However, it can be addressed using an iterative approach where we solve one variable while fixing the others, alternatively. The following presents the detailed procedure:

Binary Code Step (B-Step): We solve for $\mathbf{B}$ by fixing $\mathbf{H}_u$ and $\mathbf{H}_v$. By fixing $\mathbf{H}_*$, we can obtain the following discrete optimization problem:

$$\begin{aligned} \min_{\mathbf{B}} J(\mathbf{B}) &= -\text{tr}(\mathbf{B}^\top \mathbf{A} \mathbf{B}) + \eta(\|\mathbf{B} - \mathbf{H}_u\|_F^2 + \|\mathbf{B} - \mathbf{H}_v\|_F^2) \\ \text{s.t.} \quad &\mathbf{B} \in \{-1, 1\}^{K \times N} \end{aligned} \quad (4.6)$$

¹For ease of presentation, we represent $*$ = $\{u, v\}$ and $\tilde{*}$ = $\{v, u\}$

Algorithm 2: Shallow CMDH

Input: Training set $\mathbf{X}_u$ and $\mathbf{X}_v$, affinity matrix $\mathbf{A}$.

Output: Parameters θ_u, θ_v .

Step 1 (Initialization):

1.1. Initialize $\mathbf{B}^{(1)}$ randomly

1.2. Initialize θ_u and θ_v according to $f(\cdot)$

Step 2 (Optimization):

for $t = 1, 2, \dots, T$ **do**

 2.1. Solve $\mathbf{B}^{(t+1)}$ using (4.8);

 2.2. Solve θ_u based on (4.10);

 2.3. Solve θ_v based on (4.10).

 If $t > 1$ and $|J_t - J_{t-1}| < \varepsilon$, go to **Return**.

end

Return: θ_u, θ_v .

By expanding the above equation and removing all constant terms, we can re-write (4.6) as follows:

$$\begin{aligned} \min_{\mathbf{B}} J(\mathbf{B}) &= -\text{tr}(\mathbf{B}^\top \mathbf{A} \mathbf{B}) - \eta \mathbf{B}(\mathbf{H}_u + \mathbf{H}_v) \\ \text{s.t.} \quad &\mathbf{B} \in \{-1, 1\}^{K \times N} \end{aligned} \quad (4.7)$$

This leads to a binary quadratic problem which can be solved through a linearization technique according to [16]. We obtain a closed-form solution as follows:

$$\mathbf{B}^{(t+1)} = \text{sgn}(2\mathbf{A}\mathbf{B}^{(t)} + \eta(\mathbf{H}_u + \mathbf{H}_v)) \quad (4.8)$$

where $\mathbf{B}^{(t)}$ is the current binary matrix at iteration t , and $\mathbf{B}^{(t+1)}$ is the new binary matrix that is used for the H-step ($\mathbf{B} = \mathbf{B}^{(t+1)}$).

Hashing Function Step (H-Step): We solve for $\mathbf{H}_*$ by fixing $\mathbf{B}$ and $\mathbf{H}_*$. The discrete optimization problem can be re-written as:

$$\min_{\mathbf{H}_*} J(\mathbf{H}_*) = \|\mathbf{B} - \mathbf{H}_*\|_F^2 \quad (4.9)$$

which can be further rewritten as follows in terms of f and θ_* .

$$\min_{\theta_*} J(\theta_*) = \|\mathbf{B} - f(\mathbf{X}_*, \theta_*)\|_F^2 \quad (4.10)$$

The solution to (4.10) varies depending on $f(\cdot)$ which can be any representative linear or nonlinear embedding function. **Algorithm 2** shows the general optimization procedure of our shallow CMDH approach.

Out-of-Sample Extension: In the training stage of our shallow CMDH, we learn shared binary codes for cross-modal data, but we also need to obtain modality-specific hashing functions for out-of-sample data points during testing. To obtain the binary codes for each query sample, we ensure that the quantization loss between binary codes and the corresponding learned real-valued codes in each modality is minimized as follows:

$$\begin{aligned} \min_{\mathbf{b}_{*,q}} J_2(\mathbf{b}_{*,q}) &= \|\mathbf{b}_{*,q} - \mathbf{h}_{*,q}\|_2^2 \\ \text{s.t.} \quad &\mathbf{b}_{*,q} \in \{-1, 1\}^K \end{aligned} \quad (4.11)$$

Similar to the B-step optimization problem, we obtain a closed-form solution for each query sample, $\mathbf{x}_{*,q}$, as follows:

$$\mathbf{b}_{u,q} = \text{sgn}(\mathbf{h}_{u,q}) \quad (4.12)$$

$$\mathbf{b}_{v,q} = \text{sgn}(\mathbf{h}_{v,q}) \quad (4.13)$$

Relation to Discrete Graph Hashing (DGH) The DGH's [16] formulation can be written as follows:

$$\begin{aligned} \max_{\mathbf{B}, \mathbf{H}} &= \text{tr}(\mathbf{B}^\top \mathbf{A} \mathbf{B}) + \rho \text{tr}(\mathbf{B}^\top \mathbf{H}) \\ \text{s.t.} \quad &\mathbf{B} \in \{1, -1\}^{K \times N}, \mathbf{H} \in \mathbb{R}^{K \times N} \\ &\mathbf{H} \mathbf{1} = 0, \mathbf{H} \mathbf{H}^\top = N \mathbf{I}^{K \times K} \end{aligned} \quad (4.14)$$

where $\mathbf{A}$ is the affinity matrix based on the anchor graph Laplacian, $\mathbf{L} = \mathbf{I}_{N \times N} - \mathbf{A}$. $\mathbf{H}$ is a continuous variable to soften the constraints $\mathbf{B}\mathbf{1} = 0$ and $\mathbf{B}\mathbf{B}^\top = N\mathbf{I}_{K \times K}$. In this formulation, the distance between $\mathbf{B}$ and $\mathbf{H}$ is minimized and controlled by a factor ρ .

While our method may have some similarities in the mathematical formulation with the Discrete Graph Hashing (DGH) work, there are three key differences of our CMDH method compared to DGH. First, our CMDH addresses the problem of cross-modal/multi-modal hashing instead of single-modal hashing by learning a unified binary code for two modalities. Hence, we have investigated a cross-modal hashing implementation based on discrete optimization. Second, unlike DGH which only extracts the affinity matrix from anchors based on distances of the samples, our method performs supervised learning, where we exploit the label information of the database in order to obtain a similarity matrix. By doing so, we obtain more representative shared binary codes. Lastly, in DGH, $\mathbf{H}$ is defined as a continuous variable solved through singular value decomposition (SVD), in which a suboptimal solution is derived. In contrast, our method defines $\mathbf{H}$ as the output of an embedding function (linear or kernel) given a feature input. Hence, we perform an iterative and alternative optimization procedure to solve the function parameters accordingly. By doing so, we represent $\mathbf{H}$ in a general way suitable for unseen data on the assumption that $\mathbf{H}$ is modelled by a defined parametric function. Our general formulation can then be extended to other embedding functions such as boosting, mixture of Gaussians, or neural network models.

4.2.1.1 CMDH-linear

We develop a CMDH-linear method by using a linear mapping, $\mathbf{W}_* \in \mathbb{R}^{K \times d_*}$, to project each training sample $\mathbf{x}_*$ as follows:

$$f_{lin}(\mathbf{x}_*, \mathbf{W}_*) = \mathbf{W}_*^\top \mathbf{x}_* \quad (4.15)$$

Algorithm 3: CMDH - *linear***Input:** Training set $\mathbf{X}_u$ and $\mathbf{X}_v$, affinity matrix $\mathbf{A}$, max iteration T **Output:** Parameters $\mathbf{W}_u, \mathbf{W}_v$.**Step 1 (Initialization):****1.1** Perform centering in $\mathbf{X}_u$ and $\mathbf{X}_v$ **1.2** Initialize $\mathbf{B}^{(1)}$ randomly**1.3** Initialize $\mathbf{W}_u$ and $\mathbf{W}_v$ randomly**Step 2 (Optimization):****for** $t = 1, 2, \dots, T$ **do** **2.1** Solve $\mathbf{B}^{(t+1)}$ using (4.8); **2.2** Solve $\mathbf{W}_u$ using (4.17); **2.3** Solve $\mathbf{W}_v$ using (4.18). If $t > 1$ and $|J_t - J_{t-1}| < \varepsilon$, go to **Return**.**end****Return:** $\mathbf{W}_u, \mathbf{W}_v$.

The optimization function in (4.10) can be re-written as:

$$\min_{\mathbf{W}_*} J(\mathbf{W}_*) = \|\mathbf{B} - \mathbf{W}_*^\top \mathbf{X}_*\|^2 \quad (4.16)$$

Since (4.16) is a regularized least squares problem, we can easily obtain a closed-form solution to $\mathbf{W}$ as follows:

$$\mathbf{W}_u = (\mathbf{X}_u \mathbf{X}_u^\top + \lambda \mathbf{I})^{-1} \mathbf{X}_u \mathbf{B}^\top \quad (4.17)$$

$$\mathbf{W}_v = (\mathbf{X}_v \mathbf{X}_v^\top + \lambda \mathbf{I})^{-1} \mathbf{X}_v \mathbf{B}^\top \quad (4.18)$$

Algorithm 3 details the implementation of our proposed CMDH-linear method.

4.2.1.2 CMDH-kernel

To better exploit the nonlinear relationship of samples, we also develop a CMDH-kernel method by using a nonlinear mapping in the kernel space. Specifically, we first map samples from each modality to a set of g points called *anchors* defined as $[\mathbf{a}_1, \dots, \mathbf{a}_g]$. Anchors can be selected randomly or using k-means². Then, we learn a linear projection matrix,

²From empirical results, there is no significant difference in terms of performance.

Algorithm 4: CMDH - *kernel*

Input: Training set $\mathbf{X}_u$ and $\mathbf{X}_v$, affinity matrix $\mathbf{A}$, number of anchor points g .

Output: Parameters $\mathbf{P}_u, \mathbf{P}_v$.

Step 1 (Initialization):

1.1 Perform centering in $\mathbf{X}_u$ and $\mathbf{X}_v$

1.2 Initialize $\mathbf{B}^{(1)}$ randomly

1.3 Select g anchor points, $\{\mathbf{a}_{u,i}\}_{i=1}^g$ and $\{\mathbf{a}_{v,i}\}_{i=1}^g$ randomly from $\mathbf{X}_u$ and $\mathbf{X}_v$, respectively

Step 2 (Optimization):

for $t = 1, 2, \dots, T$ **do**

 2.1 Solve $\mathbf{B}^{(t+1)}$ using (4.8);

 2.2 Solve $\mathbf{P}_u$ using (4.21);

 2.3 Solve $\mathbf{P}_v$ using (4.22).

 If $t > 1$ and $|J_t - J_{t-1}| < \varepsilon$, go to **Return**.

end

Return: $\mathbf{P}_u, \mathbf{P}_v$.

$\mathbf{P}_* \in \mathbb{R}^{K \times g}$, which maps samples from a kernel space into a low-dimensional space.

$$f_{\text{kernel}}(\mathbf{x}_*, \mathbf{P}_*) = \mathbf{P}_*^\top \kappa(\mathbf{x}_*) \quad (4.19)$$

where $\kappa(\mathbf{x}) = [\exp(d(\mathbf{x}, \mathbf{a}_1)/\sigma), \dots, \exp(d(\mathbf{x}, \mathbf{a}_g)/\sigma)]$ is the RBF kernel mapping function with σ as the kernel bandwidth parameter, and $d(\cdot)$ is the ℓ_2 -distance function.

Similarly, the optimization function from (4.10) can be re-written as:

$$\min_{\mathbf{P}_*} J(\mathbf{P}_*) = \|\mathbf{B} - \mathbf{P}_*^\top \kappa_*(\mathbf{X}_*)\|_F^2 \quad (4.20)$$

It can be seen that (4.20) is also a regularized least squares problem, and $\mathbf{P}$ can be easily solved using a closed-form solution:

$$\mathbf{P}_u = (\kappa(\mathbf{X}_u)\kappa(\mathbf{X}_u)^\top)^{-1}\kappa(\mathbf{X}_u)\mathbf{B}^\top \quad (4.21)$$

$$\mathbf{P}_v = (\kappa(\mathbf{X}_v)\kappa(\mathbf{X}_v)^\top)^{-1}\kappa(\mathbf{X}_v)\mathbf{B}^\top \quad (4.22)$$

Algorithm 4 details the implementation of our proposed CMDH-kernel method.

4.2.2 Shallow CMDH extensions

4.2.2.1 Unsupervised CMDH (unCMDH)

While our shallow CMDH is developed for supervised hashing, it can easily be extended to unsupervised, where only the cross-modal pair information is available.³ An estimated affinity matrix, $\tilde{\mathbf{A}}$, can be obtained based on the neighborhood structure of each modality using anchor graphs [62]. We assume that instances that are very much near (distance-wise) to each other in both modalities are semantically related to some extent.

Following [62], we first randomly select g data pairs from the training set and let them be anchor pairs $\{\mathbf{a}_{u,i}, \mathbf{a}_{v,i}\}_{i=1}^g$. For each modality, we define the data-to-anchor affinity matrix, $\Gamma_* \in \mathbb{R}^{N \times g}$, as follows:

$$\Gamma_{*,i,j} = \begin{cases} \frac{\exp(-d(\mathbf{x}_{*,i}, \mathbf{a}_{*,j}/\sigma))}{\sum_{j' \in \langle i \rangle} \exp(-d(\mathbf{x}_{*,i}, \mathbf{a}_{*,j'})/\sigma)} & \forall j \in \langle i \rangle \\ 0 & \text{otherwise} \end{cases} \quad (4.23)$$

where $\langle i \rangle \subset [1, \dots, g]$ denotes the indices of g' nearest anchors ($g' \ll g$) based on ℓ_2 -distance, and σ is the bandwidth. The data-to-data affinity matrix can then be computed as $\mathbf{A}_* = \Gamma_* \Lambda_*^{-1} \Gamma_*^\top \in \mathbb{R}^{N \times N}$ where $\Lambda_* = \text{diag}(\Gamma_*^\top \mathbf{1}) \in \mathbb{R}^{g \times g}$. The final affinity matrix is then the summation of the affinity matrices of both modalities:

$$\tilde{\mathbf{A}} = \mathbf{A}_u + \mathbf{A}_v \quad (4.24)$$

This affinity matrix is then normalized so that each column would have a sum of 1. We follow the optimization procedure in **Algorithm 2** except we replace the original $\mathbf{A}$ with $\tilde{\mathbf{A}}$.

³Previously published works [18, 80, 81] have stated that knowing each paired instance during training is considered as "unsupervised" learning, but this can also be called as semi-supervised due to pairwise information. But different from single-modal semi-supervised setting, we only know that a modality instance has a counterpart modality (e.g. image1 is same as text1). However, we have no information if this instance is similar to other instances (e.g. is image1 same as text2? is image1 same as text3?).

4.2.2.2 Multi-Modal Discrete Hashing (MMDH)

Our method can also be easily extended for multi-modal experiments. We also propose a Multi-Modal Discrete Hashing (MMDH) method to learn binary codes from multiple modalities.

Given a group of M different modalities $U = \{u^{(1)}, \dots, u^{(M)}\}$ and based on the initial formulation in (4.1), we obtain a new objective formulation suitable for multiple modalities as follows:

$$\begin{aligned} \min_{\mathbf{b}, \{\mathbf{h}_{u^{(m)}}\}_{m=1}^M} J &= \sum_{i,j=1, i \neq j}^N \|\mathbf{b}_i - \mathbf{b}_j\|_2^2 A_{ij} + \eta \sum_{m=1}^M \sum_{i=1}^N \|\mathbf{b}_i - \mathbf{h}_{u^{(m)},i}\|_2^2 \\ \text{s.t.} \quad &\mathbf{b}_i \in \{-1, 1\}^K \end{aligned} \quad (4.25)$$

Similarly, following the simplification and derivation of shallow CMDH, we can obtain a new B-step optimization procedure as follows:

$$\mathbf{B}^{(t+1)} = \text{sgn} \left(2\mathbf{A}\mathbf{B}^{(t)} + \eta \sum_{m=1}^M \mathbf{H}_{u^{(m)}} \right) \quad (4.26)$$

The H-step optimization procedure remains the same but with updates for M modalities. This is then repeated until convergence. The new optimization procedure is presented in **Algorithm 5**. It is then straightforward to implement the linear and kernel forms of MMDH based on **Algorithms 3** and **4**, respectively.

4.2.3 Deep CMDH

Motivated by the advances of deep networks, we propose a deep CMDH. Unlike the shallow CMDH, the nonlinear relationship of samples with the same modality and the relationship of samples from different modalities can be well exploited through hierarchical nonlinear mappings.

Algorithm 5: MMDH

Input: Training set $\mathbf{X}_{u(1)}, \dots, \mathbf{X}_{u(M)}$,
 affinity matrix $\mathbf{A}$.
Output: Parameters $\{\theta_{u(m)}\}_{m=1}^M$.
Step 1 (Initialization):
 1.1. Initialize $\mathbf{B}^{(1)}$ randomly
 1.2. Initialize $\{\theta_{u(m)}\}_{m=1}^M$ according to $f(\cdot)$
Step 2 (Optimization):
for $t = 1, 2, \dots, T$ **do**
 2.1. Solve $\mathbf{B}^{(t+1)}$ using (4.26);
 2.2. Solve $\{\theta_{u(m)}\}_{m=1}^M$:
 for $m = 1, 2, \dots, M$ **do**
 Solve $\theta_{u(m)}$ based on (4.10).
 end
 If $t > 1$ and $|J_t - J_{t-1}| < \varepsilon$, go to **Return**.
end
Return: $\{\theta_{u(m)}\}_{m=1}^M$.

Given a pair of training samples $\{\mathbf{x}_{u,i}, \mathbf{x}_{v,i}\}_{i=1}^N$, we obtain the modality-specific real-valued neural codes, $\mathbf{h}_*$, from the top layer of each network (parametrized by θ_u and θ_v) as follows:

$$\mathbf{h}_{*,i} = f_{\text{deep}}(\mathbf{x}_{*,i}, \theta_*) = \phi_*(\mathbf{x}_{*,i}, \theta_*) \quad (4.27)$$

where $\phi_*(\cdot, \cdot)$ performs hierarchical nonlinear mappings such that $\phi_* : \mathbb{R}^{d_*} \rightarrow \mathbb{R}^k$. These real-valued codes can be embedded into the hamming space directly by applying a $\text{sgn}(\cdot)$ function to the real-valued network output as follows:

$$\mathbf{b}_{u,i} = \text{sgn}(\mathbf{h}_{u,i}) \quad (4.28)$$

$$\mathbf{b}_{v,i} = \text{sgn}(\mathbf{h}_{v,i}) \quad (4.29)$$

But similar to shallow CMDH, we want to reduce the modality gap between the binary codes obtained from each modality as much as possible. Hence, we define a shared binary code, $\mathbf{b}_i = \mathbf{b}_{u,i} = \mathbf{b}_{v,i}$, to represent the cross-modal pairs during training. By doing so, the neural codes from different modalities with similar semantics are encoded to the same

Figure 4.2: The basic idea of our proposed deep CMDH for multimedia retrieval. We learn a couple of hashing networks and a joint binary code matrix using an alternative and iterative optimization procedure. The unified binary code B is learned through a discrete optimization procedure to avoid relaxing the binary constraints. The class projection matrix M is learned by solving the SVM model that maps the binary code to its corresponding label ground truth. The network parameters are learned through back-propagation by minimizing the quantization loss between real-valued network codes (H_u and H_v) and shared binary code and maximizing the variance of the real-valued codes.

binary code, implicitly reducing the modality gap. Hence, our CMHN model aims to learn a unified binary code matrix $B = [b_1, b_2, \dots, b_N] \in \{-1, 1\}^{N \times K}$, and the parameters of the cross-modal hashing networks $\theta_u \triangleq \{W_u^{(l)}, c_u^{(l)}\}_{l=1}^L$ and $\theta_v \triangleq \{W_v^{(l)}, c_v^{(l)}\}_{l=1}^L$ simultaneously. $W_*^{(l)}$ and $c_*^{(l)}$ are the projection matrix and bias vector for the l th layer, and L is the top-most layer in the network. Also similar to shallow CMDH, we learn the binary codes and hash networks alternately in a discrete manner so that we can preserve the binary constraints to avoid the approximation loss caused by relaxation. Also, to make our binary codes discriminative such that samples that are semantically relevant (or irrelevant) have similar (or dissimilar) binary codes as much as possible, we exploit the label information to enforce maximizing the semantics correlation using a classification-based hinge loss criterion. Figure 4.2 shows the pipeline of our proposed deep CMDH approach during training. We now detail the joint formulation and optimization procedure below.

Learning Unified Binary codes: To learn discriminative unified binary codes, we fol-

low the assumption that these codes should be able to perform well on a multi-classification problem. Hence, we create a hinge loss function which ensures that the unified binary code can be linearly transformed to its corresponding label information as much as possible. Suppose for each cross-modal pair $\{\mathbf{x}_{u,i}, \mathbf{x}_{v,i}\}$ we have its annotated label information $\mathbf{y}_i \in \{1, 0\}^C$, where C is the number of class labels and $y_{i,c} = 1$ if the i th sample belongs to class c and 0 otherwise, we can have a multi-class projection matrix defined as $\mathbf{M} = [\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_C] \in \mathbb{R}^{K \times C}$ that maps $\mathbf{b}_i$ to $\mathbf{y}_i$ through a hinge loss defined as follows:

$$\begin{aligned} \min_{\mathbf{B}, \mathbf{M}} J_1 &= \|\mathbf{M}\|_F^2 + \sum_{i=1}^N \xi_i \\ \forall i, c \quad y_{i,c}(\mathbf{m}_c^\top \mathbf{b}_i) &\geq 1 - \xi_i \end{aligned} \quad (4.30)$$

where the ξ_i is the slack variable. We minimize the hinge loss to learn both the unified binary code matrix and the multi-class projection matrix.

Learning Deep Cross-modal Networks: To learn the parameters θ_u and θ_v for our cross-modal network, we employ the following formulation:

$$\min_{\theta_u, \theta_v} J_2 = (\|\mathbf{B} - \mathbf{H}_u\|_F^2 + \|\mathbf{B} - \mathbf{H}_v\|_F^2) - \alpha (\text{tr}(\mathbf{H}_u \mathbf{H}_u^\top) + \text{tr}(\mathbf{H}_v \mathbf{H}_v^\top)) \quad (4.31)$$

where $\mathbf{H}_u$ and $\mathbf{H}_v$ are the neural code matrix obtained from (4.27).

As can be seen, we learn the deep hashing networks from a single unified binary code by minimizing the quantization loss between real-valued neural codes and binary codes. By doing so, the energy of the samples can be well preserved in the learned hashing networks. In addition, we add a regularizer term parameterized by α which maximizes the variances of the zero-centered neural code from different modalities. By doing so, the binary bits are as independent as possible for each modality [38, 41].

Iterative Optimization: Combining the two objectives aforementioned, we are able to formulate the following optimization objective to learn compact binary codes $\mathbf{B}$, the classification projection matrix $\mathbf{M}$, and the parameters for the cross-modal hashing networks θ_u and θ_v :

$$\min_{\mathbf{B}, \mathbf{M}, \theta_u, \theta_v} J = J_1 + \lambda_1 J_2 \quad (4.32)$$

where λ_1 is a constant parameter to balance the effects of the two terms in the objective function.

The optimization problem in (4.32) is non-convex due to the binary constraints, which makes it difficult to solve. However, it can be addressed using an iterative approach where we keep other variables fixed and solve one alternately. We learn the binary code, multi-class projection matrix, and hashing functions simultaneously as follows:

Classification Step (C-Step): Fixing $\mathbf{B}$ and parameters θ_* , we are left with an SVM formulation which can be solved through a standard solver⁴ to learn the classification matrix $\mathbf{M}$ in (4.30).

Binary Code Step (B-Step): Fixing $\mathbf{M}$ and parameters θ_* , we can simplify (4.32) as follows:

$$\begin{aligned} \min_{\mathbf{b}_i} J(\mathbf{b}_i) &= - \sum_{c=1}^C \mathbf{m}_c^\top \mathbf{b}_i + \lambda_1 (\|\mathbf{b}_i - \mathbf{h}_{u,i}\|_2^2 + \|\mathbf{b}_i - \mathbf{h}_{v,i}\|_2^2) \\ \text{s. t.} \quad &\mathbf{b}_i \in \{-1, 1\}^{1 \times K} \end{aligned} \quad (4.33)$$

We learn $\mathbf{B}$ through a discrete optimization technique. (4.33) becomes a binary quadratic problem which can be solved through a linear gradient technique similar to [16] as follows:

$$\mathbf{B}^{(t+1)} = \text{sgn}(\mathbf{Y}\mathbf{M}^\top + \lambda_1(\mathbf{H}_u + \mathbf{H}_v)) \quad (4.34)$$

⁴we use LibSVM: <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>

where $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_N]$.

Hashing Function Step (H-Step): To solve the network parameters, we fix $\mathbf{B}$ and $\mathbf{M}$ from (4.32) and obtain the following formulation:

$$\begin{aligned} \min_{\theta_u, \theta_v} J(\theta_u, \theta_v) = & \lambda_1 (\|\mathbf{B} - \mathbf{H}_u\|_F^2 + \|\mathbf{B} - \mathbf{H}_v\|_F^2) \\ & - \lambda_1 \alpha (\text{tr}(\mathbf{H}_u \mathbf{H}_u^\top) + \text{tr}(\mathbf{H}_v \mathbf{H}_v^\top)) \end{aligned} \quad (4.35)$$

We employ the batch-wise gradient descent method to learn network parameters for each modality. We obtain the gradient of J in (4.35) with respect to the real-valued neural code. For each layer of the network, the gradients can easily be computed through the chain rule during back-propagation. The parameters of the networks are updated using these gradients based on a given learning rate, momentum, and weight decay.

Out-of-Sample Extension: For new instances or query data, we simply use the learned hashing networks to obtain the neural codes and finally binarize them using the $\text{sgn}(\cdot)$ function as shown in (4.28) and (4.29).

Network Configuration: Our deep architecture was implemented under the MatConvNet⁵ [140] framework. For the *image hashing network*, our deep model used the pre-trained VGG-net [126] as our initial convolution and pooling layers up to FC7 and stacked the number of new FC layers with dimensions of $[4096 \rightarrow 500 \rightarrow 200 \rightarrow K]$ for all experiments. We performed end-to-end learning with the learning rate at the new fully connected layers to be 0.01. To avoid over-fitting and ruining the representative abstract features already learned during the pre-training, we reduced the learning rate for the remaining convolution and FC layers to 0.0001. For the *text hashing network*, we used a series of fully-connected networks and used the pre-processed text features, provided by each experiment, as input. We set the FC layers as $[1386 \rightarrow 500 \rightarrow K]$, and $[1000 \rightarrow 500 \rightarrow K]$,

⁵<http://www.vlfeat.org/matconvnet/>

for the IAPRTC12, and NUS-WIDE dataset, respectively. For the new hidden fully connected layers of both image and text network, we used the ReLU activation as the nonlinear activation function. For the top layer, we used the hyperbolic tangent (tanh) function of the network to squeeze the representation to a $\{-1, 1\}$ range which ensures that the quantization loss can be reduced as much as possible. The weights in the new fully connected layers are initialized using the Xavier initialization [141]. The other network learning parameters, momentum and weight decay, were set to 0.9 and 0.0001, respectively. The loss function parameters, λ_1 and α , were set to 0.2 and 0.001, respectively, which were obtained by cross-validation using $K = 16$ bits.

4.3 Experiments

We conducted experiments on four widely used benchmark datasets in multimedia retrieval to evaluate our shallow CMDH and deep CMDH methods. Particularly, we conducted experiments on three datasets (Wiki, MIRFlickr, NUS-WIDE) to evaluate our shallow CMDH method and two datasets (IAPRTC12, NUS-WIDE) to evaluate our deep CMDH method. The following describes the details of the datasets and experimental set-up.

4.3.1 Datasets and Experimental Set-up

4.3.1.1 Datasets

Wiki dataset [71]⁶: contains 2866 image-text pairs collected from featured Wikipedia articles. For each pair, the image is represented by a 128-dimensional Bag-of-Visual-Words (BoVW) SIFT feature vector, and the text article is represented by a 10-dimensional feature vector which is computed by the Latent Dirichlet Allocation (LDA) model. Each image-text pair is annotated by one of 10 categories. Similar to previous works [81, 82], we

⁶<http://www.svcl.ucsd.edu/projects/crossmodal/>

used 25% samples from the dataset for query and employed the rest as training and gallery instances.

MIRFlickr dataset [142]⁷: contains 25000 image-text pairs collected from Flickr, where images are annotated with textual tags. Each image is represented with a 150-dimensional edge histogram, and each text is represented by a 500-dimensional feature vector which is also computed by the LDA model. Each image-text pair is annotated with one or more of the pre-defined 24 labels. Similar to [82], we only used samples where the labels appeared at least 20 times in the dataset. We randomly selected only 5% samples of the dataset for query and used the rest as gallery samples. We also randomly selected 5000 pairs from the gallery set to train the model.

NUS-WIDE dataset [129]⁸: contains 269,648 image-text pairs collected from the web where images are annotated with 81 label tags. Following the same settings in previous works [80–82], we selected the top 10 most frequent labels and obtained a new subset which consists of 186,577 image-text pairs. For our shallow CMDH experiments, each image is represented by a 500-dimensional BoVW SIFT feature vector and each text is represented by a 1000-dimensional bag-of-words (BoW) feature vector. We used 1% samples of the dataset as query instances and used the rest as gallery samples. We also randomly selected 5000 pairs in the gallery set to train the model. For our deep CMDH experiments, we used raw images as input to our cross-modal hashing network.

IAPRTC-12 dataset [143]⁹: contains 19627 images with corresponding sentence descriptions. These image-sentence pairs present various semantics such as landscape, action and people categories. Similar to [20], we used the top 22 frequent labels from the 275

⁷<http://press.liacs.nl/mirflickr/>

⁸<http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm>

⁹<http://www.imageclef.org/photodata>

concepts obtained generated from the segmentation task. For the text features, we pre-processed the sentence data by removing the *stop words* and extracted a 500-dimensional bag-of-words (BoW) feature. For our deep CMDH, we used raw images as input to the image network. We randomly selected 100 pairs per class as the query set and the remaining data as the gallery set.

4.3.1.2 Evaluation Metrics

To evaluate the performance of different cross-modal hashing methods, we employed the *mean average precision* (mAP) which is the mean of the average precision (AP) of the top R retrieved instances as follows:

$$AP = \frac{1}{M} \sum_{r=1}^R prec(r) \cdot rel(r) \quad (4.36)$$

where M is the number of relevant instances in the database for the query instance, $prec(r)$ denotes the precision of the top r retrieved set, and $rel(r)$ is an indicator of relevance of a given rank (which is set to 1 if relevant and 0 otherwise). For the MIRFlickr and NUS-WIDE datasets which have multiple labels, a retrieved instance is considered relevant for a query if they share at least one label. In our experiments, we set $R = 100$ for all datasets. Besides mAP, we also used the precision-recall curve and top-N precision curve to evaluate the performance of different methods.

Additionally, it is important that a ranking metric is evaluated for datasets with multiple labels. We also used the Normalized Discounted Cumulative Gain (NDCG) [144] for evaluation. For a given query sample $\mathbf{x}_q$, the NDCG is defined as follows:

$$NDCG@p = \frac{1}{Z} \sum_{i=1}^p \frac{2^{r_i} - 1}{\log(1 + i)} \quad (4.37)$$

where Z is the normalized constant and p is the number of retrieved samples in the ranking list. r_i represents the similarity level valued z if the query and i th sample in gallery share

z similar labels, and valued zero if they do not share any label. The NDCG evaluates the ranking by strongly penalizing errors in higher ranked items.

4.3.2 Experimental Results for Shallow CMDH

4.3.2.1 Comparisons with State-of-the-art Cross-Modal Hashing Methods

We compared our CMDH with several state-of-the-art cross-modal hashing methods including CMFH [81], LSSH [80], PDH [18], CCA-ITQ [41], CVH [19], SCM-Orth [67], SCM-Seq [67], SePH-rnd [82] and SePH-km [82]. Specifically, CMFH, LSSH, PDH, and CCA-ITQ exploit the correspondence of cross-modal similarity to learn hash codes, and CVH, SCM-Orth, SCM-Seq, SePH-rnd, and SePH-km exploit the label information of samples to learn discriminative hash codes. The standard implementations from the original authors using the default parameters were used for all methods except CVH. We carefully implemented the CVH method since its code is not publicly available. We conducted experiments with the same randomly selected training, gallery and query sets and repeated the experiments 10 times. The final performance is obtained by taking the average of the results. For our CMDH-kernel and CMDH-linear, we set the balancing parameter η to be 0.5 for both methods. To be consistent with the kernel mapping of SePH, we used 500 anchors and σ of 0.6 for the CMDH-kernel. For other compared methods, we set the default parameters according to the original papers.

For consistency and fair comparison of the different methods, we first evaluated these methods using the out-of-sample extension of the gallery set. Tables 4.1-4.2 show the mAP results of different methods on different datasets with varying the binary code length. Figures 4.3-4.5 show the precision-recall curves and top-N precision curves on the Wiki, MirFlickr, and NUS-WIDE datasets, respectively. We can clearly see from these results that our CMDH-kernel has the best performance along with SePH, and outperforms all other

Table 4.1: mAP performance comparison of our shallow CMDH methods and different cross-modal hashing methods, where images were used as query samples and texts/tags were employed as gallery samples, respectively.

Method	Wiki					MIRFlickr25k					NUS-WIDE				
	16	32	64	128	16	32	64	128	16	32	64	128	16	32	64
CVH [19]	0.1988	0.1772	0.1613	0.16	0.6209	0.6112	0.6057	0.5933	0.4211	0.4197	0.4119	0.4125			
CCA-ITQ [41]	0.2342	0.2406	0.2306	0.2241	0.6448	0.6420	0.6333	0.6350	0.4265	0.4267	0.4200	0.4152			
PDH [18]	0.1841	0.1799	0.1858	0.1887	0.6202	0.6291	0.6321	0.6366	0.5021	0.5282	0.5432	0.5434			
LSSH [80]	0.2239	0.226	0.2199	0.2149	0.6361	0.6297	0.6247	0.6292	0.4780	0.4836	0.4815	0.4940			
CMFH [81]	0.2260	0.2466	0.2484	0.2547	0.6432	0.6403	0.6304	0.6204	0.4456	0.4570	0.4752	0.4740			
unCMDH-ker	0.2354	0.2373	0.2612	0.2582	0.6937	0.6947	0.7053	0.7083	0.5184	0.5411	0.5683	0.5784			
unCMDH-lin	0.2157	0.2296	0.2331	0.2437	0.6715	0.6810	0.6849	0.6929	0.4300	0.4242	0.4224	0.4215			
SCM-orth [67]	0.1985	0.1794	0.1644	0.1615	0.6863	0.6913	0.6983	0.6996	0.5306	0.5373	0.5481	0.5433			
SCM-seq [67]	0.1388	0.1398	0.1389	0.1377	0.6498	0.6348	0.6160	0.6120	0.4874	0.4722	0.4452	0.4133			
SePH-rnd [82]	0.2681	0.2862	0.2937	0.2915	0.6986	0.7057	0.7122	0.7129	0.5558	0.5596	0.5665	0.5807			
SePH-km [82]	0.2691	0.2775	0.2940	0.2928	0.7020	0.7052	0.7117	0.7174	0.5503	0.5603	0.5724	0.5854			
CMDH-ker	0.2705	0.2866	0.2894	0.2982	0.6915	0.7046	0.7126	0.72204	0.5622	0.5691	0.5720	0.5849			
CMDH - lin	0.2484	0.2576	0.2700	0.2731	0.6861	0.6815	0.6867	0.6950	0.5370	0.5450	0.5623	0.5705			

Table 4.2: mAP performance comparison of our shallow CMDH methods and different cross-modal hashing methods, where texts/tags were used as query samples and images were employed as gallery samples, respectively.

Method	Wiki				MIRFlickr25k				NUS-WIDE			
	16	32	64	128	16	32	64	128	16	32	64	128
CVH [19]	0.2692	0.2297	0.2047	0.1821	0.6176	0.6121	0.6055	0.5985	0.4161	0.4114	0.4079	0.4102
CCA-ITQ [41]	0.3735	0.3627	0.3514	0.3328	0.6353	0.6303	0.6277	0.6256	0.4201	0.4171	0.4164	0.4192
PDH [18]	0.1937	0.1958	0.2020	0.2026	0.6152	0.6220	0.6301	0.6360	0.4881	0.5120	0.5350	0.5451
LSSH [80]	0.5318	0.5606	0.5662	0.5660	0.6462	0.6530	0.6560	0.6690	0.4789	0.5078	0.5167	0.5156
CMFH [81]	0.3463	0.3701	0.3815	0.3980	0.6336	0.6367	0.6362	0.6358	0.4553	0.4683	0.4744	0.4769
unCMDH-ker	0.3606	0.4132	0.4746	0.5046	0.7015	0.7082	0.7217	0.7216	0.5166	0.5307	0.5764	0.5918
unCMDH-lin	0.3519	0.3784	0.3926	0.4041	0.6742	0.6754	0.6745	0.6839	0.4371	0.4271	0.4257	0.4309
SCM-orth [67]	0.2593	0.2170	0.1898	0.1769	0.6747	0.6867	0.7024	0.7080	0.5162	0.5444	0.5527	0.5621
SCM-seq [67]	0.1454	0.1489	0.1465	0.1459	0.6587	0.6342	0.6158	0.6106	0.4878	0.4702	0.4529	0.4429
SePH-rnd [82]	0.6100	0.6411	0.6512	0.6619	0.7037	0.7220	0.7358	0.7451	0.5777	0.5820	0.5931	0.6075
SePH-km [82]	0.6053	0.6383	0.6483	0.6520	0.7052	0.7201	0.7358	0.7454	0.5833	0.5812	0.6191	0.6245
CMDH-ker	0.6125	0.6438	0.6598	0.6642	0.7157	0.7273	0.7432	0.7551	0.5885	0.5936	0.6005	0.6142
CMDH - lin	0.4133	0.4388	0.4576	0.4613	0.6953	0.6917	0.7001	0.7103	0.5547	0.5588	0.5697	0.5805

Figure 4.3: Precision-Recall (PR) curves and Precision-N curves on the Wiki dataset of varying code lengths for Image-to-Text and Text-to-Image tasks.

compared cross-modal hashing methods. While our CMDH-kernel is competitive with the SePH in terms of retrieval accuracy, our method has less computational costs due to simpler optimization procedure. It can also be seen that there are instances where some methods show a decrease in performance when the bit length reaches 128 dimensions, particularly CMFH and SCM. Our method does not suffer from this limitation.

Overall, our implementation shows competitive performance in all three datasets which may be attributed to two key concepts. First, unlike CVH, CCA-ITQ, and PDH, our method

Table 4.3: mAP performance comparison of our shallow CMDH methods and different cross-modal hashing methods on different datasets where shared binary codes were used as gallery samples.

	$\mathcal{I} \rightarrow \mathcal{T}$							
	Wiki				MIRFlickr25k			
Method	16	32	64	128	16	32	64	128
LSSH [80]	0.3892	0.3866	0.3861	0.3806	0.6407	0.6450	0.6460	0.6547
CMFH [81]	0.2742	0.2876	0.2960	0.3045	0.6408	0.6327	0.6306	0.6244
unCMDH - <i>ker</i>	0.3725	0.3928	0.4062	0.4138	0.6981	0.6959	0.7029	0.7146
unCMDH - <i>lin</i>	0.3238	0.3355	0.3517	0.3635	0.6820	0.6798	0.6842	0.6875
SePH - <i>rnd</i> [82]	0.4336	0.4455	0.4620	0.4862	0.7189	0.7229	0.7291	0.7352
SePH - <i>km</i> [82]	0.4482	0.4601	0.4735	0.4818	0.7279	0.7283	0.7333	0.7325
CMDH - <i>ker</i>	0.4610	0.4651	0.4729	0.4869	0.7238	0.7203	0.7312	0.7316
CMDH - <i>lin</i>	0.4364	0.4479	0.4401	0.4573	0.6856	0.6844	0.6833	0.6818
	$\mathcal{T} \rightarrow \mathcal{I}$							
	Wiki				MIRFlickr25k			
Method	16	32	64	128	16	32	64	128
LSSH [80]	0.6119	0.6427	0.6560	0.6840	0.7033	0.7268	0.7352	0.7405
CMFH [81]	0.4004	0.4212	0.4582	0.4833	0.7002	0.7191	0.7387	0.7597
unCMDH - <i>ker</i>	0.5776	0.6341	0.6376	0.6500	0.7563	0.7611	0.7854	0.7848
unCMDH - <i>lin</i>	0.6060	0.6160	0.6219	0.6401	0.7389	0.7582	0.7653	0.7825
SePH - <i>rnd</i> [82]	0.7166	0.7192	0.7252	0.7281	0.7945	0.8047	0.8194	0.8275
SePH - <i>km</i> [82]	0.7081	0.7225	0.7269	0.7296	0.8074	0.8086	0.8226	0.8344
CMDH - <i>ker</i>	0.7427	0.7416	0.7494	0.7502	0.8102	0.8185	0.8364	0.8390
CMDH - <i>lin</i>	0.6982	0.7001	0.7040	0.7102	0.7761	0.7903	0.8010	0.8032

Figure 4.4: Precision-Recall (PR) curves and Precision-N curves on the MIRFlickr dataset of varying code lengths for Image-to-Text and Text-to-Image tasks.

learns a unified binary code which implicitly eliminates the modality gap between the two modalities. This, in turn, learns a more representative hashing function for each modality. Second, different from LSSH, CMFH, and SePH, our method learns the unified binary code as a separate variable in a discrete manner. This avoids the possible approximation loss caused by any relaxation during optimization. We can also see from the results that representing the hashing function as a nonlinear representation, such as a kernel function, can significantly improve the performance. While the SePH-km also performs well as it

uses a kernel representation, it tries to minimize the quantization loss of real-valued output and binary code explicitly. In contrast, we learn our binary codes and real-valued codes as separate variables. It can also be noted that the SePH-km performs quite well particularly at higher bit lengths. This is because their method is able to preserve the semantic affinities between samples regardless of the bit length requirement. Particularly, SePH preserves the semantic structure by minimizing the Kullback-Leibler divergence between the probability of observing the semantic similarity of training pairs and the probability of observing the similarity of pairs in the Hamming space. The optimization procedure consists of a regularizer and relaxing the binary codes. These details add more complexity in the gradient optimization which leads to learning effective binary codes but longer training time. (See more details in Computational Complexity subsection.)

4.3.2.2 Comparison with State-of-the-art Cross-Modal Hashing Methods using Shared Binary Code as Gallery

To better show the advantage of learning shared binary codes, we also conducted experiments on the Wiki and MIRFlickr datasets where our CMDH was trained on the whole gallery set and the learned shared binary codes were used for search during testing. We compared our CMDH with three other cross-modal hashing methods which also learn shared binary codes for retrieval, namely LSSH, CMFH, and SePH. To further improve the performance of different methods on the Wiki experiment, we extracted convolutional neural networks (CNN) feature for each image instead of the conventional SIFT feature. We used the pre-defined deep model in [126], a CNN-F structure which consists of 5 convolutional and 3 fully-connected layers, to represent each image to a 4096-dimensional CNN feature and employed PCA to reduce it into 512 dimensions. Table 4.3 shows the mAP results for this experiment where the binary code length was set at 16, 32, 64 and 128, respectively. We can see that the performance of all methods on the MIRFlickr dataset

Figure 4.5: Precision-Recall (PR) curves and Precision-N curves on the NUS-WIDE dataset of varying code lengths for Image-to-Text and Text-to-Image tasks.

significantly improved because the shared binary codes were used for retrieval. Moreover, the performance is further improved on the Wiki dataset due to the use of CNN features. This also shows that our method is flexible to any type of feature used.

It can be noticed in Tables 4.1 and 4.3 the $I \rightarrow T$ experiment showed lower mAP than $T \rightarrow I$ across all methods; with our approach is comparable with the performance of SePH. This is because it is generally more difficult to retrieve information from a text database since text data are more challenging to represent as compact features since it

Table 4.4: mAP performance comparison of our MMDH and other multi-modal hashing method on the Wiki dataset where shared binary codes were used as gallery samples.

	Method	16 bits	32 bits	64 bits	128 bits
Image-SIFT feature	CMFH-MM [81]	0.1992	0.2030	0.2090	0.2123
	unMMDH-ker	0.2258	0.2510	0.2577	0.2624
	unMMDH-lin	0.2326	0.2377	0.2528	0.2499
	MMDH-ker	0.2437	0.2529	0.2668	0.2651
	MMDH-lin	0.2154	0.2407	0.2422	0.2486
Text Feature	CMFH-MM [81]	0.3802	0.4034	0.4376	0.4671
	unMMDH-ker	0.4239	0.5229	0.5626	0.5870
	unMMDH-lin	0.4678	0.5283	0.5735	0.5785
	MMDH-ker	0.6634	0.6755	0.6894	0.7026
	MMDH-lin	0.6544	0.6677	0.6746	0.6797
Image-CNN feature	CMFH-MM [81]	0.2573	0.2697	0.2761	0.2828
	unMMDH-ker	0.3035	0.3516	0.3733	0.3898
	unMMDH-lin	0.2745	0.3228	0.3548	0.3661
	MMDH-ker	0.3704	0.3867	0.4005	0.4058
	MMDH-lin	0.3050	0.3297	0.3286	0.3330

deals with word association, stop words, and sequences, compared to images where we can exploit edge, color, and spatial information easily. Since our text features are from a Bag-of-Words (BoW) model, this may have limited the information available for our two-staged compact binary code learning approach hence obtained sub-optimal convergence. Nevertheless, our method is still comparable with the SePH but ours do not require lengthy optimization.

4.3.2.3 Comparisons with State-of-the-art Unsupervised Cross-Modal Hashing Methods

We also include the performance of our unCMDH in Tables 4.1-4.3. As can be seen, it is competitive with other unsupervised methods particularly with the popular LSSH and CMFH method. This shows that our framework is flexible for scenarios with or without label information available.

Table 4.5: mAP performance comparison of our MMDH and other multi-modal hashing method on the NUS-WIDE dataset where out-of-sample binary codes were used as gallery samples.

	Method	16 bits	32 bits	64 bits	128 bits
Image SIFT-BoW feature $\rightarrow$ Text Feature	CMFH-MM [81]	0.4327	0.4582	0.4728	0.4730
	unMMDH-ker	0.4897	0.5048	0.5252	0.5428
	unMMDH-lin	0.3853	0.4078	0.42336	0.4308
	MMDH-ker	0.5435	0.5542	0.5690	0.5783
	MMDH-lin	0.4868	0.5172	0.5148	0.5443
Text Feature $\rightarrow$ Image SIFT-BoW feature	CMFH-MM [81]	0.4495	0.4668	0.4690	0.4687
	unMMDH-ker	0.4824	0.4995	0.5233	0.5483
	unMMDH-lin	0.3818	0.3931	0.4070	0.4229
	MMDH-ker	0.5594	0.5821	0.6037	0.6081
	MMDH-lin	0.4680	0.5025	0.5033	0.5369
Image block-wise LAB feature $\rightarrow$ Text Feature	CMFH-MM [81]	0.3642	0.3710	0.3759	0.3821
	unMMDH-ker	0.3685	0.3795	0.3751	0.3820
	unMMDH-lin	0.3673	0.3725	0.3787	0.3713
	MMDH-ker	0.3676	0.3656	0.3780	0.3728
	MMDH-lin	0.3746	0.3806	0.3839	0.3876

4.3.2.4 Comparisons with State-of-the-art Multi-Modal Hashing Methods

To show the advantage of our MMDH method, we conducted another experiment that uses a multi-modal set-up. We implemented the Wiki experiment using three modalities. The different modalities used are the SIFT image features, CNN image features, and text features. Among the methods compared previously, only CMFH presents a multi-modal extension which we call CMFH-MM. For our implementation, we used the learned unified binary codes as gallery codes. Table 4.4 shows the mAP of our MMDH method compared to CMFH-MM. It can be seen that the performance is lower than that of Table 4.3. This is because it is more challenging to learn a unified binary code that would be consistent with the three modalities provided. Nevertheless, our MMDH method still achieves a competitive or even better performance than CMFH for multi-modal experiments.

Similarly, we performed our multi-modal hashing method in the NUS-WIDE experi-

ment. Here, we used the out-of-sample extension approach to obtain the binary codes for the gallery set. We used the block-wise color histogram (LAB) image feature as the additional modality. Table 4.5 shows the mAP for the NUS-WIDE multi-modal experiments. As can be seen, regardless of the features used, our methods, MMDH-ker and MMDH-lin, outperform the CMFH-MM during retrieval. This shows that our methods are flexible when it comes to the type of pre-processed image feature being used. We also note that a dip in performance is encountered when doing Image LAB feature $\rightarrow$ Text retrieval as compared to the Image SIFT-BoW feature $\rightarrow$ Text retrieval simply because LAB image features are not very representative of the image’s edge content, unlike SIFT image feature. It can also be noticed that in the Image LAB feature $\rightarrow$ Text retrieval, MMDH-lin and MMDH-ker are quite similar in performance; this may be because the LAB features are simple and are already easy to represent linearly, and thus do not need to be in kernel form. From this observation, we can say that not all hand-crafted feature space need to be determined by a kernel function. And for simplicity, one can just use the MMDH-lin if we aim for a model that requires less computational complexity.

4.3.2.5 Comparisons of Different Learning Strategies

To further show the advantage of the discrete optimization and joint binary codes learning strategy of our CMDH, we constructed two other baselines: relaxed cross-modal hashing (RCMH) and individual cross-modal discrete hashing (ICMDH).

The RCMH learns binary codes by removing the binary constraint on (4.1) and treating it as real-valued code during optimization. This is similar to several popular hashing methods [38, 41, 82], where the binary variable is re-defined without the $\text{sgn}(\cdot)$ operation and instead used its signed-magnitude value during relaxed optimization. Here, we also ignored the constraint in the discriminative loss term of (4.6) and instead used the real-valued code

in (4.2) and (4.3) such that the objective becomes:

$$J = -\text{tr}(\mathbf{H}_u^\top \mathbf{A} \mathbf{H}_u + \mathbf{H}_v^\top \mathbf{A} \mathbf{H}_v) + \|\mathbf{B} - \mathbf{H}_u\|_F^2 + \|\mathbf{B} - \mathbf{H}_v\|_F^2 \quad (4.38)$$

The optimization procedure can then be solved using gradient descent method.

For ICMDH, instead of shared binary codes, we learned individual binary codes for each modality such that the formulation from (4.6) becomes:

$$J = \sum_{*=u,v} \left(-\text{tr}(\mathbf{B}_*^\top \mathbf{A} \mathbf{B}_*) + \|\mathbf{B}_* - \mathbf{H}_*\|_F^2 \right) + \|\mathbf{B}_u - \mathbf{B}_v\|_F^2 \quad (4.39)$$

where the last term ensures that the binary code pair $\{\mathbf{b}_{u,i}, \mathbf{b}_{v,i}\}$ from a cross-modal pair instance is similar as possible. The optimization procedure can then be solved in a way similar to our iterative discrete optimization except that we try to solve two sets of binary codes.

We repeated the first experiment on the Wiki and MIRFlickr datasets for RCMH and ICMDH. Table 4.6 shows the average mAP (from mAPs obtained in image-text and text-image retrieval experiments) of our CMDH and the other two baseline methods. It can be seen that our CMDH outperforms both RCMH and ICMDH, which indicates that both the discrete optimization technique and the unified binary codes learning strategy are significant in the retrieval performance of our shallow CMDH method.

4.3.2.6 Further Analysis

Parameter Sensitivity: We investigated the sensitivity of our optimization method with respect to the balancing parameter η . We performed experiments at varying values of $\eta = [0.2; 0.4; 0.5; 0.6; 0.8; 1]$ when the code length was set as 16, 32, 64 and 128, respectively. Figures 4.6(a) to 4.6(c) show the mAP of our method versus different length of the binary codes. As can be seen, varying the balance parameter may give different responses in

Table 4.6: Average mAP performance of our CMDH on Wiki and MIRFlickr datasets when different learning strategies were employed.

Method	Wiki		MIRFlickr25k	
	64 bits	128 bits	64 bits	128 bits
CMDH- <i>ker</i>	0.4768	0.4812	0.7279	0.7387
RCMH- <i>ker</i>	0.2976	0.3148	0.5857	0.6086
ICMDH- <i>ker</i>	0.3575	0.3664	0.6384	0.6446
CMDH- <i>lin</i>	0.3638	0.3672	0.6934	0.7026
RCMH- <i>lin</i>	0.3245	0.3223	0.6283	0.6268
ICMDH- <i>lin</i>	0.3490	0.3567	0.6625	0.6591

terms of accuracy for different datasets. In terms of varying bit length, the response is quite similar for each dataset. It is important to note that setting η to 0.2 or 1, which implies that one criterion is being overpowered by the other, consistently do not provide the best results. Hence, the range of $\eta = [0.4, 0.8]$ would be the best setting for this method in general. However, for a specific dataset, it is best to determine this parameter through a k-fold cross-validation testing to obtain the optimal results.

Convergence: We evaluated the convergence of our optimization procedure for both CMDH-kernel and CMDH-linear. Figures 4.8(a)- 4.8(f) show the value of the objective function during training on the NUS-WIDE dataset for various bit lengths. We can see that CMDH-kernel converged with 20-30 iterations and CMDH-linear converged around 100-150 iterations. This tells us that a linear model takes a longer time to obtain an optimum solution compared to the kernel model.

Computational Complexity: We compared the train and test time for different cross-modal hashing methods. The test time represents the time it took to encode in an out-of-sample extension implementation for all query and gallery samples. Our hardware consists of a PC configuration of 3.20 GHz i5-3470 CPU and 32.0 GB RAM. Table 4.7 shows the train and search times for the Wiki and NUS-WIDE dataset, which has 2150 and 5000 training instances, respectively. As can be seen, SePH requires the largest time for training

Figure 4.6: mAP performance of our CMDH versus different values of η on the Wiki, MIRFlickr, and NUS-WIDE datasets.

despite its strong performance. This is mainly due to the complex optimization of minimizing the Kullback-Leibler divergence of their objective function. On the other hand, our CMDH presents an acceptable speed with better performance than other cross-modal hashing methods. Since our optimization is iterative and straightforward, we can easily add more training samples without sacrificing computational time as shown in Figure 4.7.

Table 4.7: Computational time (s) of different cross-modal hashing methods on the Wiki and NUS-WIDE datasets for code length of 16 bits.

Method	Wiki		NUS-WIDE	
	Train	Search	Train	Search
CVH	0.129	.0018	1.103	0.639
CCA-ITQ	0.198	.0049	1.345	2.218
PDH	0.746	.0044	16.77	1.644
LSSH	44.36	3.356	106.2	5.8e3
CMFH	0.327	.0122	344.2	1.651
SCM-orth	0.071	.0037	15.42	1.424
SCM-seq	0.035	.0037	0.953	1.231
SePH-rnd	211.8	0.242	1.2e3	3.214
SePH-km	224.2	0.285	1.2e3	3.541
CMDH-ker	2.124	0.042	34.95	2.881
CMDH-lin	1.557	0.004	109.67	2.494

4.3.3 Experimental Results for Deep CMDH

We compared our deep CMDH with different state-of-the-art shallow cross-modal hashing methods which can be categorized as unsupervised (LSSH, CMFH) or supervised (SePH, CMDH),

Figure 4.7: Training time of different cross-modal hashing methods versus different sizes of the training set.

Table 4.8: mAP performance of our deep CMDH and different cross-modal hashing methods on the IAPRTC12 dataset.

	Method	16 bits	32 bits	64 bits	128 bits
$I \rightarrow T$	CMFH [81]	0.5601	0.5829	0.6079	0.6179
	LSSH [80]	0.5440	0.5769	0.5964	0.5985
	SePH - km [82]	0.6177	0.6447	0.6500	0.6781
	DisCMH [83]	0.6174	0.6596	0.6503	0.6594
	DNNH-C [55]	0.4801	0.5093	0.5259	0.5349
	DVSH [20]	0.5696	0.6321	0.6964	0.7236
	Deep CMDH	0.6483	0.7274	0.7974	0.8251
	Deep CMDH (o)	0.5768	0.7062	0.7780	0.8060
$T \rightarrow I$	CMFH [81]	0.5592	0.5834	0.6084	0.6187
	LSSH [80]	0.4868	0.5264	0.5547	0.5724
	SePH - km [82]	0.6105	0.6340	0.6404	0.6730
	DisCMH [83]	0.6532	0.6910	0.6921	0.6949
	DNNH-C [55]	0.4692	0.4838	0.4905	0.5053
	DVSH [20]	0.6037	0.6395	0.6806	0.6751
	Deep CMDH	0.6687	0.6925	0.7535	0.7925
	Deep CMDH (o)	0.6716	0.6615	0.6677	0.6490

Table 4.9: mAP performance comparison of our deep CMDH and different cross-modal hashing methods on the NUS-WIDE dataset.

	Method	16 bits	32 bits	64 bits	128 bits
$I \rightarrow T$	CMFH [81]	0.4772	0.5301	0.5763	0.6258
	LSSH [80]	0.5547	0.5734	0.5980	0.5968
	SePH - km [82]	0.6177	0.6447	0.6500	0.6781
	DisCMH [83]	0.6826	0.7583	0.7752	0.7605
	CAH [86]	0.4920	0.5084	0.5407	0.5628
	DCMH [22]	0.6249	0.6355	0.6720	-
	Deep CMDH	0.7893	0.8170	0.8236	0.8289
	Deep CMDH (o)	0.6558	0.7480	0.7818	0.7614
$T \rightarrow I$	CMFH [81]	0.4965	0.5432	0.5995	0.6405
	LSSH [80]	0.5857	0.6242	0.6293	0.6464
	SePH - km [82]	0.6604	0.6766	0.7043	0.7024
	DisCMH [83]	0.6519	0.7378	0.7535	0.7511
	CAH [86]	0.5019	0.5135	0.5451	0.5800
	DCMH [22]	0.6791	0.6829	0.6906	-
	Deep CMDH	0.6829	0.7469	0.7651	0.7772
	Deep CMDH (o)	0.6643	0.6950	0.7170	0.7062

Figure 4.8: Objective function values of our CMDH-kernel and CMDH-linear on the NUS-WIDE dataset when the code length is set as 16, 32 and 64.

DisCMH)¹⁰. All four shallow methods also learn a shared binary code from a cross-modal pair. In addition, we also compared our method with current deep cross-modal hashing methods¹¹. We used the CNN features extracted at the FC7 layer of the same pre-trained model as image features for the shallow methods and our deep CMDH methods.

In this experiment, we used the whole gallery as training data to learn the hashing functions. For each dataset, we performed two cross-modal retrieval tasks: image-to-text retrieval and text-to-image retrieval. Tables 4.8 and 4.9 show the mAP performance by Hamming Ranking. It can be observed that our method provides the best performance compared to the shallow cross-modal hashing methods. This may be because of the strong nonlinear representation captured by deep networks. While the SePH performs nonlinear

¹⁰We ran the experiments ourselves using the author’s code except for DisCMH, where we implemented ourselves.

¹¹Results are obtained from the respective author’s papers except for DNH-C which were obtained from DVSH paper. We used the same experimental set-up as mentioned in the papers.

Figure 4.9: NDCG performance of different cross-modal hashing methods for the IAPRTC12 and NUSWIDE database.

transformations, it was done explicitly through kernels which cannot really maximize the information from raw data. While the DisCMH method gave competitive results in some instances, it did not consistently improve as the bit length increased.

Because only the shallow methods perform unified binary code learning, we also tested our method in an out-of-sample extension that we name as deep CMDH(o). Our deep CMDH(o) used the output of the learned hashing networks to obtain the binary codes as gallery instead of the learned unified binary code. As can be seen in Tables 4.8 and 4.9, our deep model gave the best performance for both image-text and text-image retrieval compared to the other deep cross-modal hashing models. This may be due to several reasons. First, the CAH method still relies on hand-crafted image features as input, while

our method performs complete network learning from raw images. Second, the DCMH method performs end-to-end learning but exploits the label information directly to the neural code output of the hash networks, and not the binary code. This can lead to approximation loss during optimization. Finally, DVSH and DNNH-C perform end-to-end supervised metric-based network training in the form of cosine hinge loss and triplet ranking loss, respectively, which may not fully obtain discriminative binary codes compared to our classification-based hinge loss.

Figures 4.9 shows the NDCG performance for both IAPRTC12 and NUS-WIDE experiment. As can be seen, our method shows the best results by a large margin compared to the other methods. This suggests that our method can address the ranking problem well during retrieval. Different from other methods which do not consider the ranking information in their objective functions and only uses hard pairwise similarities, our method exploits the label information fully which implicitly learns ranking well.

4.3.4 Discussion

The above experimental results suggest the following three key observations:

- (i) Our discrete cross-modal hashing methods, both shallow and deep, achieve very competitive or even better performance with the state-of-the-art shallow cross-modal hashing methods on different benchmark datasets. This is because we perform an optimization step that avoids relaxation in the binary constraints, which lessens the information loss when learning hashing functions. Moreover, we learned a joint binary code to implicitly reduce the modality gap between cross-modal data pairs.
- (ii) Our unCMDH and MMDH methods also show competitive performances. This shows the flexibility of our approach for different scenarios in cross-modal retrieval.

- (iii) Our shallow CMDH approach is not only competitive in retrieval performance but also yields an acceptable training speed compared to the state-of-the-art SePH and LSSH methods. The latter two can obtain good mAP performance but require significant time for training.
- (iv) Our deep CMDH method also shows competitive performance compared to current deep cross-modal hashing methods in both using the unified binary code learned and using the out-of-sample extension for the gallery set.

4.4 Chapter Summary

In this chapter, we have proposed a cross-modal discrete hashing (CMDH) method to learn compact binary codes for cross-modality retrieval. Our approach learns shared binary codes for all modalities to implicitly reduce modality gap and is optimized in a discrete manner to avoid possible approximation loss. Specifically, we first presented a shallow CMDH where we described two forms under the proposed framework: CMDH-linear and CMDH-kernel which perform linear and nonlinear mappings to learn binary codes, respectively. Furthermore, we extended our shallow CMDH method to unsupervised CMDH (unCMDH) and multi-modal discrete hashing (MMDH) to make it suitable for unsupervised hashing and multi-modal hashing. Finally, we performed Deep CMDH to learn more representative binary features in a deep framework. Experimental results on four widely used cross-modal datasets clearly showed the effectiveness of our proposed approach. The work presented in this chapter has been published in [31] and [30].

Chapter 5

Deep Variational and Structural Hashing

In this chapter, we propose a deep variational and structural hashing (DVStH) method to learn compact binary codes for image retrieval. Unlike most existing deep hashing methods which use a series of convolution and fully-connected layers to learn binary features, we develop a probabilistic framework to infer latent feature representations inside the network. Then, we design a struct layer, instead of a bottleneck hash layer, to obtain the binary codes through a simple encoding procedure. By doing these steps, we are able to obtain binary codes discriminatively and generatively. To make it applicable to cross-modal multimedia retrieval, we extend our method to a cross-modal deep variational and structural hashing (CM-DVStH). We design a deep fusion network with a struct layer to maximize the correlation between image-text input pairs during the training so that a unified binary code can be obtained. We then design modality-specific hashing networks suitable for the out-of-sample extension. Here, we train a network for each modality which outputs a latent representation that is as close as possible to the binary codes inferred from the fusion network. Experimental results on five benchmark datasets are presented to show the efficacy of the proposed approach.

Figure 5.1: Different hashing network architectures (a)-(c) in comparison with our proposed network (d). The base network learns the abstract features of data. The *fc* is the fully-connected layers. y is the label information. C is the classification layer. K is the number of bits.

5.1 Introduction

Recent deep hashing networks [49, 54, 56, 63, 66] consist of a base network and a hash layer trained under a specific loss function. The base network is the series of convolution, pooling and fully-connected (*fc*) layers, which learns the abstract features from the high-dimensional feature data; the hash layer outputs the representative feature vector that is

binarized through sigmoid-thresholding to obtain the final binary code; and the loss function is the cost measured to optimize and train the parameters of the network. Figure 5.1(a) illustrates the basic framework of a deep hashing network.

There are currently two shortcomings of this framework. First, while current deep hashing frameworks can learn nonlinear features, it is not modelled based on a probabilistic framework which may limit its flexibility in producing general latent features and capturing diversity from underlying training samples [118, 145]. Deep generative models with variational inference address this by combining the strengths of deep neural networks and probabilistic reasoning which have shown effective in prediction [145] and may be advantageous for scalable hash code learning [121]. Second, a hash layer is a bottleneck since the number of nodes is limited to the required binary code length of K bits. Particularly, if a retrieval system can only afford 8 bits for search, then the deep hashing network would have an 8-dimensional hash layer. This makes training through back-propagation difficult. This can be seen in Figure 5.2 with a classification problem on an AlexNet on the CIFAR-10 dataset where we vary the dimension of a bottleneck layer located before the classification layer. As can be seen, doubling the bottleneck layer length leads to significant improvements in accuracy mostly in lower dimensions, which is typically the range for the binary codes. Given these findings, we introduce two key features of our hashing model:

- (i) We extend the typical hashing network into a deep generative framework. Figure 5.1(b) illustrates a hashing network with a variational architecture, where we treat the whole network as an encoder-decoder problem. The encoder consists of a series of convolution and fc layers and a stochastic layer to sample a latent output from high-dimensional samples. The stochastic layer is designed such that the latent output comes from variational distribution parametrized by a probabilistic model (defined with parameters μ and σ^2) which gives strong generability to the model. The

Figure 5.2: Toy example of varying bottleneck layer length in the test accuracy for the CIFAR10 classification problem.

decoder maps the latent output into class labels, y . This encoder-decoder framework forms a deep network trained end-to-end. To the best of our knowledge, we are the first to utilize this kind of framework for both image retrieval and cross-modal retrieval research.

- (ii) We replace the hash layer with a struct layer to gather more information during training. Figure 5.1(c) illustrates a hashing framework using a struct layer. As can be seen, the struct layer comprises multiple blocks where each block would represent a number of binary codes. If each block would have 4 nodes to represent 2 bits, our network for a 16-bit search system would have a 32-dimensional struct layer length instead of 16-dimensional hash layer. Since the struct layer is wider (more nodes) than the hash layer, more information can be exploited during training. During testing, instead of thresholding each node of a hash layer, the output of each block is quantized and concatenated together to obtain the final binary code.

Figure 5.1(d) shows the network which combines both the two key contributions of our DVStH method. A more detailed representation is seen in Figure 5.3. In the next

Figure 5.3: Basic idea of our proposed DVStH method for image retrieval. We use a base network which consists of a series of convolution, pooling and fc layers to obtain a representative feature vector. This feature vector is passed through a variational block to obtain a probabilistic latent representation which is then passed through a struct layer. Our network is trained end-to-end by two objective functions. First, the output of the struct layer is optimized such that it can minimize a classification loss. Second, the latent variable is modelled such that approximated posterior distribution in the form of Multivariate Gaussian is close to prior regularized by the Kullback-Liebler divergence (KLD) criterion. During retrieval, given a query, we extract the query binary code by getting the output of the struct layer and perform block quantization.

section, we first present the DVStH method for single-modality hashing. We then present the proposed CM-DVStH method for cross-modality hashing.

5.2 Deep Variational and Structural Hashing

Let $\mathcal{S} = \{\mathcal{X}, \mathcal{Y}\}$ be our training set with $\mathcal{X} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_N\}$, $\mathbf{X}_i \in \mathbb{R}^{h \times w \times 3}$ as the RGB images with height h and width w . The images would have a corresponding ground truth label $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N\}$, $\mathbf{y}_i \in \{0, 1\}^{C \times N}$, where $\mathbf{y}_i = [y_{i,1}, y_{i,2} \dots y_{i,C}]$ such that $y_{i,c} = 1$ if $\mathbf{X}_i$ belongs to class c and 0 otherwise. Our DVStH aims to learn a series of hash

functions to obtain K -bit compact binary vectors:

$$\mathcal{F}_{\mathcal{X}} : \mathbb{R}^{h \times w \times 3 \times N} \rightarrow \{0, 1\}^{K \times N} \quad (5.1)$$

which exploit the label information and preserve the semantic relation between samples.

We achieve this by designing an end-to-end deep hashing network, parameterized by $\Theta = \{\theta_{base}, \theta_{var}, \theta_{struct}, \theta_{class}\}$, which optimizes a weighted hybrid loss consisting of a classification loss and a variational encoder loss. By doing so, we can have binary codes that are both discriminative and general for unseen samples. Our network consists of a base network, $\mathbf{x}_i = \phi_{base}(\mathbf{X}_i, \theta_{base})$, that performs a series of convolution, pooling and fully-connected layers. The output of this base network would be the mid-level feature vector $\mathbf{x}_i$. This feature vector is fed to a variational block, $\mathbf{z}_i = \phi_{var}(\mathbf{x}_i, \theta_{var})$ to obtain a latent representation based on an approximated probabilistic distribution that follows some prior distribution. The latent representation is passed to a struct layer, $\mathbf{s}_i = \phi_{struct}(\mathbf{z}_i, \theta_{struct})$, which consists of struct blocks such that a softmax function on each block output. The struct layer output is projected to a class layer, $o = \phi_{class}(\mathbf{s}_i, \theta_{class})$, to get the target scores. These target scores are optimized by the classification criterion to learn all the network weight parameters. During binary code extraction, the block outputs of the struct layer are quantized to binary block codes and are finally concatenated for storage and retrieval. Each part is specifically discussed below.

Variational Block: Inspired by the success of variational encoders [118], we employ a probabilistic interpretation to the hashing network. VAE was proposed by Kingma [118], where a parametric generative model is implemented in a deep structure by using a probabilistic approach to sample latent representations. This underlying generative process would enhance the learning of the whole network structure and make it more general,

suitable for out-of-sample extension. The latent variable is modelled by an approximate inference model given the visible variables and a prior distribution.

Particularly, we assume that given the output feature from the base network, $\mathbf{x}_i$, a latent representation, $\mathbf{z}_i$, can be defined by a posterior distribution, $p_\theta(\mathbf{z}_i|\mathbf{x}_i)$. We identify a proposal distribution $q(\mathbf{z}|\mathbf{x}_i)$ to approximate this posterior distribution to be normally distributed defined as:

$$q(\mathbf{z}_i|\mathbf{x}_i) = \mathcal{N}(\mathbf{z}_i|\mu_i, \sigma_i^2 \mathbf{I}) \quad (5.2)$$

By doing a re-parameterization trick [118], we can sample $\mathbf{z}_i$ as:

$$\begin{aligned} \mathbf{z}_i^l &= \mu_i + \sigma_i \odot \epsilon^l \\ \epsilon^l &\sim \mathcal{N}(0, 1) \end{aligned} \quad (5.3)$$

where ϵ^l means the l th sample of noise, $\odot$ denotes element-wise multiplication, μ_i and σ_i would be the output of the nonlinear projection from the hashing network. With (5.3), we make the latent representation differentiable and capable of back-propagation.

We then assume that the proposal distribution should follow a prior distribution over the latent variable defined as a Multivariate Gaussian:

$$p_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, \mathbf{I}) \quad (5.4)$$

We can enforce this assumption by using the Kullback-Liebler divergence (KLD) derived as:

$$\begin{aligned} \mathcal{L}_{KLD} &= -KLD(q(\mathbf{z}_i|\mathbf{x}_i)||p_\theta(\mathbf{z})) \\ &= \frac{1}{2} \sum_{j=1}^J (1 + \log((\sigma_i^{(j)})^2) - (\mu_i^{(j)})^2 - (\sigma_i^{(j)})^2) \end{aligned} \quad (5.5)$$

where j is the j th element of μ and σ . The KLD would act as a regularizer to the proposal distribution.

Unlike the VAE, our work uses supervised information to map our modelled latent values to semantic values suitable to be encoded as representative binary codes. The encoder part of our hashing network would be from the raw data input to the latent variable ($\mathbf{X}_i \rightarrow \mathbf{z}_i$). The decoder part would be the latent variable to the label input ($\mathbf{z}_i \rightarrow \mathbf{y}_i$).

Struct Layer: From the latent variable, we impose structure in the succeeding fc layer by splitting it into M blocks such that ϕ_{struct} is parameterized by $\theta_{struct} = \{\theta_{struct}^{(m)}\}_{m=1}^M$. Each block would project the latent sample into a distinct semantic representation. Each struct block vector is represented as:

$$\mathbf{s}_{m,i} = \phi_{struct}^{(m)}(\mathbf{z}_i, \theta_{struct}^{(m)}) \quad (5.6)$$

where $\phi_{struct}^{(m)}$ is the nonlinear projections done on $\mathbf{z}_n$ for the m th block. Suppose each struct block, $\mathbf{s}_{m,n}|_{m=1}^M$, is of length S , the shared struct layer output would be:

$$\mathbf{s}_i = [\text{softmax}(s_{1,i}), \text{softmax}(s_{2,i}), \dots, \text{softmax}(s_{M,i})] \quad (5.7)$$

where $\mathbf{s}_i \in \mathbb{R}^{1 \times MS}$. Softmax is applied for each block output to help maximize the potential of one element on each block. This is important since each block would be one-hot encoded during binarization. A softmax function would prevent any approximation loss during encoding.

Classification Loss: Our end-to-end deep network is trained such that the struct layer output would be efficient in solving the classification problem, which would lead to encoding optimal and representative binary codes. The output of the struct layer is fed into the class layer of length C classes, parameterized by θ_{class} . We can then solve a cross-entropy

Figure 5.4: Visual representation of how the discriminative loss for multi-label target data is obtained. The hash layer output is projected to a $2 \times C$ -dimensional class layer which has a + and - class layer that represents the probability that input samples do and do not belong to the class label c , respectively. For each node that represents a label c , softmax function is applied and binary cross-entropy (BCE) loss is computed. This is done for all classes and summed to obtain the total loss

loss given as:

$$\begin{aligned} \mathcal{L}_{class} = & - \sum_{i=1}^N \sum_{c=1}^C y_{i,c} \log(\phi_{class}^c(\mathbf{s}_i, \theta_{class})) \\ & + (1 - y_{i,c}) \log(1 - \phi_{class}^c(\mathbf{s}_i, \theta_{class})) \end{aligned} \quad (5.8)$$

where ϕ_{class}^c is the softmax output of the class layer for class c . For multi-label classification where each sample is represented by one or more labels, our class layer would have a length of $2 \times C$ and solve the binary cross-entropy for each class independently given as:

$$\begin{aligned} \mathcal{L}_{class(mult)} = & \sum_{i=1}^N \sum_{c=1}^C \ell_{bce}(\phi_{class}^{c,+}(\mathbf{s}_i, \theta_{class}), y_{i,c}) \\ & + \ell_{bce}(\phi_{class}^{c,-}(\mathbf{s}_i, \theta_{class}), 1 - y_{i,c}) \end{aligned} \quad (5.9)$$

where

$$\ell_{bce}(p, y) = -(y \log(p) + (1 - y) \log(1 - p)) \quad (5.10)$$

$\phi_{class}^{c,+}(\mathbf{s}_i, \theta_{class}) \in [0, 1]$ and $\phi_{class}^{c,-}(\mathbf{s}_i, \theta_{class}) \in [0, 1]$ denote the scalar probabilities when the model predicts that the i th training sample belongs or does not belong the class label c , respectively. This is obtained by taking the sigmoid activation of the class layer output and applying a binary softmax on each class node as shown in Figure 5.4.

Overall Loss: The overall formulation of our method is as follows:

$$\min_{\theta} \mathcal{L} = \mathcal{L}_{class} + \eta \mathcal{L}_{KLD} \quad (5.11)$$

$\mathcal{L}_{class}$ ensures that the target scores (output of the class layer) are similar as much possible with the label information. $\mathcal{L}_{KLD}$ ensures that the KLD between the proposed distribution and prior distribution for the latent variable is minimized. η is a hyper-parameter to balance both terms. To solve this optimization problem, we perform the standard batch-wise gradient descent. **Algorithm 6** summarizes the detailed procedure of our DVStH.

Binarization: During testing, we use the struct layer output to obtain the binary codes. Given M struct blocks, the binary code for each block can be obtained as follows:

$$\mathbf{b}_{m,i} = \text{BIN}(g(\mathbf{s}_{m,i})) \quad (5.12)$$

and the final binary code is $\mathbf{b}_i = [\mathbf{b}_{1,i}, \mathbf{b}_{2,i}, \dots, \mathbf{b}_{M,i}] \in \mathbb{R}^K$. $g(\cdot)$ encodes the struct block output to a one-hot vector, and $\text{BIN}(\cdot)$ quantizes¹ the one-hot vector to a reduced binary code such that $b_{m,i} \in \{-1, 1\}^{\log_2(S)}$. $\text{BIN}(\cdot)$ is implemented by taking the binary form of the index having the highest value in the one-hot vector (argmax). For example, given a block of size 4 with an argmax value of 3, the binary form is given as '11'. This is done for

¹The struct layer is carefully designed such that the length of each block would be of a power of 2.

Algorithm 6: DVStH - deep variational and structural hashing

Input: Training set $\{\mathcal{X}, \mathcal{Y}\}$, network learning parameters (learning rate, momentum, optimizer, etc), objective function parameter η .

Output: Network parameters Θ

Step 1 (Initialization):

- Initialize deep hashing network parameters

Step 2 (Network Learning):

for $t = 1, 2, \dots, Epoch$ **do**

for *each mini-batch* **do**

for *each i sample in mini-batch* **do**

 - Get base network output $\mathbf{x}_i$

 - Split output to μ_i and σ_i .

 - Sample $\mathbf{z}_i$ from (5.3).

 - Get struct output, $\mathbf{s}_i = \phi_{struct}(\mathbf{z}_i)$

 - Obtain target scores from $o_i = \phi_{class}(\mathbf{s}_i, \theta_{class})$

end

 - Solve for $\mathcal{L}_{class}$ and $\mathcal{L}_{KLD}$ using (5.8)/(5.9) and (5.5).

 - Obtain the top-layer gradients

 - Perform back propagation for the whole network: $\Theta \leftarrow -\Delta_{\Theta}(\mathcal{L}_{class} + \eta\mathcal{L}_{KLD})$

end

end

Return: Θ .

M blocks. In this case, the length of the binary code would be $K = M \log_2(S)$. Figure 5.5 shows an illustration on how to obtain the K -bit binary code from the struct layer output of M blocks with length S .

While the SuBiC [68] method also consists of a struct layer, the binary code is represented by one-hot encoded vectors from each block. These vectors are used to measure the similarity through their asymmetric distance. Differently, we are able to exploit more information by binarizing each one-hot encoded vector such that more representative nodes are trained. While DQN [146] and DVSQ [147] also perform quantization for retrieval, their methods are more computationally complex since the network output is mapped to different clusters from a lookup table to determine the binary codes activated. Additionally, they also use the Asymmetric Quantizer Distance (AQD) [4] to measure the similarity between samples, which is more costly than our Hamming distance metric.

Figure 5.5: Binary code encoding in DVStH. To obtain the binary code given an input, we obtain the struct layer output and perform one-hot encoding for each struct block. Each one-hot vector is then transformed to binary form. In this example, the number of blocks is set to $M = 3$, the number of nodes per block is set to $S = 4$, and the number of bits is set to $K = 6$. $\otimes$ symbol is defined as concatenation. The darkness of the node represents the probability score where a completely black node represents an output of 1.

5.3 Cross-Modal Deep Variational and Structural Hashing (CM-DVStH)

Our DVStH can be extended for cross-modal retrieval. Unlike single-modal retrieval where both the query example and the database are from the same modality, the key idea of cross-modal retrieval is to retrieve samples from another modality which is different from that of the query example but shares similar semantics (e.g. image and text). This is challenging as different modalities should be transformed to a common subspace such that the modality gap is reduced. Figure 5.6 illustrates the basic idea of the proposed approach. We propose a two-stage end-to-end deep architecture for cross-modal hashing. First, we train a cross-modal fusion hashing network to extract unified binary codes such that we are able to implicitly maximize the semantic correlation between the two modalities given image-text

Figure 5.6: Basic idea of our proposed CM-DVStH for cross-modality multimedia retrieval. Given a gallery set represented by two modalities (image and text), we learn a fusion hashing network and modality-specific networks: First, we train a fusion network by exploiting the correlation of the cross-modal input and solving a classification-based criterion. Once the fusion network is learned we can use it to infer the binary codes which are used for learning the modality-specific networks. Second, we learn modality-specific hashing networks (one for each modality) such that a latent representation is modelled based on two criteria: (1) given the image-text pair, the latent variable is forced to be as similar as possible to the inferred binary code from the fusion network through a negative log-likelihood criterion. (2) the latent representation is also modelled such that approximated posterior distribution in the form of Multivariate Gaussian is close to a prior regularized by the KLD criterion. During retrieval, given a query, we extract the query binary code using the learned modality-specific hashing network and obtain the most similar binary codes from the gallery, indexed to retrieve the most relevant images.

training data pairs and its corresponding label information. We perform this in a *discriminative* manner by using a struct layer trained under a classification-based loss. Second, we model the modality-specific hashing networks such that the output latent variable is similar to the binary code inferred from the fusion network via a negative log-likelihood criterion. This latent variable is obtained in a *probabilistic* manner where it is sampled based on an approximate posterior distribution regularized by a prior through a KLD criterion. We now detail these networks as follows:

5.3.1 Cross-Modal Fusion Network

Let $\mathcal{X}_u = [\mathbf{X}_{u,1}, \mathbf{X}_{u,2}, \dots, \mathbf{X}_{u,N}] \in \mathbb{R}^{d_u \times N}$ and $\mathcal{X}_v = [\mathbf{X}_{v,1}, \mathbf{X}_{v,2}, \dots, \mathbf{X}_{v,N}] \in \mathbb{R}^{d_v \times N}$ be the training sets from different modalities, where u and v represent two different modalities, N is the number of training samples in each modality, and d_u and d_v are the feature dimension for each sample in modalities u and v , respectively. Our fusion network aims to transform the cross-modal sample pair into a compact binary feature vector as follows:

$$f_{u,v} : (\mathbb{R}^{d_u}, \mathbb{R}^{d_v}) \rightarrow \{0, 1\}^K \quad (5.13)$$

where K is the length of the binary feature vector. Given image and text as the modality pairs, the fusion network would comprise of image and text networks as branch networks, fused into a latent network. The image base network, $\phi_{base}^u(\mathbf{X}_{u,i}, \theta_{base}^u)$ consists of convolution, pooling layers and fc layers. The text base network, $\phi_{base}^v(\mathbf{X}_{v,i}, \theta_{base}^v)$ consists of a series of fc layers². The latent network, $\phi_w(\mathbf{x}_u, \mathbf{x}_v, \theta_w)$, comprises of a fusion operation³ and series of fc layers:

$$\mathbf{w}_i = s(\phi_{base}^u(\mathbf{X}_{u,i}, \theta_{base}^u) \otimes \phi_{base}^v(\mathbf{X}_{v,i}, \theta_{base}^v)) \quad (5.14)$$

$$\mathbf{h}_i = \phi_w(\mathbf{w}_i, \theta_w) \quad (5.15)$$

where $s(\cdot)$ is the nonlinear activation function, $\otimes$ is the concatenation operation, and $\mathbf{h}_i$ is the output of the fusion network. We then feed the output of the latent network to a struct layer where each struct block output would be obtained as follows:

$$\mathbf{s}_{m,i} = \phi_{struct}^{(m)}(\mathbf{h}_i, \theta_{struct}^{(m)}) \quad (5.16)$$

²A more comprehensive text network can be designed to obtain more representative text features. For simplicity, our paper uses a network with fully-connected layers.

³Based on empirical analysis, we see that a concatenation operation seem to provide better performance than summation.

Algorithm 7: CM-DVStH - cross-modal fusion network

Input: Training set $\mathcal{X}_u$ and $\mathcal{X}_v$, network learning parameters(learning rate, momentum, optimizer, etc).

Output: fusion network parameters, $\Theta_{fuse} = \{\theta_{base}^u, \theta_{base}^v, \theta_w, \theta_{struct}\}$

Step 1 (Initialization):

Initialize fusion network parameters

Step 2 (Fusion Network Learning):

for $t = 1, 2, \dots, Epoch$ **do**

for each mini-batch **do**

 - Get w_i according to (5.14)

 - Get struct output using (5.16)

 - Obtain target scores according to $o_i = \phi_{class}(\mathbf{s}_i, \theta_{class})$

end

 - Solve for $\mathcal{L}_{class}$ using (5.8)/(5.9)

 - Obtain the top-layer gradients

 - Perform back propagation for the whole network:

$\Theta_{fuse} \leftarrow -\Delta_{\Theta_{fuse}}(\mathcal{L}_{class})$

end

Return: Θ_{fuse} .

Similar to DVStH, we optimize the network by setting the struct output to be effective in solving the classification problem using the cross-entropy loss in (5.8) and (5.9). We employ the batch-wise gradient descent method to learn parameters of the fusion network.

Algorithm 7 summarizes the detailed procedure of the cross-modal fusion network of our CM-DVStH.

5.3.2 Modality-Specific Networks

After learning a fusion network, we can then infer representative binary codes for training the modality-specific networks for encoding out-of-sample input. The aim of modality-specific networks is to directly map each cross-modal sample pair into its corresponding binary code inferred from the fusion network:

$$g_u : \mathbb{R}^{d_u} \rightarrow \{0, 1\}^K, \quad g_v : \mathbb{R}^{d_v} \rightarrow \{0, 1\}^K \quad (5.17)$$

Each modality-specific network consists of a base network, variational block and hash layer parameterized by $\theta^* = \{\theta_{base}^*, \theta_{var}^*, \theta_{hash}^*\}$ ⁴. We obtain latent variable $\mathbf{z}_i^*$ similarly extracted using (5.3) and is fed to a hash layer, $\mathbf{h}_i^* = \phi_{hash}(\mathbf{z}_i^*, \theta_{hash}^*)$, to obtain a real-valued hash code. In order to ensure that the latent real-valued hash code would be similar to the learned binary codes in the fusion network, we interpret the real-valued hash code as a posterior probability as follows:

$$p(b_{k,i} | h_{k,i}^*) = \begin{cases} \varsigma(h_{k,i}^*) & \text{if } b_{k,i} = 1 \\ 1 - \varsigma(h_{k,i}^*) & \text{if } b_{k,i} = 0 \end{cases} \quad (5.18)$$

where ς is a sigmoid function and k is the k th bit of the binary code. From these approximations, the network learning formulation can then be written as:

$$\begin{aligned} \min_{\theta^*} \mathcal{L} &= \sum_{i=1}^N \sum_{k=1}^K \mathcal{L}_{NLL} + \sum_{i=1}^N \eta \mathcal{L}_{KLD} \\ &= - \sum_{i=1}^N \sum_{k=1}^K b_{k,i} h_{k,i}^* + \log(1 + e^{h_{k,i}^*}) \\ &\quad - \frac{\eta}{2} \sum_{i=1}^N \sum_{j=1}^J (1 + \log((\sigma_i^{(j)*})^2) - (\mu_i^{(j)*})^2 - (\sigma_i^{(j)*})^2) \end{aligned} \quad (5.19)$$

$\mathcal{L}_{NLL}$ ensures that the binary data likelihood under the approximate posterior distribution is maximized. $\mathcal{L}_{KLD}$ ensures that the KLD between the proposed distribution and prior distribution for the latent variable is minimized. Finally, η is a constant parameter to balance the two loss terms. (5.19) can be easily optimized by taking the gradient of the objective function and performing batch-wise back-propagation. **Algorithm 8** summarizes the detailed procedure of the modality-specific networks of our CM-DVStH.

For new instances or query data, we simply use the learned modality-specific networks to obtain the output real-valued codes and finally binarize them using the $\text{sgn}(\cdot)$ function. During retrieval, given a text/image query, we extract the query binary code using the

⁴where $*$ = $\{u, v\}$ which represents each modality.

Algorithm 8: CM-DVStH - modality-specific network

Input: Training set $\mathcal{X}_u$ and $\mathcal{X}_v$, fusion network parameter Θ_{fuse} , network learning parameters, objective function parameter η

Output: Network parameters Θ_u and Θ_v

Step 1 (Initialization):
 - Initialize modality-specific network parameters

Step 2 (Modality-Specific Hashing Network Learning):
for $t = 1, 2, \dots, Epoch$ **do**
 for each mini-batch do
 for each n sample in mini-batch do
 - obtain $\mathbf{s}_i$ according to (5.16)
 - obtain $\mathbf{b}_i$ according to (5.12)
 for $* = image(u), text(v)$ **do**
 - obtain output from base network
 - Split output to μ_i^* and σ_i^* .
 - Sample $\mathbf{z}_i^*$ similar to (5.3).
 - Get $\mathbf{h}_i^* = \phi_{hash}(\mathbf{z}_i^*, \theta_{hash}^*)$.
 end
 end
 - Solve for $\mathcal{L}_{NLL}$ and $\mathcal{L}_{KLD}$ according to (5.19).
 - Obtain the top-layer gradients
 - Perform back propagation for the whole network: $\Theta_* \leftarrow -\Delta_{\Theta_*}(\mathcal{L}_{NLL} + \eta\mathcal{L}_{KLD})$
 end
end
Return: $\{\Theta_u, \Theta_v\}$.

learned text/image hashing network and obtain the most similar binary codes from the gallery. These gallery codes would be indices to retrieve the most relevant images in the database.

Difference with CMDVH: Our CM-DVStH is an extension based on our previous model called Cross-Modal Deep Variational Hashing (CMDVH) [32]. Unlike CMDVH that has a fusion network which learns the unified binary code through an iterative and alternate optimization procedure with a hinge loss criterion, our CM-DVStH uses an end-to-end gradient optimization with a cross-entropy loss. By doing so, training is faster and less memory intensive. Additionally, the fusion network consists of struct layer which is wider than the hash layer, which would process more information during optimization.

Block-wise binarization is done in the struct layer to obtain the final binary codes.

5.4 Experiments

We conducted experiments on five widely used benchmark datasets in multimedia retrieval to evaluate our DVStH and CM-DVStH methods. Particularly, we conducted experiments on three datasets (CIFAR10, ImageNet, NUS-WIDE) to evaluate our DVStH method for image retrieval and three datasets (MIRFLICKR25k, IAPRTC12, NUS-WIDE) to evaluate our CM-DVStH method for cross-modal retrieval. The following describes the details of the experiments and results.

5.4.1 Datasets and Experimental Set-up

5.4.1.1 Datasets

CIFAR-10 dataset⁵ [148]: consists of 60000 32×3 color images annotated from 10 object classes. This dataset is split into a *train* and *test* set with 50000 and 10000 samples, respectively. From the *test* set, we randomly selected 1000 samples, 100 per class, as the query samples, and the remaining is used as gallery set. The *train* set is used for training the network.

ImageNet dataset⁶ [149]: contains over 1.2 million images in the *train set* and 50000 images in the *validation set* from 1000 object categories extracted for the Large Scale Visual Recognition Challenge (ILSVRC). We used the *train* set for training the deep model and split the *validation* set randomly and selected 2000 images as query samples and the remaining images as gallery set. As suggested by [150], we also did a cross-domain experiment such that we tested our ImageNet-trained network with retrieval experiments using

⁵<https://www.cs.toronto.edu/~kriz/cifar.html>

⁶<http://www.image-net.org/challenges/LSVRC/2012/>

the Pascal VOC 2007⁷ [127] and Caltech101⁸ dataset. By doing so, we can show that our cross-modal network is flexible for retrieval on different datasets of varying classes. For both Pascal and Caltech101 dataset, we randomly selected 1000 images as query and remaining images as gallery set.

NUS-WIDE Dataset⁹ [129]: contains 269648 images, which are collected from the popular photo-sharing and storage website, Flickr, and is annotated with 81 concepts. This dataset contains a large number of images and more diverse concepts which make the retrieval task more difficult. Following the same setting in [146, 147], we used images that contain at least one of the 21 most frequent concepts, where each concept would consist of at least 5000 images. In our case, a total of 166391 images were successfully downloaded and used in our experiment. We randomly selected 5000 images as query samples and took the remaining as gallery. From the gallery set, we randomly selected 10000 images as training samples. For cross-modal retrieval experiments, we followed the same setting as [21, 87] where we randomly selected 100 pairs per class as the query set, 500 pairs per class as the training set. We used pre-computed tag occurrence vectors as text features by the dataset providers.

MIRFLICKR25k dataset¹⁰ [142]: contains 25000 images extracted from one million images also from the Flickr website and annotated with 38 concepts with 24 concepts (bird, tree, people, indoor, sky, night), while the remaining 14 concepts have stricter labelling which are only annotated when the concepts seen are very salient to the image. For the text features, we extracted a 1386-dimensional bag-of-words (BoW) representation. Following the same settings in [21, 87], 1000 images were selected in random as query samples, 5000 for training, and the remaining images as gallery.

⁷<http://host.robots.ox.ac.uk/pascal/VOC/voc2007/>

⁸<http://www.vision.caltech.edu/ImageDatasets/Caltech101/>

⁹<http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm>

¹⁰<http://press.liacs.nl/mirflickr/>

IAPRTC12 dataset¹¹ [143]: contains 19627 images with corresponding sentence descriptions. These image-sentence pairs present various semantics such as landscape, action, and people. Similar to [20], we used the top 22 frequent labels from the 275 concepts obtained generated from the segmentation task¹². For the text features, we pre-processed the sentence data removing the *stop words* and extracted a 500-dimensional bag-of-words (BoW) representation. We randomly selected 100 pairs per class as the query set, 4000 images as the training set, and the remaining data as the gallery set.

5.4.1.2 Evaluation Metrics

For the single-modal retrieval experiments, we mainly used the mean average precision (mAP), which computes the mean of all queries' average precision, AP , defined as follows:

$$AP = \frac{1}{M} \sum_{r=1}^R prec(r) \cdot rel(r) \quad (5.20)$$

where M is the number of relevant instances in the retrieved set, $prec(r)$ denotes the precision of the top r retrieved set, and $rel(r)$ is an indicator of relevance of a given rank (which is set to 1 if relevant and 0 otherwise). Here, two samples are similar as long as they share at least one similar label. Additionally, the NUS-WIDE have multiple labels for each sample so it is important that a ranking metric is also evaluated. Hence, we used the Normalized Discounted Cumulative Gain (NDCG) and Average Cumulative Gain (ACG) metric. For a given query sample $\mathbf{x}_q$, these criteria are defined as follows:

$$NDCG@p = \frac{1}{Z} \sum_{i=1}^p \frac{2^{r_i} - 1}{\log(1 + i)} \quad (5.21)$$

$$ACG@p = \frac{1}{p} \sum_{i=1}^p r_i \quad (5.22)$$

¹¹<http://imageclef.photodata>.

¹²<http://imageclef/SIAPRdata>.

where Z is the normalized constant and p is the number of retrieved samples in the ranking list. r_i represents the similarity level valued z if the query and i th sample in gallery share z similar labels, and valued zero if they do not share any label. The NDCG evaluates the ranking by penalizing errors in higher ranked items more strongly while ACG takes the average of the similarity levels of data within the retrieved samples.

For cross-modal retrieval experiments, we performed image-text retrieval and text-image retrieval, which search texts from a query image and search images from a query text, respectively. To measure the mAP for our experiments, we followed the same set-up as [20] and [21] where we set $R = 50$ for the MIRFLICKR25k/NUS-WIDE and $R = 500$ for the IAPRTC12. Since all three databases contain multiple labels for each sample, the NDCG and ACG criterion are also evaluated.

5.4.1.3 Network Configuration

Our deep architecture and experiments were implemented under the Pytorch framework¹³. We used different network configurations for different experiments to have a fair comparison with the compared methods:

- (i) **Single-Domain Single-Modality Retrieval:** Similar to [56], we constructed an image network with three convolution layers with 5×5 filters with dimensions of 32, 32 and 64. It is followed by a fully-connected layer with 500 nodes. Our variational layers (μ -layer and σ -layer) would have 500 nodes and a struct layer of $[500 \rightarrow 300 \rightarrow MS]$ with $S = 4$.
- (ii) **Cross-Domain Single-Modality Retrieval:** Similar to [68], we used the VGG-128 [126] as base network. Particularly, we followed the five convolution layers

¹³<http://pytorch.org/>

and fully connected layers up until the fc7 layer with a length of 4096 dimensions. Our variational layers would have a length of 4096 dimensions, and a struct layer of $[4096 \rightarrow 128 \rightarrow MS]$ with $S = 4$.

- (iii) **Single-Domain Single-Modality Multi-label Retrieval:** Similar to [146, 147], we used the AlexNet [104] as base network. Particularly, we followed the convolution layers and fully connected layers up until the fc7 layer with a length of 4096 dimensions. Our variational layers would have a length of 4096 dimensions, and a struct layer of $[4096 \rightarrow 128 \rightarrow MS]$ with $S = 2$. Because we do multi-label retrieval, we used a class layer of length $2 \times C$ and used (5.10).
- (iv) **Cross-Modality Retrieval:** the fusion network is designed with a *image hashing network* which used the VGG-net¹⁴ architecture as our initial convolution and pooling layers up to fc7. New fc layers are then added with dimensions of $[4096 \rightarrow 500 \rightarrow 200]$ for all datasets. The *text hashing network* is designed with a series of fully-connected networks and used the pre-processed text features, given by each experiment, as input. We set the fc layers as $[1386 \rightarrow 500 \rightarrow 200]$, $[500 \rightarrow 500 \rightarrow 200]$, and $[1000 \rightarrow 500 \rightarrow 200]$ for the MIRFLICKR, IAPRTC12 and NUS-WIDE dataset, respectively. For the latent network which fuses the output of image and text network, we used fc layers with dimensions of $[200 \rightarrow 500 \rightarrow MS]$ with $S = 4$. For the modality-specific networks, we used the similar image and text networks but with a variational layer of length 200.

For all single-modal experiments, the new fc layers used the ReLU activation as the nonlinear activation function. For the CIFAR experiment, we trained the network from

¹⁴<https://download.pytorch.org/models/vgg11-bbd30ac9.pth>

scratch for 150 epochs and batch size of 128 with an initial learning rate of 0.01 and learning decay of 0.1 at the 100th epoch. Stochastic gradient descent (SGD) is used to update the weights. For the ImageNet and NUS-WIDE experiment, the training procedure is set with a learning rate of 0.0001 and batch size of 64 for 20 epochs. Fewer epochs are applied as the training samples are larger for these datasets. Additionally, the Adam optimizer [151] is used to update the weights.

For cross-modal experiments, the fusion network in CM-DVStH used a learning rate of 0.0001, batch size of 64, epoch size of 50, and the Adam optimizer for weight update. Similarly, the training for each modality-specific network is set with a learning rate of 0.0001, batch size of 64 and epoch size of 30. The epoch sizes are decided empirically based on the convergence of the loss function. For pre-trained base networks, we freeze the learning rate of the bottom layers, mostly in the convolution layers (Conv1 - Conv2). This is so we can avoid ruining the representative abstract features already learned from pre-training.

5.4.2 Experimental Results

5.4.2.1 Comparisons with Single-Modal Deep Hashing methods

We compared our DVStH to several deep hashing methods which include three methods (DRSCH, DSRH, DNNH) that use the triplet-wise supervision, four methods (CNNH+, DSH, BDNN, ADSH) that use pairwise supervision, and two methods (DLBHC, SuBiC) that use point-wise supervision similar to ours. Differently, DLBHC learns a hash layer in a fine-tuned network and use sigmoid thresholding to binarize the output of the hash layer to obtain the binary codes. SuBiC uses one-hot encoded vectors as binary bits and does not include a variational block.

Table 5.1: mAP performance comparison of our DVStH and different deep hashing models for the single-domain category retrieval experiment on the CIFAR10 dataset

Method	12	24	36	48
CNNH+ [52]	0.5425	0.5604	0.5640	0.5574
DLBHC [54]	0.5503	0.5803	0.5778	0.5885
DNNH [55]	0.5708	0.5875	0.5899	0.5904
DSH [56]	0.6157	0.6512	0.6607	0.6755
KSH-CNN [42]	-	0.4298	-	0.4577
DSRH [53]	-	0.6108	-	0.6177
DRSCH [152]	-	0.6219	-	0.6305
BDNN [64]	-	0.6521	-	0.6653
SuBiC [68]	0.6349	0.6719	0.6823	0.6863
ADSH [65]	0.5062	0.6525	0.6902	0.7051
DVStH (Ours)	0.6669	0.6978	0.7131	0.7146

Table 5.1 shows the mAP performance of different deep learning methods compared to our DVStH. The compared results were obtained from [68] except for the ADSH where we performed the experiment from their provided code. We used the same base network for all deep networks. As can be seen, our method outperforms the other deep learning methods. This may be because the other methods use pairwise or triplet similarity information to learn binary codes. In contrast, our method trains from a point-wise discriminative loss which directly learns optimal binary codes. In addition, our method does not require generating several triplets or pairwise samples which would make convergence simpler than methods that require such samples.

5.4.2.2 Comparisons with Single-Modal Deep Hashing methods with Cross-Domain experiments

We also performed cross-domain deep hashing in order to ensure that our method is flexible in partial supervision. We tested our hashing network with samples having unseen classes. In this set-up, we trained our network with classes from the ImageNet database and tested it with two different datasets (Pascal VOC 2007, Caltech101). We compared the performance

Table 5.2: mAP performance comparison of our DVStH and other quantization/hashing-based models for the cross-domain category retrieval experiment where the Pascal VOC2007 and Caltech101 are used as test datasets and ImageNet training set is used for training.¹⁶

Method	VOC2007	Caltech101	ImageNet
DSH [56]	0.4914	0.2852	0.1665
PQ [4]	0.4965	0.3089	0.1650
CKM [5]	0.4995	0.3179	0.1737
LSQ [153]	0.4993	0.3372	0.1882
SuBiC 2-layer [68]	0.5600	0.3923	0.2543
SuBiC 3-layer [68]	0.5588	0.4033	0.2810
DVStH	0.5522	0.4219	0.2826

of our method with three popular quantization methods (PQ, CKM, LSQ) and two deep hashing methods (DSH, SuBiC).

Table 5.2 shows the mAP performance of the different methods for the cross-domain experiments. Compared results were obtained from [68] where we followed the same experimental set-up provided in the paper. We also show in the table the ImageNet single-domain experiment for reference. As can be seen, our method outperforms the other methods in the 64-bit experiment. Specifically, our method has a large performance difference across all datasets compared to the conventional quantization methods which use CNN features. This tells that our learning method is flexible in generating the binary codes for retrieval across various datasets. While the SuBiC method also performs well, most probably due to point-wise supervision similar to ours, our DVStH is still better most notably in the Caltech101 and ImageNet validation set evaluation. This may be due to a more representative approach in extracting the binary codes through a binarization scheme and more general network learning through a probabilistic modelling.

¹⁶ImageNet column is single-domain retrieval experiment where the validation set is used for testing

Table 5.3: mAP performance comparison of our DVStH and deep hashing methods for the multi-label single-modality retrieval experiment on the NUS-WIDE dataset.

Method	8	16	24	32
CNNH [52]	0.586	0.609	0.628	0.635
DNNH [55]	0.638	0.652	0.667	0.687
DHN [154]	0.668	0.702	0.713	0.716
DSH [56]	0.653	0.688	0.695	0.699
DQN [146]	0.721	0.735	0.747	0.752
DVSQ [147]	0.780	0.790	0.792	0.797
ADSH [65]	0.719	0.774	0.790	0.793
DVStH	0.766	0.792	0.813	0.819

Table 5.4: NDCG@100 and ACG@100 performance comparison of our DVStH and different deep hashing methods on the NUS-WIDE dataset.

Method	NDCG@100				ACG@100			
	8	16	24	32	8	16	24	32
PCA-ITQ [41]	0.347	0.402	0.431	0.446	0.780	0.926	0.954	0.979
CCA-ITQ [41]	0.342	0.330	0.165	0.165	0.811	0.797	0.408	0.408
KSH [42]	0.423	0.455	0.482	0.492	0.945	1.011	1.063	1.092
SPLH [38]	0.405	0.449	0.462	0.468	0.892	0.997	1.028	1.041
SDH [8]	0.309	0.375	0.369	0.371	0.741	0.838	0.854	0.868
FastHash [44]	0.385	0.416	0.451	0.477	0.862	0.913	1.011	1.061
DSRH [53]	-	0.415	-	0.470	-	1.160	-	1.300
DPSH [63]	0.402	0.469	0.480	0.496	0.925	1.070	1.098	1.144
DTSH [66]	0.454	0.482	0.493	0.510	1.020	1.100	1.108	1.153
ADSH [65]	0.442	0.483	0.500	0.503	0.968	1.077	1.118	1.111
DVStH	0.480	0.516	0.541	0.548	1.083	1.169	1.228	1.250

5.4.2.3 Comparisons with Single-Modal Shallow/Deep Hashing methods with Single-Domain Multi-label experiments

We also performed multi-label deep hashing to exploit multiple label information in the NUS-WIDE database. We compared our method with several deep and shallow hashing methods. For the conventional hashing methods, we again used the CNN features extracted from the fc7 layer of the AlexNet. All compared methods are supervised except for PCA-ITQ.

Table 5.3 shows the mAP performance of different deep hashing methods compared to our method. The compared results were obtained from [147] except for the ADSH which we implemented given the author’s available code. All the deep hashing models used the same base network. As can be seen, our method shows better performance and is competitive with the DVSQ. The DVSQ employs a quantization scheme but performed a metric-based margin loss to exploit the discriminative information of the samples. Moreover, the DVSQ also performs nearest neighbour mapping with cluster centers in order to quantize the binary codes. This makes it more computationally complex than our binarization procedure.

Table 5.4 shows the NDCG and ACG performance of different deep and shallow hashing methods compared to our method. For all compared methods, we ran the experiment with the code provided by the respective authors except for DSRH where we obtained the results from the author’s paper. As can be seen, our method performs better since the DPSH and DTSH implements a pairwise and triplet-wise metric learning approach, respectively; the DSRH also requires triplets to optimize their ranking criterion and the ADSH learns their network using an asymmetric pairwise loss. This shows that our method is also efficient for rank-based retrieval.

5.4.2.4 Comparisons with Shallow Cross-Modal Hashing methods

We compared our CM-DVStH with different state-of-the-art shallow cross-modal hashing methods which can be grouped to unsupervised (CVH, PDH, LSSH, CMFH) and supervised (CCA-ITQ, SePH, DisCMH)¹⁷. To have a fair comparison, the shallow methods used CNN features extracted at the fc7 layer for the images from the pre-trained model also used by our CM-DVStH method. Table 5.5 shows the mAP performance by Hamming Ranking and Figure 5.7 shows the NDCG and ACG performance. It can be observed that our

¹⁷Authors provided their codes except for DisCMH in which we implemented ourselves.

Table 5.5: mAP performance comparison of our CM-DVStH and different cross-modal hashing methods on different datasets.

		$\mathcal{I} \rightarrow \mathcal{T}$											
		MIRFLICKR25k				IAPRTC12				NUS-WIDE			
Method		16	32	64	128	16	32	64	128	16	32	64	128
CVH [19]		0.543	0.543	0.608	0.612	0.354	0.356	0.359	0.361	0.457	0.456	0.448	0.447
CCA-ITQ [41]		0.627	0.619	0.612	0.621	0.357	0.359	0.364	0.364	0.485	0.495	0.486	0.488
PDH [18]		0.770	0.779	0.803	0.823	0.561	0.585	0.602	0.617	0.613	0.647	0.667	0.693
LSSH [80]		0.734	0.761	0.787	0.785	0.501	0.530	0.555	0.564	0.587	0.610	0.626	0.638
CMFH [81]		0.558	0.558	0.561	0.560	0.343	0.344	0.347	0.347	0.606	0.655	0.674	0.697
SePH - <i>km</i> [82]		0.809	0.823	0.851	0.859	0.581	0.611	0.627	0.645	0.699	0.710	0.710	0.718
DisCMH [83]		0.544	0.544	0.544	0.544	0.342	0.333	0.323	0.266	0.492	0.554	0.462	0.439
CM-DVStH		0.882	0.916	0.928	0.941	0.619	0.744	0.773	0.781	0.797	0.854	0.862	0.869
		$\mathcal{T} \rightarrow \mathcal{I}$											
		MIRFLICKR25k				IAPRTC12				NUS-WIDE			
Method		16	32	64	128	16	32	64	128	16	32	64	128
CVH [19]		0.589	0.577	0.599	0.599	0.355	0.354	0.356	0.356	0.503	0.512	0.514	0.530
CCA-ITQ [41]		0.615	0.608	0.611	0.611	0.357	0.360	0.359	0.361	0.545	0.568	0.568	0.572
PDH [18]		0.749	0.769	0.781	0.781	0.563	0.589	0.604	0.618	0.632	0.665	0.690	0.700
LSSH [80]		0.716	0.744	0.755	0.756	0.466	0.501	0.536	0.554	0.575	0.620	0.621	0.633
CMFH [81]		0.566	0.562	0.566	0.563	0.339	0.339	0.339	0.340	0.639	0.666	0.685	0.697
SePH - <i>km</i> [82]		0.765	0.770	0.786	0.797	0.576	0.608	0.614	0.639	0.634	0.639	0.654	0.656
DisCMH [83]		0.544	0.544	0.544	0.544	0.338	0.348	0.342	0.302	0.533	0.592	0.525	0.522
CM-DVStH		0.788	0.815	0.825	0.8321	0.604	0.689	0.720	0.734	0.698	0.778	0.782	0.789

Figure 5.7: NDCG and ACG performance of different cross-modal hashing methods for the MIRFLICKR and IAPRTC12 database.

method provided the best performance compared to the shallow cross-modal hashing methods. This is because first, our CM-DVStH model is a deep model which can capture the nonlinearities of raw data effectively. Although SePH also performs nonlinear transformations, it was done explicitly through kernels which cannot really maximize the information from raw data. Second, most cross-modal methods use the same similarity weight during learning for sample pairs as long as the pair shares at least one similar label. Differently, ours exploit the label information fully to be able to address the ranking problem during training.

Table 5.6: mAP performance comparison of our CM-DVStH and different deep cross-modal hashing methods on the cross-modal retrieval experiments.

MIRFLICKR25k					
	Method	16	32	64	128
$I \rightarrow T$	DCMH [22]	0.758	0.799	0.815	0.837
	CHN [21]	0.822	0.848	0.878	0.881
	CDQ [87]	0.864	0.862	-	-
	CMDVH [32]	0.753	0.765	0.791	0.793
	CM-DVStH	0.891	0.919	0.933	0.940
$T \rightarrow I$	DCMH [22]	0.701	0.729	0.746	0.770
	CHN [21]	0.775	0.789	0.817	0.826
	CDQ [87]	0.848	0.850	-	-
	CMDVH [32]	0.755	0.751	0.783	0.794
	CM-DVStH	0.798	0.820	0.858	0.878
IAPRTC12					
	Method	16	32	64	128
$I \rightarrow T$	DNNH-C [55]	0.480	0.509	0.526	0.535
	DCMH [22]	0.443	0.491	0.559	0.556
	DVSH [20]	0.570	0.632	0.686	0.724
	TVDB [88]	0.629	0.697	0.731	0.772
	CMDVH [32]	0.528	0.594	0.642	0.640
	CM-DVStH	0.685	0.749	0.773	0.788
$T \rightarrow I$	DNNH-C [55]	0.469	0.484	0.490	0.505
	DCMH [22]	0.486	0.487	0.499	0.541
	DVSH [20]	0.604	0.640	0.681	0.675
	TVDB [88]	0.674	0.678	0.704	0.721
	CMDVH [32]	0.514	0.571	0.602	0.612
	CM-DVStH	0.643	0.719	0.720	0.747
NUS-WIDE					
	Method	16	32	64	128
$I \rightarrow T$	DCMH [22]	0.735	0.763	0.780	0.791
	CHN [21]	0.800	0.815	0.835	0.866
	CDQ [87]	0.850	0.849	-	-
	CMDVH [32]	0.743	0.766	0.757	0.784
	CM-DVStH	0.797	0.854	0.862	0.869
$T \rightarrow I$	DCMH [22]	0.690	0.710	0.736	0.756
	CHN [21]	0.761	0.770	-	-
	CDQ [87]	0.832	0.848	-	-
	CMDVH [32]	0.667	0.729	0.757	0.775
	CM-DVStH	0.697	0.778	0.782	0.789

5.4.2.5 Comparisons with Current Deep Cross-Modal Hashing Methods

We also compared our method with current cross-modal deep hashing methods as shown in Table 5.6. Results from the compared methods were obtained from the respective papers, and we used the same experiment settings as mentioned in the papers.¹⁸ It can be seen that our model has the best results in almost all bit lengths across the three datasets. This may be due to several reasons; The DVSH and CHN both perform end-to-end supervised metric-based network training in the form of cosine hinge loss. DNNH-C performs metric-based network training through triplet-ranking loss. Similarly, the DCMH and TVDB perform supervision through optimizing a loss function based on a pairwise similarity matrix. CDQ minimizes an adaptive cross-entropy loss from image-text pairs. These cross-modal hashing methods may not fully obtain discriminative binary codes from their loss functions since they are limiting it to pairwise and triplet-wise supervision. Differently, our method performs point-wise supervision based on a multi-classification loss.

It is important to note that while TVDB shows competitive performance most particularly in the 16-bit experiment, it implements a more complex text and image network using LSTMs and RPNs. CDQ also performs well but is due to a more finer distance computation through asymmetric computation using real-value and hash codes. This is a quantization-based method which has a trade-off between speed and performance. Particularly, it uses the AQD metric for retrieval which computes the distance between query and database based on a stored look-up table. This is generally more costly than using purely binary codes and measuring the Hamming ranking metric like our method. The AQD is similar to ADC and the timings can be seen in Figure 1.3b. Hence, while CDQ achieves better performance, our method is still better in terms of both performance and speed.

¹⁸Deep cross-modal hashing method's results for IAPRTC12 experiment are from [88], CHN/DCMH results for MIRFLICKR/NUSWIDE experiment are from [21], CDQ results for MIRFLICKR/NUSWIDE experiment are from [87].

Figure 5.8: mAP performance of different variants of our method.

5.4.3 Further Analysis

5.4.3.1 Network Analysis for DVStH

We examined different variants of our DVStH method to understand the importance of different parts of our implementation. First, the DVStH1a implements hash network training without the variational block. In this set-up, we directly pass the output of the base network to the struct layer. This disregards the probabilistic design and makes it more similar to the deep hashing architectures in the literature. Second, DVStH1b implements a hash network without the struct layer. Here, the variational block is done after the fc layer and before the hash layer with K -dim length. Third, DVStH2 implements a hash network with no variational block nor a struct layer. This leads to a base network connected to a final fc layer with K -dim length. In this case, binary bits are obtained by extracting the output of the final fc layer and binarized through the sigmoid-threshold function.

Figure 5.8 shows the performance of these variants on the CIFAR10 and ImageNet. Note that all these methods are trained with same initialization and base network. We see that adding the variational block is helpful in improving the performance of our hashing network, most noticeable at higher bit codes. More importantly, changing the hash layer to a struct layer showed a large performance gap in the ImageNet. This is promising as the

Figure 5.9: mAP performance of varying values of η of our method on the CIFAR10 dataset.

ImageNet is more difficult having more classes and samples. From these observations, we can say that having the variational aspect yields a general interpretation for the feature vector. Also, imposing a block-wise structure through the struct layer addresses the limitation of having a bottleneck hash layer and leads to more informative hash codes.

It is also interesting to see for DVStH that at different datasets, the variational and struct layer can give different levels of contribution. Particularly, for the CIFAR dataset, the variational block contributed more on the performance than the struct layer. For the ImageNet, the struct layer gave more contribution. This is likely due to the fact that ImageNet has more classes to learn than the CIFAR dataset; hence, a struct layer which is wider than a hash layer is needed.

We also investigated the effect of the weight term, η , from (5.11). Figure 5.9 shows the performance at different bits at varying η . It can be seen that the DVStH is relatively robust at a certain range for η but then decreases in performance at $\eta = 10^{-4}$ which signals that increasing the weights further for the $\mathcal{L}_{KLD}$ may not be effective as it overpowers the $\mathcal{L}_{class}$ which makes the network less discriminative. It is also important to note that the $\mathcal{L}_{KLD}$ is derived from squared values which increases its scalar value as compared to the

other loss functions. Hence, a low η is set to balance the scalar values accordingly for efficient optimization.

5.4.3.2 Network Analysis for CM-DVStH

We have investigated variants of our CM-DVStH method to further examine the importance of each aspect of our architecture and learning method. CM-DVStH1a disregards the probabilistic interpretation of the modality-specific network and simply learns the binary codes from a negative log-likelihood loss. CM-DVStH1b ignores the struct layer in the cross-modal fusion network which assumes that simply adding a hash layer to optimize the classification loss would be representative enough to train the network and obtain binary codes. CM-DVStH2 does not include any variational block nor struct layer. Table 5.7 shows the performance of the variants compared to our method in the MIRFLICKR25k and IAPRTC12 databases. It can be observed that a struct layer is still important to maximize the information during training as shown in the dip of performance for CM-DVStH1b and CM-DVStH2. Also, modeling latent variables through a variational block in the modality-specific network is helpful for encoding binary codes, which shows that forming a model with probabilistic properties is suitable for hashing new outside data. Also shown in 5.7 are the results of CM-DVStH1a with more fully-connected layers (2 levels deeper) named “CM-DVStH1a(more layers)”.

While adding more fully-connected layers on CM-DVStH1 may improve its performance and gave better results for high bit lengths by a small margin, but our CM-DVStH is better by a significant margin for low bit lengths (16-bit). This may be because adding more layers with a bottleneck hash layer of 16-bits may just cause overfitting and not helpful in generalizing the weights for the task. It is also important note that CM-DVStH1a with more layers requires more weight parameters which adds to the computational cost.

Table 5.7: mAP performance of different variants of our CM-DVStH method on the MIR-FLICKR25k and IAPRTC12 dataset.

MIRFLICKR25k					
	Method	16	32	64	128
$I \rightarrow T$	CM-DVStH1a(more layers)	0.812	0.872	0.930	0.943
	CM-DVStH1a	0.822	0.898	0.936	0.944
	CM-DVStH2b	0.843	0.863	0.901	0.928
	CM-DVStH2	0.822	0.854	0.906	0.925
	CM-DVStH	0.891	0.919	0.933	0.940
$T \rightarrow I$	CM-DVStH1a(more layers)	0.767	0.772	0.814	0.817
	CM-DVStH1a	0.775	0.817	0.848	0.866
	CM-DVStH1b	0.773	0.792	0.824	0.848
	CM-DVStH2	0.772	0.771	0.811	0.830
	CM-DVStH	0.798	0.820	0.850	0.878
IAPRTC12					
	Method	16	32	64	128
$I \rightarrow T$	CM-DVStH1a(more layers)	0.649	0.757	0.770	0.789
	CM-DVStH1a	0.615	0.744	0.762	0.783
	CM-DVStH1b	0.548	0.723	0.753	0.776
	CM-DVStH2	0.536	0.720	0.760	0.762
	CM-DVStH	0.685	0.749	0.773	0.788
$T \rightarrow I$	CM-DVStH1a(more layers)	0.607	0.713	0.716	0.747
	CM-DVStH1a	0.620	0.707	0.714	0.745
	CM-DVStH1b	0.522	0.678	0.707	0.730
	CM-DVStH2	0.500	0.660	0.710	0.720
	CM-DVStH	0.643	0.719	0.720	0.747

5.4.3.3 Balanced Bits Analysis

One criterion for a good hashing model is when it can generate binary codes such that each bit produces a balance of 1's and 0's as much as possible [26]. Unlike some hashing models [26, 49, 64, 76], our hashing network does not explicitly ensure balanced bits in the loss function. Nevertheless, we empirically show that our loss functions and binarization can still implicitly produce balanced bits which contribute to the retrieval performance. We calculated the mean variance of bits from the large-scale gallery set of the NUS-WIDE experiment for various hashing models. Particularly, we compared it with deep hashing models (ADSH, DPSH) and various shallow hashing models (CCA-ITQ, KSH, FH). Among compared methods, the CCA-ITQ performs rotation on the subspace in order to maximize the variance of each bit. Figure 5.10 shows the mean variance of the different methods

Figure 5.10: Mean variance of compared methods in the gallery set of the NUS-WIDE in comparison with its corresponding mAP performance.

and its corresponding mAP performance. As expected, the CCA-ITQ provides the highest mean variance as its objective function focuses on maximizing the variance of each bit. However, the CCA-ITQ performs linear transformations which may not fully exploit the discriminative information from images during hash code learning. Differently, our method still provides high mean variance comparable to CCA-ITQ while also learning representative codes through deep nonlinear transformations.

5.4.3.4 Complexity Analysis

We measured the time complexity of different methods during testing. This is important as visual search should not sacrifice speed over accuracy. We first measure the encoding time in the query set consisting of 1000 images from the ImageNet database under the 64-bit experiment. Figure 5.11 shows the encoding time measured. As can be seen, hashing methods that include encoding steps, such as our DVStH and SuBiC, would require more time. Also as expected, a lower struct size, S , would require more encoding time as it requires to quantize more blocks. Nevertheless, the additional encoding time seems negligible considering that the feature extraction of the base network contributes approximately 80% of the overall encoding time on the DVStH. In addition, the difference between

our method and that of using only a hash layer (DVStH2) is approximately less than 10%, which is a good trade-off considering our DVStH can achieve better performance as seen in Figure 5.8. Even the encoding time for a shallow method (ITQ) using CNN features is similar to our DVStH.

We also compute the retrieval time of a query sample from a gallery set which composes of 10000 binary codes. Hashing-based methods such as our DVStH and CM-DVStH would use the Hamming Ranking metric where we only use the bitwise XOR operation to obtain the similarity. While quantization-based methods (DQN, CDQ, and DVSQ) use AQD which exploits the inner product distance for the similarity measure. SUBiC also performs some asymmetric distance metric using the query’s real-valued network output. In this 64-bit retrieval experiment, we follow the parameters set by [147] and [68] for the bottleneck layer length, number of subspaces, and number of nodes for each block. Fig. 5.12 shows the retrieval time measured. It can be seen that the Hamming Ranking is approximately 7 times faster than AQD and similar to SUBiC. However, SUBiC requires only M additions and uses real-valued query codes which are larger in memory than binary codes. We also present the speed when using AQD with look-up table search. Here, pre-computed query-specific $M \times K$ look-up table is used to immediately lookup for the distances of each subspace based on the codebook index. While this is much faster than AQD, it requires additional storage on look-up tables for every query sample which results to additional memory cost. Hence, while deep quantization methods have been competitive in terms of accuracy, our method is still suitable for retrieval in terms of overall efficiency.

In terms of training time complexity, because our deep hashing network is applied in an end-to-end fashion, it is expected that more iterations are required for convergence as compared to shallow hashing models which use straightforward hand-crafted features. Nevertheless, our DVStH is a single-stage procedure with point-wise supervision which makes it

Figure 5.11: The encoding time for each image on the ImageNet database. The black line denotes the feature extraction time of the base network.

Figure 5.12: The retrieval time of a query sample on a database.

less expensive than other deep hashing networks which take time to generate pairs or triplets during training to preserve ranking or similarities based on their designed loss functions. Additionally, generating such sample pairs or triplets would increase the optimization and convergence time as it requires to consider several sample combinations.

5.4.4 Discussion

We make the following observations from the above experimental results:

- (i) Our DVStH achieves competitive performance in comparison to other state-of-the-art shallow and deep hashing methods for single-domain, cross-domain, and multi-label image retrieval scenarios. There are three reasons: 1) We learn the network param-

eters discriminatively by getting efficient binary codes from optimizing a classification loss. 2) We perform quantization in the struct layer to obtain the binary codes. The struct layer is wider than the hash layer which would exploit more information during network training. 3) We learn general network features by sampling a latent representation which is modelled by a probabilistic distribution.

- (ii) Our CM-DVStH achieves competitive performance in comparison to other state-of-the-art shallow and deep cross-modal hashing methods for cross-modal retrieval. There are two reasons: 1) We learn a fusion network that maximizes the correlation of both modalities and obtains representative binary codes to be learned for the modality-specific networks. 2) The modality-specific networks directly map input data to representative binary codes and ensure it follows a probabilistic distribution for learning more general network features.
- (iii) Both the variational block and struct layer contributes to the final performance of both our DVStH and CM-DVStH method as shown in network analysis experiments in which we investigated the performance of different variants of our network.

5.5 Chapter Summary

In this chapter, we have proposed a deep variational and structural hashing (DVStH) for multimedia search. Our DVStH method includes variational block and a struct layer to learn more general and discriminative binary codes efficient for single-modal retrieval. Our DVStH is extended to CM-DVStH for cross-modal multimedia search. Our CM-DVStH method learns a fusion network to learn binary codes from cross-modal training pairs. We then learn modality-specific hashing networks, one for each modality, for the out-of-sample extension. Experimental results on five multimedia retrieval datasets have shown

the effectiveness of the proposed methods. Part of the work presented in this chapter has been published in [32].

Chapter 6

Deep Video Hashing

In this chapter, we propose a deep video hashing (DVH) method for video search. Unlike most existing video hashing methods which first extract features for every frame and then use conventional image hashing techniques, our DVH learns binary codes for the entire video with a deep learning framework so that both the temporal and discriminative information can be well exploited. Specifically, we fuse the temporal information across different frames within each video to learn the feature representation under two criteria: 1) the distance between a spatio-temporal feature pair obtained at the top layer is small if they are from the same class, and large if they are from different classes, and 2) the quantization loss between the real-valued features and the binary codes is minimized. We exploit different deep architectures to utilize spatial-temporal information and compare them with single-frame based deep models and state-of-the-art image hashing methods. Experimental results demonstrate the effectiveness of our proposed method.

6.1 Introduction

Because videos provide diverse and complex visual patterns consisting of low-level visual content in each frame as well as high-level structured content across frames, video search has been more challenging than image search. Moreover, each video contains a number

Figure 6.1: Basic idea of our proposed DVH approach. We pass a set of successive image frames with a pre-defined frame number (in this figure, $p = 4$) to the convolution and pooling layers to obtain frame-wise CNN features. Then, we perform temporal fusion in the fully-connected layers. At the final stage of our deep network, we apply the $\text{sgn}(\cdot)$ in the activation outputs and obtain the compact binary codes. This is done at consecutive frame-sets to obtain the binary codes for a single video. During training, the network parameters are learned using back-propagation with a large-margin cost function such that video features that contain similar semantics/label information are close to each other, while video features that are dissimilar are far apart as possible, and a quantization loss criterion such that the real feature codes and binary codes are similar as much as possible.

of image frames that require exhaustive computation between frames. This is impractical when retrieving in large-scale video databases. Hence, how to develop a framework for video search where discriminative information from videos can be well exploited remains an important problem in visual search.

Video hashing for content-based retrieval can be divided into three categories. The first extracts a single representative feature vector from a video and then performs hashing. The second treats each frame as an image and performs image hashing first such that the resulting frame-to-frame hamming distances of two videos are then combined through averaging. The last selects several representative frames and employs image hashing on these selected frames. While these frameworks are straight-forward, they do not exploit the temporal information of videos in the learning stage. Furthermore, the second category

is computationally expensive considering each video would usually have a minimum of 30 to 50 frames. Hence, it is desirable to design a video hashing framework which aims to 1) learn strong binary codes from a video by also leveraging temporal information, and 2) reduce the number of binary codes to represent each video while still preserving the semantic content.

To address these challenges, we propose a Deep Video Hashing (DVH) method for video search. Since Convolutional Neural Networks (CNN) have shown promising performance in several computer vision tasks due to their strong feature representation capability, we employ CNN in our video hashing framework. The basic idea of our approach is shown in Figure 6.1. Specifically, we build an end-to-end deep CNN learning framework which utilizes spatio-temporal information after the stacked convolution-pooling layers to extract representative video features and obtain compact binary codes. We discriminatively train our model using a Siamese network [155] that maximizes the inter-class distance and minimizes the intra-class distance of the video feature pairs. In addition, we minimize the quantization loss between real-value codes and binary codes during training. We explore different spatial-temporal feature pooling architectures and compare them to single-frame CNN architectures as well as state-of-the-art image-based hashing methods. Experimental results on two video datasets show the effectiveness of our proposed approach.

6.2 Deep Video Hashing

In this section, we first present the basic idea of the learning-based video hashing model. We then describe our deep video hashing (DVH) method and its implementation details.

6.2.1 Learning-based Video Hashing

Let $\mathcal{X} = \{\mathbf{X}_i, \mathbf{y}_i\}_{i=1}^N$ be a collection of N videos where $\mathbf{X}_i = [\mathbf{x}_{i,1}, \mathbf{x}_{i,2}, \dots, \mathbf{x}_{i,f_i}] \in \mathbb{R}^{f_i \times h \times w \times 3}$ is the i th video with successive f_i frames, $\mathbf{y}_i$ is the label information, and $\mathbf{x}_{i,j} \in \mathbb{R}^{h \times w \times 3}$ is the j th image feature frame of video $\mathbf{X}_i$ with a height h and width w . The objective of learning-based video hashing is to learn K hashing functions to project each video into a single or multiple K -bit compact binary vectors as follows:

$$\mathcal{F}_{\mathbf{X}_i} : \mathbb{R}^{f_i \times h \times w \times 3} \rightarrow \{-1, 1\}^{K \times g_i} \quad (6.1)$$

where $g_i \in [1, f_i]^1$.

To obtain hashing functions, we learn a linear projection matrix, $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_K] \in \mathbb{R}^{d \times K}$ to map the video features into compact binary codes. To obtain a single binary vector for the i th video, $\mathbf{b}_i \in \{-1, 1\}^K$, we first extract a compact single feature representation for the i th video defined as $\tilde{\mathbf{x}}_i \in \mathbb{R}^d$ where d is the feature length, and then project it linearly as follows:

$$\mathbf{b}_i = \text{sgn}(\mathbf{W}^\top \tilde{\mathbf{x}}_i) \quad (6.2)$$

However, it is difficult to represent a whole video as a single binary code without losing significant amount of information. Hence, instead, a single video can be transformed to multiple binary vectors by treating each frame as an image feature through a linear or nonlinear embedding function, $f(\mathbf{x}_i) \in \mathbb{R}^d$, and performing image-based hashing. To obtain multiple binary vectors for the i th video, $\mathbf{B}_i \in \{-1, 1\}^{K \times f_i}$, we compute using the frame-wise feature representations as follows:

$$\mathbf{B}_i = \text{sgn}(\mathbf{W}^\top f(\mathbf{X}_i)) \quad (6.3)$$

¹During learning, the binary codes are set to $\{-1, 1\}$ to ensure proper centering, but they can then be set to $\{0, 1\}$ during retrieval.

Conventional learning-based video hashing methods make use of the aforementioned hand-crafted single representation features [100, 103] and/or use a single linear projection [99], which may not effectively capture the nonlinear relationship of video representations and cannot exploit temporal information present in videos.

6.2.2 Deep Video Hashing

Our work employs a deep learning model to learn several nonlinear projections to obtain compact binary codes, where both the discriminative and temporal information of videos are exploited in an end-to-end learning framework. By doing so, we are able to learn powerful video representations in a spatial-temporal level. Unlike previous video hashing methods which either learn hashing functions from a single video feature representation or frame-by-frame, we process a set of successive frames to obtain a single binary vector which leads to a fewer number ($g_i < f_i$) of binary codes to represent the i th video. By doing so, we are able to minimize the number of binary codes to represent each video but still extract significant information helpful for search.

As shown in Figure 6.1, given a fixed frame size p , we have a set of image frames $\mathbf{I}_i \in \mathbb{R}^{p \times h \times w \times 3}$, where $\mathbf{I}_i \subset \mathbf{X}_i$, that is passed through a series of convolution, pooling and fully connected layers. By letting $\phi(\cdot)$ be the output at the last fully connected layer where it contains K nodes, the binary code for the set of image frames of the i th video is computed as follows:

$$\mathbf{b}_i = \text{sgn}(\phi(\mathbf{I}_i)) \quad (6.4)$$

There are two important intuitions for our DVH model: (1) By performing several nonlinear transformations with a discriminative criterion, more robust visual representation can be obtained; (2) By performing the temporal pooling, we can implicitly exploit the relevant

Figure 6.2: Different DVH architectures with frame fusion. The vertical bars represent the raw image frames and each color represents a single video, 'C' represents the stacked convolution and pooling layers, 'FC' represents the fully-connected layers, 'P' represents the temporal pooling layers, and 'LM' represents the large-margin cost function, respectively.

frames and extract a balance of global and local information from video frames. By doing so, the noisy frames which may degrade the quality of the binary codes can be implicitly ignored. We discuss how to exploit both the temporal and discriminative information in our deep architecture below:

6.2.2.1 Temporal Information

Since videos typically represent motion-based features across time, it is necessary to consider temporal information to fully obtain video-level representation. In order to exploit temporal information in deep networks, we perform pooling operations across frames between the fully-connected layers. In our work, we perform fusion of temporal information through average pooling². We describe three deep networks with various feature pooling architectures:

Early Fusion: The early fusion architecture first passes image frames through the convolution and pooling layers and then fuses the information at the first fully connected layer, as shown in Figure 6.2(a).

Late Fusion: The late fusion architecture first passes image frames through the convolution and pooling layers up to the other fully-connected layers and then fuses the information at the last fully connected layer, as shown in Figure 6.2(b).

Slow Fusion: The slow fusion architecture is a balance of the early and late fusion. Image frames are passed through the convolution and pooling layers and then fused in a hierarchical manner such that smaller temporal windows are used as it approaches the top layer. In this work, a two-stage fusion strategy is implemented. Figure 6.2(c) details the architecture of this fusion strategy.

6.2.2.2 Discriminative and Binary Information

To learn the parameters in the deep network discriminatively, we employ the Siamese network [155] with a large-margin learning framework rather than the conventional contrastive divergence criterion [156]. Specifically, we present a new formulation which consists of

²We found that max pooling does not give very representative information compared to average pooling.

two new objective criteria for binary code learning. The first objective performs discriminative learning. Specifically, given two sets of image frames, $\mathbf{I}_i$ and $\mathbf{I}_j$, we minimize the intra-class variation and maximize the inter-class variation of the binary feature representation at the top layer of these two networks, simultaneously. Given their Hamming distance $d_h(\mathbf{b}_i, \mathbf{b}_j)$ at the top layer, where $\mathbf{b}_i = \text{sgn}(\phi(\mathbf{I}_i))$ and $\mathbf{b}_j = \text{sgn}(\phi(\mathbf{I}_j))$, we expect that $d_h(\cdot)$ is small if $\mathbf{I}_i$ and $\mathbf{I}_j$ are the same class, and large if they are from different classes, which is formulated as the following constraints:

$$d_h(\mathbf{b}_i, \mathbf{b}_j) \leq \tau_1, \quad \text{if } S_{i,j} = 1 \quad (6.5)$$

$$d_h(\mathbf{b}_i, \mathbf{b}_j) \geq \tau_2, \quad \text{if } S_{i,j} = -1 \quad (6.6)$$

where τ_1 and τ_2 are the small and large thresholds, respectively. By combining (6.5) and (6.6), we can simplify the constraints to:

$$S_{i,j}(\tau - d_h(\mathbf{b}_i, \mathbf{b}_j)) > 1 \quad (6.7)$$

where $\tau_1 = \tau - 1$ and $\tau_2 = \tau + 1$, and $S_{i,j} = 1$ means that the i th and j th image frame set are from the same class, and $S_{i,j} = -1$ indicates that they are from different classes. This leads to the following objective function:

$$J_1 = \zeta(1 - S_{i,j}(\tau - d_h(\mathbf{b}_i, \mathbf{b}_j))) \quad (6.8)$$

where $\zeta(z)$ is a generalized logistic loss function which is a smooth approximation of the hinge loss function: $z = \max(z, 0)$, and defined as follows:

$$\zeta(z) = \frac{1}{\rho} \log(1 + \exp(\rho z)) \quad (6.9)$$

where ρ is the sharpness parameter set to 10 and τ is the threshold parameter set to $K/4$.

The second objective is to ensure efficient binary codes by minimizing the quantization

loss [41] between real-valued codes and binary codes as follows:

$$J_2 = \|\phi(\mathbf{I}_i) - \mathbf{b}_i\|_2^2 + \|\phi(\mathbf{I}_j) - \mathbf{b}_j\|_2^2 \quad (6.10)$$

Hence, the final objective function for our DVH is formulated as:

$$\begin{aligned} \min_{\mathbf{b}_i, \mathbf{b}_j} J &= J_1 + \lambda J_2 \\ &= \zeta(1 - S_{i,j}(\tau - d_h(\mathbf{b}_i, \mathbf{b}_j))) + \lambda(\|\phi(\mathbf{I}_i) - \mathbf{b}_i\|_2^2 + \|\phi(\mathbf{I}_j) - \mathbf{b}_j\|_2^2) \end{aligned} \quad (6.11)$$

where J_1 exploits the discriminative information, J_2 minimizes the quantization loss, and λ is a constant parameter which balances the two criterions.

We use the standard mini-batch gradient descent and back-propagation to solve the optimization problem. We first relax the binary constraints in the first objective and use the signed magnitude real-valued codes. Given $\mathbf{h}_i = \phi(\mathbf{I}_i)$ and $\mathbf{h}_j = \phi(\mathbf{I}_j)$ are the real-valued codes from the top layer, back-propagation is implemented first by taking the derivative of J with respect to $\mathbf{h}_i$ and $\mathbf{h}_j$:

$$\frac{\partial J}{\partial \mathbf{h}_i} = \zeta'(z) S_{i,j}(\mathbf{h}_i - \mathbf{h}_j) + \lambda(\mathbf{h}_i - \mathbf{b}_i) \quad (6.12)$$

$$\frac{\partial J}{\partial \mathbf{h}_j} = \zeta'(z) S_{i,j}(\mathbf{h}_j - \mathbf{h}_i) + \lambda(\mathbf{h}_j - \mathbf{b}_j) \quad (6.13)$$

These gradients are then back-propagated to update the weights at the earlier layers.

6.2.2.3 Extracting Binary Codes from One Video

We extract multiple binary codes for each video using our DVH through a set of consecutive image frames in the video with a specific stride. Figure 6.3 shows the hashing procedure for one video using our model. The over-all hamming distance, D_H , for a pair of videos are then obtained by getting the average of the hamming distances, d_h , for each pair of binary

Figure 6.3: Extracting binary codes for a single video from DVH. We extract the binary code in a pre-defined number of consecutive frames, p , and shift to the next set based on a fixed stride value. In this figure, given 8 frames, the DVH fuses $p = 4$ frames with a stride of 2, resulting to 3 final compact binary codes.

codes as follows:

$$D_H(\mathbf{X}_i, \mathbf{X}_j) = \frac{1}{g_i g_j} \sum_{u=1}^{g_i} \sum_{v=1}^{g_j} d_h(\mathbf{b}_{u,i}, \mathbf{b}_{v,j}) \quad (6.14)$$

where g_i and g_j is the number of frame sets for videos $\mathbf{X}_i$ and $\mathbf{X}_j$. $\mathbf{b}_{u,i}$ and $\mathbf{b}_{v,j}$ would be the binary code extracted from u th and v th frame set of video $\mathbf{X}_i$ and $\mathbf{X}_j$, respectively. By doing so, we are able to compare the similarity of videos with less computation than using frame-to-frame comparisons, which are more representative than using single features for each video.

6.2.3 Implementation Details

Our deep video model composed firstly of stacked convolution and pooling layers, with parameters obtained from pre-trained models. It is then connected to a series of fully connected layers such that the top-most layer outputs K -dimensional features. The hidden fully connected layers used rectified linear unit (ReLU) as the activation function, while the top-most layer has a hyperbolic tangent activation to ensure centered feature and have balanced $\{-1, 1\}$ values. The parameters in the fully connected layers are initialized using

Table 6.1: DVH architectural implementations in the experiments.

Early Fusion	Late Fusion	Slow Fusion
fc7 - 4096	fc7 - 4096	fc7 - 4096
pool at p frames	-	pool at p_1 frames
fc8 - 500	fc8 - 500	fc8 - 500
-	pool at p frames	pool at p_2 frames
fc9 - 200	fc9 - 200	fc9 - 200
K	K	K

the Xavier initialization [141]³. To be consistent, all models have a fully-connected layers of dimensions $[4096 \rightarrow 500 \rightarrow 200 \rightarrow K]$. To avoid over-fitting, we trained only the parameters for the fully connected layers. Our deep architecture and experiments were implemented under the MatConvNet⁴ framework. The learning rate, momentum, and weight decay were set to 0.002, 0.9, and 0.0001, respectively. Table 6.1 summarizes the architectural design of different DVH methods after the stacked convolution and pooling layers.

At the training stage, we iteratively passed through all the training videos where we randomly chose video pairs. For each video pair, we randomly chose a set of p successive frames and then packed them into batches to pass into the network. We ensure that the positive and negative pairs for each batch are in an approximate 1:2 ratio. The training procedure converged when the loss does not change within a certain threshold for an epoch. For all experiments, ρ was empirically set to 10 to obtain a smooth approximation for the hinge loss. We experimented with different values and found that the results within this range appear to be particularly insensitive to ρ .

³We initialize the biases to be 0 and the weights at each layer as $\mathbf{W} = U \left[-\sqrt{\frac{6}{n_{in}+n_{out}}}, \sqrt{\frac{6}{n_{in}+n_{out}}} \right]$ where $\mathbf{W} \in \mathbb{R}^{n_{in} \times n_{out}}$.

⁴<http://www.vlfeat.org/matconvnet/>

6.3 Experiments

To evaluate the effectiveness of our proposed DVH method for video search, we conducted experiments on two video datasets namely the CCV and JHMDB datasets. Figures 6.4 and 6.5 show sample video frames from these two datasets. The details of the experiments and the results are described in the following sections.

6.3.1 Datasets and Experimental Set-up

6.3.1.1 Datasets

Columbia Consumer Video (CCV) dataset [157]⁵: consists of 9317 videos with an average duration of 80 seconds extracted from YouTube. The videos were categorized to 20 different categories such as *basketball*, *wedding* and *music performance*. Similar to [99], we sampled frames every 2 seconds and ensured that each video had a minimum of 30 frames. Since most of the categories in this dataset are events, the videos contain large variations between frames, making the task very challenging. In our experiments, we randomly selected 20 videos per category for training, 25 videos per category as the query data, and 100 videos per category as the gallery data. This resulted in 400, 500, and 2000 videos for the training, query, and gallery sets, respectively.

For our deep model, we used the pre-trained VGG-net [158]⁶ as our stacked convolution and pooling layers. We used a batch size of 200 and a frame size of $p = 10$. For our Slow Fusion architecture, the first pooling layer fuses the data of $p_1 = 5$ frames with a stride of 2, the second pooling layer fuses the data of the final $p_2 = 3$ frames. During testing, we obtained the binary codes for each video at a frame stride of 5.

⁵<http://www.ee.columbia.edu/ln/dvmm/CCV/>

⁶<http://www.vlfeat.org/matconvnet/models/imagenet-vgg-f.mat>

Figure 6.4: Two sample videos for the Columbia Consumer Video (CCV) database. The first video belongs to the *music performance* category, while the second video belongs to the *baseball* category.

Figure 6.5: Two sample videos for the Joint-annotated HMDB (JHMDB) database. The first video represents a *shooting* action video, while the second video represents a *catching* action video.

Joint-annotated HMDB (JHMDB) dataset [159]⁷: consists of 928 action videos having 36 to 55 frames per video for human motion recognition. The action videos are categorized into 21 human actions such as *brushing hair*, *clapping*, and *climbing*. Although action recognition makes use of flow and RGB information [160], we only used the full body optical flow representation for simplicity. In our experiments, we randomly selected 10 videos per category as training samples, 10 videos per category as query samples, and 20 videos per category as gallery samples. This resulted in 210, 210, and 420 videos for the training, query, and gallery sets, respectively.

For our deep model, we used the CNN motion network by [161] as our pre-trained stacked convolution and pooling layers. We used a batch size of 50 and a frame size of $p = 10$. For our Slow Fusion architecture, the first pooling layer fuses the data of $p_1 = 5$

⁷<http://jhmdb.is.tue.mpg.de/>

Figure 6.6: Deep baseline architectures based on frame-by-frame training for video hashing. The first baseline is similar to image hashing, where each video frame is treated as a single image. The second baseline adds a temporal criterion during training such that given two frames that are temporally close should have a similar compact feature as much as possible.

frames with a stride of 2, the second pooling layer fuses the data of the final $p_2 = 3$ frames. During testing, we obtained the binary codes at a stride of 2.

6.3.1.2 Evaluation Metrics

To measure the performance of our DVH, we used the Hamming ranking and Hamming look-up as evaluation metrics to compare the performance of different methods. For Hamming ranking, the mean Average Precision (mAP) and Precision@ N are evaluated. The mAP is the mean of the average precision of the top retrieved samples across all queries. The Precision@ N is the percentage of true labels among the top N retrieved samples. For Hamming look-up, the precision over all the samples that is within a hamming radius of $r = 2$ is evaluated. At $K = 64$, Hamming look-up precision is not evaluated because it is

an impractical evaluation metric for large bit lengths.

6.3.2 Experimental Results

6.3.2.1 Comparison with Different Deep Baselines

We first compared our DVH with three baseline deep architectures which do not use temporal fusion in the fully-connected layers. The baseline methods are described below:

Single-Frame: This model considers each frame of the video as a single image, $\mathbf{x}_i$, with its own label information. Similar to DVH, we used a large-margin criterion for the Siamese network to learn the parameters. Differently, we only used single image frames as the input and without temporal fusion. The cost function is defined as:

$$J = \zeta(1 - S_{i,j}(\tau - d(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)))) \quad (6.15)$$

Single-Frame + Temporal: This model considers the temporal information during training. Particularly, aside from the large-margin criterion, we ensure that frames which are close (temporal-wise) to each other have similar binary codes as much as possible. The new cost function is defined as:

$$J = \zeta(1 - S_{i_1,j}(\tau - d(\phi(\mathbf{x}_{i_1}), \phi(\mathbf{x}_j)))) + \nu \|\phi(\mathbf{x}_{i_1}) - \phi(\mathbf{x}_{i_2})\|_2^2 \quad (6.16)$$

where ν is the balancing term, and i_1 and i_2 are two randomly selected image frames from the same video which are apart by a minimum of five frames. In the experiments, we used $\nu = 0.1$. Figure 6.6 shows the architectures of the first two baseline methods.

Video-Level Feature: In this model, we pooled all frame-level features from the single-frame deep model to compute video-level features to evaluate the large-margin criterion. By doing so, we obtained a single global binary vector for each video.

Table 6.2: Performance results comparison of our DVH with the baseline deep architecture on the CCV dataset.

Method	Hamming ranking (mAP, %)			precision (%) @ N = 100			prec. (%) @ r=2	
	16	32	64	16	32	64	16	32
Single	32.62	34.23	35.02	37.37	38.86	38.50	23.38	5.63
Single+Temporal	33.60	35.60	37.27	38.05	39.74	40.93	30.76	16.44
Video-Level	29.45	30.79	29.19	34.44	35.98	34.05	22.48	11.11
Early Fusion	37.18	40.86	41.54	40.11	41.89	42.41	36.61	22.80
Late Fusion	38.54	41.08	41.51	40.29	42.08	42.23	37.32	23.10
Slow Fusion	38.27	40.80	41.41	39.95	41.88	42.34	36.55	23.06

Table 6.3: Performance results comparison of our DVH with the baseline deep architecture on the JHMDB dataset.

Method	Hamming ranking (mAP, %)			precision (%) @ N = 10			prec. (%) @ r=2	
	16	32	64	16	32	64	16	32
Single	32.73	33.89	31.74	42.67	43.81	44.05	6.19	0
Single+Temporal	33.19	34.85	35.58	43.81	45.33	45.19	9.05	0
Video-Level	30.07	30.95	32.53	39.10	41.67	41.86	6.58	0
Early Fusion	35.19	37.43	37.95	46.48	47.62	48.19	31.31	12.46
Late Fusion	34.93	36.78	37.53	45.10	48.05	48.14	29.67	10.10
Slow Fusion	34.86	36.59	36.63	44.52	47.90	48.24	25.25	8.67

Tables 6.2 and 6.3 show the performance of different methods on the CCV and JHMDB datasets, respectively. As can be seen, our DVH architectures outperform the other baseline architectures. It is interesting to see that the second baseline (Single+Temporal) outperforms the first baseline (Single), which shows that temporal information is important. The video-level model also yields competitive performance but often achieves worse performance than the single-frame deep model. This is because it is difficult to represent all the relevant information from a video to only a single binary code. The late fusion and early fusion DVH architectures obtain the best performance on the CCV and JHMDB datasets, respectively. For CCV, a late fusion would mean that it prefers to exploit high-level global information, while for JHMDB an Early Fusion shows that it prefers to exploit local motion information [116]. This is reasonable because JHMDB focuses on action information while CCV is more on events which are generally holistic.

Figure 6.7: Retrieval time on the (a) CCV and (b) JHMDB datasets.

Table 6.4: Performance results comparison of our DVH and different learning-based hashing methods on the CCV dataset.

Method	Hamming ranking (mAP, %)			precision (%) @ N = 100			prec. (%) @ r=2	
	16	32	64	16	32	64	16	32
PCAH [162]	20.83	21.45	19.37	25.80	26.50	25.51	3.03	0
PCA-ITQ [41]	22.49	24.13	24.42	27.71	28.99	29.61	13.43	0
AGH [62]	14.91	15.22	11.24	20.52	23.37	20.16	13.43	1.58
KSH [42]	32.43	34.34	35.40	36.27	38.33	38.75	18.27	7.64
CCA-ITQ [41]	36.58	38.18	38.32	39.13	40.41	40.51	16.15	7.17
FastHash [44]	34.72	38.37	38.47	38.83	40.85	41.37	12.73	5.36
DVH	38.54	41.08	41.51	40.29	42.08	42.23	37.32	23.10

We also examined the retrieval time between our model and the single-frame model, which is shown in Figure 6.7. As can be seen, a longer retrieval time is needed since the baseline architecture transforms each image frame into a binary code. Differently, our DVH method performs temporal fusion so fewer binary code comparisons are required resulting in a faster retrieval time for the whole query-gallery set. This is most obvious on the CCV dataset since more frames are present in a single video and a larger number of gallery and query videos are used.

Table 6.5: Performance results comparison of our DVH and different learning-based hashing methods on the JHDMB dataset.

Method	Hamming ranking (mAP, %)			precision (%) @ N = 10			prec. (%) @ r=2	
	16	32	64	16	32	64	16	32
PCAH [162]	16.89	17.64	18.30	27.05	31.31	31.81	0	0
PCA-ITQ [41]	13.80	14.16	14.44	22.52	25.19	27.05	0.58	0
AGH [62]	13.74	14.30	16.90	23.38	26.86	27.57	0.12	0
KSH [42]	27.50	28.32	33.51	39.14	39.05	42.38	0	0
CCA-ITQ [41]	27.20	31.96	31.44	43.48	47.56	47.43	1.51	0
FastHash [44]	31.19	33.72	36.63	42.29	44.33	46.52	0	0
DVH	35.19	37.43	37.95	46.48	47.62	48.42	31.31	12.46

6.3.2.2 Comparison with State-of-the-Art Learning-based Hashing methods

We also compared our DVH method with several popular hashing methods including PCA Hashing [162], PCA-ITQ [41], Anchor Graph Hashing (AGH) [62], Kernel Supervised Hashing (KSH) [42], CCA-ITQ [41], and FastHash [44]. Specifically, PCAH, PCA-ITQ, and AGH are unsupervised hashing methods, and KSH, CCA-ITQ, and FastHash exploit the label information of samples to learn discriminative hash codes. The standard implementations of all methods are from the original authors and the default parameters were set based on their respective papers. For consistency, the experiments were carried out with the same selected training, gallery and query sets. For the different hashing methods being compared, we extracted 4096-dimensional CNN features for each image frame and encoded its respective binary code. We did this for all image frames for each video. The Hamming distance of two videos would then be the average of all hamming distances between the binary codes of the first video and the binary codes of the second video.

Tables 6.4 and 6.5 show the performance of different hashing methods on the CCV and JHMDB experiments, respectively. Here, we present the DVH results which yielded the best performance among the different architectures (CCV - Late, JHMDB - Early). As can be seen, our method consistently outperforms the conventional hashing methods.

Figure 6.8: Precision-Recall (PR) curves on the CCV dataset versus varying code lengths.

Figure 6.9: Precision-N curves on the CCV dataset versus varying code lengths.

This shows that representing the video to binary codes through hierarchical nonlinear mapping gives strong representation. Most surprising is the Hamming Look-up results which show significant improvement across varying bit lengths. Figures 6.8-6.11 show the recall-precision curves and Precision@ N curves on the CCV and JHMDB datasets. We see that

Figure 6.10: Precision-Recall (PR) curves on the JHMDB dataset versus varying code lengths.

Figure 6.11: Precision-N curves on the JHMDB dataset versus varying code lengths.

 Figure 6.12: mAP performance of our DVH method at varying λ for the 64-bit experiment on the (a) CCV and (b) JHMDB datasets, respectively.

our method outperforms the compared methods in most scenarios.

6.3.2.3 Comparison with Different Video Hashing Methods

We compared our method with two video hashing methods as shown in Table 6.6. For Video Hashing with both Discriminative commonality and Temporal consistency (VHDT) [99], the mAP results were obtained from the original paper. However, their method used SIFT BoW features so it is difficult to directly compare. To approximate, we have applied ITQ-SIFT and found that it is comparable with VHDT. Our DVH is much better than ITQ using CNN feature input (ITQ-CNN) which can tell us that our method would be better than

Table 6.6: Hamming ranking (mAP, %) results in comparison with other video hashing algorithms on the CCV dataset.

Method	16 bits	32 bits	64 bits
DVH-CNN	38.54	41.08	41.51
ITQ-CNN	22.49	24.13	24.42
ITQ-SIFT	12.33	14.63	14.52
VHDT-SIFT [99]	11.40	13.00	15.90
CVC-CNN [103]	27.53	32.16	36.14

Table 6.7: Hamming ranking (mAP, %) results of our DVH with different values of p and $stride$ on the CCV dataset.

Method	16 bits	32 bits	64 bits
$p = 2, stride = 2$	37.81	39.48	40.46
$p = 5, stride = 5$	37.81	40.78	41.09
$p = 10, stride = 5$	38.54	41.08	41.51
$p = 20, stride = 5$	37.25	40.11	41.31

VHDT-CNN. This may be because VHDT performs linear transformations which may not really capture the nonlinearity of data in videos.

For the Compact Video Coding (CVC) method [103], we used the publicly released code, tuned the parameters to obtain the best possible result and used CNN features to construct the covariance feature for each video. As can be seen, our DVH outperforms CVC at all bit lengths. This is because CVC converted the whole video into a single feature code which may lead to loss of important information. Differently, our DVH method considered the temporal and discriminative information of each video.

Table 6.8: Hamming ranking (mAP, %) results of our DVH in different values of p and $stride$ on the JHMDB dataset.

Method	16 bits	32 bits	64 bits
$p = 2, stride = 2$	32.48	32.58	33.55
$p = 5, stride = 5$	33.21	35.32	35.30
$p = 10, stride = 5$	35.19	37.46	37.95
$p = 20, stride = 5$	37.97	35.82	36.43

6.3.3 Further Analysis

6.3.3.1 Parameter Sensitivity for the Quantization Loss

We also analyzed the varying values of λ during training to see the contribution of the two criteria in the overall performance of our DVH method. Figure 6.12 shows the mAP performance of our DVH method at $\lambda = [0, 0.01, 0.005, 0.001, 0.0005, 0.0001]$ for the CCV and JHMDB datasets. As expected, we can see that the discriminative information makes most of the contribution since even at $\lambda = 0$, the performance is very competitive. Nevertheless, minimizing the quantization loss still provides an improvement in the overall performance. However, it is important to see that the quantization loss criterion should not overpower the discriminative criterion. In our experiments, the optimal value for λ is at a range of $[0.005, 0.01]$.

6.3.3.2 Analysis on varying frame size

We also conducted experiments on varying values of the frame size p and the stride value. p is the number of frames used as input in the deep network to obtain a single binary code. Stride is the number of non-overlapped frames when selecting the next image set fed into DVH for each video (see Figure 6.3). We used the best fusion algorithm for each dataset (CCV-Late, JHMDB-Early). As seen in Tables 6.7 and 6.8,, our method shows a decline in mAP for both datasets at $p < 10$ probably because consecutive frames are very much similar which does not allow useful temporal information to be exploited. Similarly, a much higher p may also reduce the performance because it extracts more global information. Hence, we can conclude that a balance of exploiting temporal and global information is more useful to obtain promising results. Hence, it is encouraged to determine the optimal value of p by examining the performance on a validation set for every dataset used.

6.3.4 Discussion

The above experimental results suggest the following three key observations:

- (i) Our DVH method achieves very competitive performance compared to other deep baseline architectures. This shows that performing temporal fusion during training contributes well to the overall performance. In addition, retrieval time is also reduced because of temporal fusion.
- (ii) Our DVH outperforms state-of-the-art image-based hashing methods. This shows that the binary codes obtained from our hashing method are strong representations due to the discriminative training we employed. Furthermore, our DVH also outperforms other video hashing methods by a large margin.
- (iii) The large-margin criterion yields the largest contribution for our DVH method. However, the binary quantization term also provides improvements in the overall performance. For the parameter p , we see that the best performance can be obtained when the parameter of p is set to 10 because it is a good balance of extracting global and local video features.

6.4 Chapter Summary

In this chapter, we have proposed a deep video hashing approach with various frame pooling architectures to learn binary codes for each video in a deep framework such that both temporal and discriminative information is well exploited. Experimental results on two video databases clearly demonstrate that our method achieves better performance with the state-of-the-art hashing methods. The work presented in this chapter was published in [34].

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this thesis, we have presented several compact feature learning methods for multimedia search. The main contribution of this thesis is we proposed and analyzed different new compact feature learning techniques to obtain discriminative query and gallery compact feature representations from raw data that are suitable for accurate retrieval systems. Specifically, our proposed methods present different machine learning tools that optimize different objectives suitable for real-valued (Chapter 3) and binary code (Chapters 4, 5, and 6) compact feature learning. All proposed methods leveraged deep neural networks to learn representative features through deep metric learning (Chapters 3 and 6) and deep generative models (Chapter 5). These were applied in respective multimedia search problems such as visual search (Chapters 5), cross-modal search (Chapters 3, 4 and 5) and video search (Chapter 6). To demonstrate the effectiveness, we conducted extensive experiments to evaluate and compare the performance of our proposed methods with existing conventional and state-of-the-art methods. The specifics of the contributions are summarized below.

In Chapter 3 we proposed a deep coupled metric learning (DCML) method for cross-modal matching to reduce the modality gap between different modalities by enhancing the representation of the learned compact features using deep neural networks. Our DCML

method used a couple of deep neural networks, one for each modality, to perform hierarchical nonlinear transformations and is trained using a unified criterion to exploit discriminative information and explicitly reduce the modality gap. Our model outperformed several state-of-the-art cross-modal methods on three image-text/tag retrieval experiments on benchmark datasets (Wiki, PASCAL VOC 2007, and NUS-WIDE). It also achieved competitive performance with heterogeneous face recognition experiments on the CASIA VIS-NIR dataset. We also showed that both the discriminative and correlation losses for our coupled network contributed to the overall performance.

In Chapter 4 we proposed a cross-modal discrete hashing (CMDH) method for cross-modal retrieval to address the issue of approximation loss caused by relaxing the binary constraint. Our CMDH performs discrete optimization while simultaneously learning a unified binary code and discriminative hashing functions. We perform our CMDH framework in both shallow and deep manner. In particular, our shallow-CMDH has hashing functions that perform linear and kernel mapping; and our deep-CMDH uses deep neural networks to perform hierarchical nonlinear mapping. Experimental results on benchmark cross-modal datasets (Wiki, MIRFLICKR, IAPRTC12, and NUSWIDE) showed the effectiveness of our proposed approach. Our kernel-CMDH outperformed the popular shallow hashing methods and the state-of-the-art SePH [82]. We then extended our shallow framework to unsupervised and multi-modal forms to handle training without label data and with multi-modal data, respectively. Finally, our deep-CMDH showed promising performance compared to other deep hashing models. This general cross-modal discrete hashing framework can be extended by designing other forms of hashing models (i.e. boosting, rotation-based).

In Chapter 5 we proposed a deep variational and structural hashing (DVStH) and cross-modal deep variational and structural hashing (CM-DVStH) for single-modal and cross-

modal retrieval, respectively. We enhanced the compact binary code learning of our deep hashing framework by using point-wise classification loss instead of a metric-based loss using pair or triplet of training samples. Additionally, our hashing network includes a variational layer to learn more general features and a struct layer to exploit more information during training. For our CM-DVStH, we designed a deep fusion network to learn unified binary codes which are used to learn the modality-specific hashing networks such that the latent feature is close as possible with the binary code inferred from the fusion network. Experimental results on the CIFAR10, ImageNet, NUS-WIDE for image retrieval and MIR-Flickr, IAPRTC12 NUS-WIDE for cross-modality retrieval were performed to evaluate our proposed methods. Our DVStH showed competitive performance compared to other state-of-the-art shallow and deep hashing methods for single-domain, cross-domain, and multi-label image retrieval scenarios. Our CM-DVStH also achieved competitive performance compared to other state-of-the-art shallow and deep cross-modal hashing methods for cross-modal retrieval. Our work suggests that deep generative models can be further explored for hash code learning to improve the general feature learning. Additionally, our work suggests a different way of obtaining binary codes in a deep network instead of the sigmoid-thresholding procedure on a hash layer done by many deep hashing literatures.

Lastly, in Chapter 6 we proposed a deep video hashing (DVH) method for video retrieval to explore efficient video hashing using deep neural networks. Specifically, we explore different architectures to perform temporal pooling and train our network using large-margin loss and quantization loss. Experimental results on the CVC and JHMDB video dataset demonstrate the effectiveness of our approach. Specifically, our DVH method showed the importance of temporal fusion in hash code learning for videos by comparing it with baseline deep architectures. It also outperformed state-of-the-art image-based hashing methods and current video hashing approaches. Our work encourages to further explore a

deep learning approach for video hashing to utilize temporal and discriminative information for binary code learning on videos.

7.2 Future Work

While the works proposed in this thesis have shown competitive performance compared to state-of-the-art compact feature learning methods on several multimedia search problems, there are still ways to improve the performance of our methods and extend them to other multimedia search problems. Hence, we present some feasible ideas that can be future research direction:

- (i) In Chapter 3 our deep model learns compact features for cross-modal matching. To extend this to learn compact binary codes would be interesting so that it would be suitable for hashing-based ANN search. Additionally, our method uses pre-computed hand-crafted and CNN features as input. To improve the performance of our method, training an end-to-end deep learning for cross-modal retrieval with raw data input is desired.
- (ii) In Chapter 3, our method used hand-crafted image features instead of raw images as input. However, using raw images may require more layers to generate abstract features as compared to its counterpart modality. Hence a future direction would be to investigate the inclusion of distributed intermediate layers which does layer-level couple minimization similar to H_2 or Equation 3.9.
- (iii) In Chapter 4 our model uses a discrete optimization technique for cross-modal hashing in a shallow and deep framework. In our deep CMDH, we solve the classification step using a linear hinge loss function. Using a nonlinear classification loss is desirable to obtain more accurate binary code to classification mapping.

- (iv) In Chapter 5 our modality-specific networks learn the inferred binary codes independently. Transfer learning techniques for deep networks [163, 164] can be explored to transfer the knowledge from the fusion network to the modality-specific networks as well as between the two modality-specific networks. By doing so, it may further minimize the modality gap between modality-specific networks in the mid-level when learning network parameters.
- (v) In Chapter 6 our deep video hashing model uses a discriminative deep network with frame fusion architecture. This uses straightforward convolution, pooling and fully-connected layers as the base network. To further exploit temporal information in videos and improve the performance in video search, it is possible to incorporate more complex time-series networks such as recurrent neural networks (RNN) [165], long short term memory (LSTM) networks [166], and 3D-CNNs [167].
- (vi) In Chapters 3, 4 and 5 the text networks use a series of fully-connected layers to obtain representative text features. A more complex network such as LSTMs [168] would help learn more representative text features for cross-modal retrieval.
- (vii) All our proposed methods employ a deep learning framework. How to improve the performance of our deep frameworks by exploring new activation functions (e.g. SeLU [169]), network architectures (e.g. SqueezeNets [170]), and training strategies are desirable.
- (viii) All our methods rely on supervised information to provide representative compact features. In many real-world applications, it is difficult to obtain enough labelled data for training. How to perform unsupervised compact feature learning by including deep generative models would be interesting.

- (ix) In many real-world applications, unseen labels commonly exist. But these unseen labels may have some latent relationships available from seen labels. We can exploit this relationship by adding an attribute-based criterion to our models. This task is called zero-shot learning. Zero-shot learning aims to learn a compact feature learning model which exploits supervised information from seen labels but are general enough to be effective in search for both seen and unseen categories. Currently, zero-shot hashing [171, 172] has not been extensively explored but is important for practical applications.
- (x) All our hashing techniques would be enhanced through the attention mechanism [173]. Particularly on images, we could first feed the images on an attention network to select the salient region (similar to [174, 175]). This would be a pre-processing step to obtain more quality information before feeding it to our hash function training or inference.
- (xi) Currently, Visual Questioning and Answering (VQA) [176] is a hot topic in computer vision and natural language processing (NLP) research. It requires to build a system (most likely a neural network) that could answer natural language questions about any given image. This is particularly challenging as it requires the understanding of several cues from textual data and relating them to the image being examined. Our work on compact feature learning and hashing can be explored in this field if the natural language questions are asked to "find or select" a set of images given textual descriptions at a large-scale as quickly as possible. From the literature on VQA, there is currently no public dataset that provides such task but may be worth exploring. Moreover, there seems to be limited interaction between hashing and VQA. Hence it is interesting to incorporate such methods for VQA to improve its role in scalable applications.

- (xii) While 3D CNN may be computationally expensive to explore for deep video hashing, there are already works improving the speed in using 3D CNNs [177, 178]. If these technologies are mature and easily available for deep frameworks, we can use these 3D CNN-based network backbones for exploiting temporal information in videos for hashing.
- (xiii) Our compact feature learning methods were experimented using only visual and text features. However, we encourage extending these methods for other data types like audio and point cloud data.

We hope that techniques developed in this thesis can motivate more work in this direction and contribute a step toward achieving accurate and efficient multimedia search.

Author's Publications

- [1] Venice Erin Liong, Jiwen Lu, Yap-Peng Tan and Jie Zhou, “Deep Variational and Structural Hashing”, *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 2018, 10.1109/TPAMI.2018.2882816.
- [2] Venice Erin Liong, Jiwen Lu, Yap-Peng Tan, “Cross-Modal Discrete Hashing”, *Pattern Recognition*, 79, pp.114-129, 2018.
- [3] Venice Erin Liong, Jiwen Lu, Yap-Peng Tan, Jie Zhou, “Cross-Modal Deep Variational Hashing”, *International Conference on Computer Vision (ICCV)*, pp.4077–4085, 2017.
- [4] Venice Erin Liong, Jiwen Lu, Yap-Peng Tan, “Learning a Cross-Modal Hashing Network for Multimedia Search”, *IEEE International Conference in Image Processing (ICIP)*, pp.3700-3704, 2017.
- [5] Venice Erin Liong, Jiwen Lu, Yap-Peng Tan, Jie Zhou, “Deep Coupled Metric Learning for Cross-Modal Matching”, *IEEE Transactions on Multimedia (T-MM)*, 19(6), pp.1234-1244, 2017.
- [6] Venice Erin Liong, Jiwen Lu, Yap-Peng Tan, Jie Zhou, “Deep Video Hashing”, *IEEE Transactions on Multimedia (T-MM)* , 19(6), pp.1209-1219, 2017.
- [7] Jiwen Lu, Venice Erin Liong, Xiuzhuang Zhou, Jie Zhou, “Simultaneous local binary feature learning and encoding for homogeneous and heterogeneous face recognition”, *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 2017, 10.1109/TPAMI.2017.2737538.
- [8] Jiwen Lu, Venice Erin Liong, Jie Zhou, “Deep Hashing for Scalable Image Search”, *IEEE Transactions on Image Processing (T-IP)* , 26(5), pp.2352-2367, 2017.
- [9] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, Jie Zhou, “Deep Hashing for Compact Binary Codes Learning”, *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp.2475 - 2483.

- [10] Venice Erin Liong, Yongxin Ge, Jiwen Lu, “Discriminative Regularized Metric Learning for Person Re-Identification”, *International Conference on Biometrics (ICB)*, 2015
- [11] Venice Erin Liong, Jiwen Lu, Yongxin Ge, “Regularized local metric learning for person re-identification”, *Pattern Recognition Letters*, 68, pp.288-296, 2015.
- [12] Jiwen Lu, Venice Erin Liong, Jie Zhou, “Simultaneous Local Binary Feature Learning and Encoding for Face Recognition”, *IEEE International Conference on Computer Vision (ICCV)*, pp.3721-3729, 2015.
- [13] Jiwen Lu, Venice Erin Liong, Xiuzhuang Zhou, Jie Zhou, “Learning Compact Binary Face Descriptor for Face Recognition”, *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 37(10), pp.2041-2056, 2015.
- [14] Jiwen Lu, Venice Erin Liong, Jie Zhou, “Cost-Sensitive Local Binary Feature Learning for Facial Age Estimation”, *IEEE Transactions on Image Processing (T-IP)*, 24(12), pp.5356-5368, 2015.
- [15] Jiwen Lu, Venice Erin Liong, Gang Wang, Pierre Moulin, “Joint Feature Learning for Face Recognition”, *IEEE Transactions on Information Forensics and Security (T-IFS)*, 10(7), pp.1371-1383, 2015.

Bibliography

- [1] 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*, vol. 3, no. 3, pp. 209–226, 1977.
- [2] C. Silpa-Anan and R. Hartley, “Optimised kd-trees for fast image descriptor matching,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.
- [3] 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.
- [4] 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, 2011.
- [5] M. Norouzi and D. J. Fleet, “Cartesian k-means,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3017–3024.
- [6] T. Ge, K. He, Q. Ke, and J. Sun, “Optimized product quantization,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 4, pp. 744–755, 2014.
- [7] 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.
- [8] F. Shen, C. Shen, W. Liu, and H. Shen, “Supervised discrete hashing,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 37–45.
- [9] J. Wang, H. T. Shen, J. Song, and J. Ji, “Hashing for similarity search: A survey,” *arXiv preprint arXiv:1408.2927*, 2014.

- [10] M. Norouzi, A. Punjani, and D. J. Fleet, “Fast search in hamming space with multi-index hashing,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3108–3115.
- [11] D. R. Hardoon and J. Shawe-Taylor, “KCCA for different level precision in content-based image retrieval,” in *International Workshop on Content-Based Multimedia Indexing*, 2003.
- [12] X. Huang, Z. Lei, M. Fan, X. Wang, and S. Z. Li, “Regularized discriminative spectral regression method for heterogeneous face matching,” *IEEE Transactions on Image Processing*, vol. 22, no. 1, pp. 353–362, 2013.
- [13] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, “Multimodal deep learning,” in *International Conference on Machine Learning*, 2011, pp. 689–696.
- [14] F. Feng, X. Wang, and R. Li, “Cross-modal retrieval with correspondence autoencoder,” in *ACM International Conference on Multimedia*, 2014, pp. 7–16.
- [15] C. Wang, H. Yang, and C. Meinel, “A deep semantic framework for multimodal representation learning,” *Multimedia Tools and Applications*, vol. 75, no. 15, pp. 9255–9276, 2016.
- [16] W. Liu, C. Mu, S. Kumar, and S.-F. Chang, “Discrete graph hashing,” in *Advances in Neural Information Processing Systems*, 2014, pp. 3419–3427.
- [17] M. M. Bronstein, A. M. Bronstein, F. Michel, and N. Paragios, “Data fusion through cross-modality metric learning using similarity-sensitive hashing,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 3594–3601.
- [18] M. Rastegari, J. Choi, S. Fakhraei, D. Hal, and L. Davis, “Predictable dual-view hashing,” in *International Conference on Machine Learning*, 2013, pp. 1328–1336.
- [19] S. Kumar and R. Udupa, “Learning hash functions for cross-view similarity search,” in *International Joint Conference on Artificial Intelligence*, vol. 22, no. 1, 2011, p. 1360.
- [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*, 2016, pp. 1445–1454.
- [21] Y. Cao, M. Long, J. Wang, and P. S. Yu, “Correlation hashing network for efficient cross-modal retrieval,” in *British Machine Vision Conference*, 2017, pp. 1–12.
- [22] Q.-Y. Jiang and W.-J. Li, “Deep cross-modal hashing,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3232–3240.

- [23] 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.
- [24] 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.
- [25] 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.
- [26] Y. Weiss, A. Torralba, and R. Fergus, “Spectral hashing,” in *Advances in Neural Information Processing Systems*, 2009, pp. 1753–1760.
- [27] A. Gionis, P. Indyk, R. Motwani *et al.*, “Similarity search in high dimensions via hashing,” in *International Conference on Very Large Databases*, vol. 99, no. 6, 1999, pp. 518–529.
- [28] R. Ji, L.-Y. Duan, J. Chen, L. Xie, H. Yao, and W. Gao, “Learning to distribute vocabulary indexing for scalable visual search,” *IEEE Transactions on Multimedia*, vol. 15, no. 1, pp. 153–166, 2013.
- [29] V. E. Liong, J. Lu, Y. P. Tan, and J. Zhou, “Deep coupled metric learning for cross-modal matching,” *IEEE Transactions on Multimedia*, vol. 19, no. 6, pp. 1234–1244, 2017.
- [30] —, “Learning a cross-modal hashing network for multimedia search,” in *IEEE International Conference on Image Processing*, 2017, pp. 3700–3704.
- [31] V. E. Liong, J. Lu, and Y. P. Tan, “Cross-modal discrete hashing,” *Pattern Recognition*, vol. 79, pp. 114–129, 2018.
- [32] V. E. Liong, J. Lu, Y. P. Tan, and J. Zhou, “Cross-modal deep variational hashing,” in *IEEE International Conference on Computer Vision*, 2017, pp. 4077–4085.
- [33] —, “Deep variational and structural hashing,” 2018, pp. 1–1.
- [34] —, “Deep video hashing,” *IEEE Transactions on Multimedia*, vol. 19, no. 6, pp. 1209–1219, 2017.
- [35] P. Indyk and R. Motwani, “Approximate nearest neighbors: towards removing the curse of dimensionality,” in *ACM Symposium on Theory of Computing*, 1998, pp. 604–613.
- [36] G. Shakhnarovich, T. Darrell, and P. Indyk, *Nearest-neighbor methods in learning and vision: theory and practice (neural information processing)*. The MIT press, 2006.

- [37] A. Andoni and P. Indyk, “Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions,” in *IEEE Symposium of Foundations of Computer Science*, 2006, pp. 459–468.
- [38] J. Wang, S. Kumar, and S.-F. Chang, “Semi-supervised hashing for large-scale search,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 12, pp. 2393–2406, 2012.
- [39] B. Kulis and K. Grauman, “Kernelized locality-sensitive hashing for scalable image search,” in *IEEE International Conference on Computer Vision*, 2009, pp. 2130–2137.
- [40] A. Andoni, P. Indyk, T. Laarhoven, I. Razenshteyn, and L. Schmidt, “Practical and optimal LSH for angular distance,” in *Advances in Neural Information Processing Systems*, 2015, pp. 1225–1233.
- [41] Y. Gong and S. Lazebnik, “Iterative quantization: A procrustean approach to learning binary codes,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 817–824.
- [42] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, “Supervised hashing with kernels,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 2074–2081.
- [43] K. He, F. Wen, and J. Sun, “K-means hashing: an affinity-preserving quantization method for learning binary compact codes,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2938–2945.
- [44] G. Lin, C. Shen, and A. van den Hengel, “Supervised hashing using graph cuts and boosted decision trees,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 11, pp. 2317–2331, 2015.
- [45] M. Hu, Y. Yang, F. Shen, N. Xie, and H. T. Shen, “Hashing with angular reconstructive embeddings,” *IEEE Transactions on Image Processing*, vol. 27, no. 2, pp. 545–555, 2018.
- [46] B. Kulis and T. Darrell, “Learning to hash with binary reconstructive embeddings,” in *Advances in Neural Information Processing Systems*, 2009, pp. 1042–1050.
- [47] X. Luo, L. Nie, X. He, Y. Wu, Z.-D. Chen, and X.-S. Xu, “Fast scalable supervised hashing,” in *ACM SIGIR Conference on Research and Development in Information Retrieval*, 2018, pp. 735–744.
- [48] F. Cakir, K. He, and S. Sclaroff, “Hashing with binary matrix pursuit,” in *European Conference on Computer Vision*, 2018, pp. 332–348.

- [49] 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.
- [50] 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.
- [51] J. Masci, M. M. Bronstein, A. Bronstein, and J. Schmidhuber, “Multimodal similarity-preserving hashing,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 4, pp. 824–830, 2014.
- [52] 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.
- [53] F. Zhao, Y. Huang, L. Wang, and T. Tan, “Deep semantic ranking based hashing for multi-label image retrieval,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1556–1564.
- [54] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen, “Deep learning of binary hash codes for fast image retrieval,” in *IEEE Conference on Computer Vision and Pattern Recognition Workshop*, 2015, pp. 27–35.
- [55] H. Lai, Y. Pan, Y. Liu, and S. Yan, “Simultaneous feature learning and hash coding with deep neural networks,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3270–3278.
- [56] H. Liu, R. Wang, S. Shan, and X. Chen, “Deep supervised hashing for fast image retrieval,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2064–2072.
- [57] R. Salakhutdinov and G. Hinton, “Semantic hashing,” *International Journal of Approximate Reasoning*, vol. 50, no. 7, pp. 969–978, 2009.
- [58] N. Srivastava and R. R. Salakhutdinov, “Multimodal learning with deep boltzmann machines,” in *Advances in Neural Information Processing Systems*, 2012, pp. 2222–2230.
- [59] K. G. Dizaji, F. Zheng, N. S. Nourabadi, Y. Yang, C. Deng, and H. Huang, “Unsupervised deep generative adversarial hashing network,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.
- [60] Z. Cao, M. Long, J. Wang, and P. S. Yu, “Hashnet: Deep learning to hash by continuation,” in *IEEE International Conference on Computer Vision*, 2017, pp. 5608–5617.

- [61] Y. Cao, M. Long, B. Liu, and J. Wang, “Deep cauchy hashing for hamming space retrieval,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1229–1237.
- [62] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, “Hashing with graphs,” in *International Conference on Machine Learning*, 2011, pp. 1–8.
- [63] W.-J. Li, S. Wang, and W.-C. Kang, “Feature learning based deep supervised hashing with pairwise labels,” in *International Joint Conference on Artificial Intelligence*, 2016, pp. 1711–1717.
- [64] T.-T. Do, A.-D. Doan, and N.-M. Cheung, “Learning to hash with binary deep neural network,” in *European Conference on Computer Vision*, 2016, pp. 219–234.
- [65] Q.-Y. Jiang and W.-J. Li, “Asymmetric deep supervised hashing,” in *AAAI Conference on Artificial Intelligence*, 2018.
- [66] X. Wang, Y. Shi, and K. M. Kitani, “Deep supervised hashing with triplet labels,” in *Asian Conference on Computer Vision*, 2016, pp. 1–15.
- [67] 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.
- [68] H. Jain, J. Zepeda, P. Perez, and R. Gribonval, “SUBIC: A supervised, structured binary code for image search,” in *IEEE International Conference on Computer Vision*, 2017, pp. 833–842.
- [69] X. Tang and X. Wang, “Face sketch recognition,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 1, pp. 50–57, 2004.
- [70] Q. Liu, X. Tang, H. Jin, H. Lu, and S. Ma, “A nonlinear approach for face sketch synthesis and recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 1005–1010.
- [71] 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 International Conference on Multimedia*, 2010, pp. 251–260.
- [72] A. Sharma and D. W. Jacobs, “Bypassing synthesis: PLS for face recognition with pose, low-resolution and sketch,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 593–600.
- [73] K. Wang, R. He, W. Wang, L. Wang, and T. Tan, “Learning coupled feature spaces for cross-modal matching,” in *IEEE International Conference on Computer Vision*, 2013, pp. 2088–2095.

- [74] C. Kang, S. Xiang, S. Liao, C. Xu, and C. Pan, “Learning consistent feature representation for cross-modal multimedia retrieval,” *IEEE Transactions on Multimedia*, vol. 17, no. 3, pp. 370–381, 2015.
- [75] Z. Xu, Y. Yang, and A. G. Hauptmann, “A discriminative cnn video representation for event detection,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1798–1807.
- [76] Y. Gong, Q. Ke, M. Isard, and S. Lazebnik, “A multi-view embedding space for modeling internet images, tags, and their semantics,” *International Journal of Computer Vision*, vol. 106, no. 2, pp. 210–233, 2014.
- [77] A. Mignon, F. Jurie *et al.*, “CMML: a new metric learning approach for cross modal matching,” in *Asian Conference on Computer Vision*, 2012, pp. 1–14.
- [78] S. Siena, V. N. Boddeti, and B. Kumar, “Maximum-margin coupled mappings for cross-domain matching,” in *IEEE International Conference on Biometrics: Theory, Applications, and Systems*, 2013, pp. 1–8.
- [79] S. Siena, V. N. Boddeti, and B. V. Kumar, “Coupled marginal fisher analysis for low-resolution face recognition,” in *European Conference on Computer Vision*, 2012, pp. 240–249.
- [80] J. Zhou, G. Ding, and Y. Guo, “Latent semantic sparse hashing for cross-modal similarity search,” in *ACM SIGIR Conference on Research and Development in Information Retrieval*, 2014, pp. 415–424.
- [81] G. Ding, Y. Guo, and J. Zhou, “Collective matrix factorization hashing for multimodal data,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2083–2090.
- [82] Z. Lin, G. Ding, M. Hu, and J. Wang, “Semantics-preserving hashing for cross-view retrieval,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3864–3872.
- [83] X. Xu, F. Shen, Y. Yang, and H. T. Shen, “Discriminant cross-modal hashing,” in *ACM International Conference on Multimedia Retrieval*, 2016, pp. 305–308.
- [84] 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.
- [85] F. Zheng, Y. Tang, and L. Shao, “Hetero-manifold regularisation for cross-modal hashing,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 5, pp. 1059–1071, 2018.

- [86] Y. Cao, M. Long, J. Wang, and H. Zhu, “Correlation autoencoder hashing for supervised cross-modal search,” in *ACM International Conference on Multimedia Retrieval*, 2016, pp. 197–204.
- [87] 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.
- [88] Y. Shen, L. Liu, L. Shao, and J. Song, “Deep binaries: Encoding semantic-rich cues for efficient textual-visual cross retrieval,” in *IEEE International Conference on Computer Vision*, 2017, pp. 4117–4126.
- [89] C. Li, C. Deng, N. Li, W. Liu, X. Gao, and D. Tao, “Self-supervised adversarial hashing networks for cross-modal retrieval,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4242–4251.
- [90] Y. Cao, B. Liu, M. Long, and J. Wang, “Cross-modal hamming hashing,” in *European Conference on Computer Vision*, 2018, pp. 202–218.
- [91] W. Yang, D. Yi, Z. Lei, J. Sang, and S. Z. Li, “2D-3D face matching using CCA,” in *IEEE Conference on Automatic Face and Gesture Recognition Conference*, 2008, pp. 1–6.
- [92] A. Sharma, A. Kumar, H. Daume, and D. W. Jacobs, “Generalized multiview analysis: A discriminative latent space,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 2160–2167.
- [93] K. Wang, R. He, L. Wang, W. Wang, and T. Tan, “Joint feature selection and subspace learning for cross-modal retrieval,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 10, pp. 2010–2023, 2016.
- [94] M. Kan, S. Shan, H. Zhang, S. Lao, and X. Chen, “Multi-view discriminant analysis,” in *European Conference on Computer Vision*, 2012, pp. 808–821.
- [95] X. Xu, A. Shimada, R.-i. Taniguchi, and L. He, “Coupled dictionary learning and feature mapping for cross-modal retrieval,” in *IEEE International Conference on Multimedia and Expo*, 2015, pp. 1–6.
- [96] C. Wang, H. Yang, and C. Meinel, “A deep semantic framework for multimodal representation learning,” *Multimedia Tools and Applications*, pp. 1–22, 2016.
- [97] C. Kang, S. Liao, Y. He, J. Wang, W. Niu, S. Xiang, and C. Pan, “Cross-modal similarity learning: A low rank bilinear formulation,” in *ACM International Conference on Information and Knowledge Management*, 2015, pp. 1251–1260.

- [98] L. Cao, Z. Li, Y. Mu, and S.-F. Chang, “Submodular video hashing: a unified framework towards video pooling and indexing,” in *ACM International Conference on Multimedia*, 2012, pp. 299–308.
- [99] G. Ye, D. Liu, J. Wang, and S.-F. Chang, “Large-scale video hashing via structure learning,” in *IEEE International Conference on Computer Vision*, 2013, pp. 2272–2279.
- [100] 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 *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4758–4767.
- [101] 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, 2019.
- [102] 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.
- [103] Y. Li, R. Wang, Z. Cui, S. Shan, and X. Chen, “Compact video code and its application to robust face retrieval in tv-series,” in *British Machine Vision Conference*, 2014, pp. 1–12.
- [104] 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.
- [105] C. Szegedy, A. Toshev, and D. Erhan, “Deep neural networks for object detection,” in *Advances in Neural Information Processing Systems*, 2013, pp. 2553–2561.
- [106] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, “Learning hierarchical features for scene labeling,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1915–1929, 2013.
- [107] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.
- [108] B. Kulis *et al.*, “Metric learning: A survey,” *Foundations and Trends® in Machine Learning*, vol. 5, no. 4, pp. 287–364, 2013.
- [109] R. Hadsell, S. Chopra, and Y. LeCun, “Dimensionality reduction by learning an invariant mapping,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2006, pp. 1735–1742.

- [110] F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 815–823.
- [111] G. Chechik, V. Sharma, U. Shalit, and S. Bengio, “Large scale online learning of image similarity through ranking,” in *Journal of Machine Learning Research*, 2010, pp. 1109–1135.
- [112] J. Hu, J. Lu, and Y.-P. Tan, “Discriminative deep metric learning for face verification in the wild,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1875–1882.
- [113] D. Yi, Z. Lei, S. Liao, and S. Z. Li, “Deep metric learning for person re-identification,” in *IAPR International Conference on Pattern Recognition*, 2014, pp. 34–39.
- [114] H. Li, Y. Li, and F. Porikli, “Deeptrack: Learning discriminative feature representations online for robust visual tracking,” *IEEE Transactions on Image Processing*, vol. 25, no. 4, pp. 1834–1848, 2016.
- [115] 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.
- [116] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, “Large-scale video classification with convolutional neural networks,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.
- [117] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, “Beyond short snippets: Deep networks for video classification,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4694–4702.
- [118] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” in *International Conference on Learning Representations*, 2014, pp. 1–14.
- [119] Y. Miao, L. Yu, and P. Blunsom, “Neural variational inference for text processing,” in *International Conference on Machine Learning*, 2016, pp. 1727–1736.
- [120] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, “Semi-supervised learning with deep generative models,” in *Advances in Neural Information Processing Systems*, 2014, pp. 3581–3589.
- [121] S. Chaidaroon and Y. Fang, “Variational deep semantic hashing for text documents,” in *ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017, pp. 75–84.

- [122] Z. Hu, J. Chen, H. Lu, and T. Zhang, “Bayesian supervised hashing,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6348–6355.
- [123] A. Li, S. Shan, X. Chen, and W. Gao, “Cross-pose face recognition based on partial least squares,” *Pattern Recognition Letters*, vol. 32, no. 15, pp. 1948–1955, 2011.
- [124] K. Q. Weinberger and L. K. Saul, “Distance metric learning for large margin nearest neighbor classification,” *Journal of Machine Learning Research*, vol. 10, pp. 207–244, 2009.
- [125] G. Hinton, “A practical guide to training restricted boltzmann machines,” *Momentum*, vol. 9, no. 1, p. 926, 2010.
- [126] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, “Return of the devil in the details: Delving deep into convolutional nets,” in *British Machine Vision Conference*, 2014.
- [127] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (voc) challenge,” *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.
- [128] S. J. Hwang and K. Grauman, “Accounting for the relative importance of objects in image retrieval,” in *British Machine Vision Conference*, 2010, pp. 1–12.
- [129] 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 *ACM International Conference on Image and Video Retrieval*, 2009, pp. 48.1 – 48.9.
- [130] S. Z. Li, D. Yi, Z. Lei, and S. Liao, “The CASIA NIR-VIS 2.0 face database,” in *IEEE Conference on Computer Vision and Pattern Recognition Workshop*, 2013, pp. 348–353.
- [131] J. Kim, J. Nam, and I. Gurevych, “Learning semantics with deep belief network for cross-language information retrieval,” in *International Conference on Computational Linguistics*, 2012, pp. 579–588.
- [132] R. Socher and L. Fei-Fei, “Connecting modalities: Semi-supervised segmentation and annotation of images using unaligned text corpora,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 966–973.
- [133] G. Toderici, H. Aradhye, M. Paşca, L. Sbaiz, and J. Yagnik, “Finding meaning on youtube: Tag recommendation and category discovery,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 3447–3454.
- [134] Y. Zheng, Y.-J. Zhang, and H. Larochelle, “Topic modeling of multimodal data: an autoregressive approach,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1370–1377.

- [135] (2012) FaceVACS software developer kit, cognitec systems. [Online]. Available: <http://www.cognitec-systems.de>
- [136] T. I. Dhamecha, P. Sharma, R. Singh, and M. Vatsa, “On effectiveness of histogram of oriented gradient features for visible to near infrared face matching,” in *IAPR International Conference on Pattern Recognition*, 2014, pp. 1788–1793.
- [137] F. Juefei-Xu, D. K. Pal, and M. Savvides, “NIR-VIS heterogeneous face recognition via cross-spectral joint dictionary learning and reconstruction,” in *IEEE Conference on Computer Vision and Pattern Recognition Workshop*, 2015, pp. 141–150.
- [138] D. Yi, Z. Lei, S. Liao, and S. Z. Li, “Shared representation learning for heterogeneous face recognition,” in *IEEE Conference on Automatic Face and Gesture Recognition Conference*, 2015, pp. 1–8.
- [139] Y. Jin, J. Lu, and Q. Ruan, “Coupled discriminative feature learning for heterogeneous face recognition,” *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 3, pp. 640–652, 2015.
- [140] A. Vedaldi and K. Lenc, “Matconvnet: Convolutional neural networks for matlab,” in *ACM International Conference on Multimedia*, 2015, pp. 689–692.
- [141] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Journal of Machine Learning Research*, 2010, pp. 249–256.
- [142] M. J. Huiskes and M. S. Lew, “The MIR Flickr retrieval evaluation,” in *ACM International Conference on Multimedia Information Retrieval*, 2008, pp. 39–43.
- [143] M. Grubinger, P. Clough, H. Müller, and T. Deselaers, “The IAPR TC-12 benchmark: A new evaluation resource for visual information systems,” in *International Conference on Language Resources and Evaluation*, vol. 5, 2006, p. 10.
- [144] K. Järvelin and J. Kekäläinen, “Ir evaluation methods for retrieving highly relevant documents,” in *ACM SIGIR Conference on Research and Development in Information Retrieval*, 2000, pp. 41–48.
- [145] D. J. Rezende, S. Mohamed, and D. Wierstra, “Stochastic backpropagation and approximate inference in deep generative models,” in *International Conference on Machine Learning*, vol. 32, no. 2, 2014, pp. 1278–1286.
- [146] Y. Cao, M. Long, J. Wang, H. Zhu, and Q. Wen, “Deep quantization network for efficient image retrieval,” in *AAAI Conference on Artificial Intelligence*, 2016, pp. 3457–3463.
- [147] Y. Cao, M. Long, J. Wang, and S. Liu, “Deep visual-semantic quantization for efficient image retrieval,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1328–1337.

- [148] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” Univ. of Toronto, Tech. Rep., 2009.
- [149] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, “Imagenet large scale visual recognition challenge,” *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.
- [150] A. Sablayrolles, M. Douze, N. Usunier, and H. Jegou, “How should we evaluate supervised hashing?” in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2017, pp. 1732–1736.
- [151] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *International Conference on Learning Representations*, 2014, pp. 1–15.
- [152] 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.
- [153] J. Martinez, J. Clement, H. H. Hoos, and J. J. Little, “Revisiting additive quantization,” in *European Conference on Computer Vision*, 2016, pp. 137–153.
- [154] H. Zhu, M. Long, J. Wang, and Y. Cao, “Deep hashing network for efficient similarity retrieval,” in *AAAI Conference on Artificial Intelligence*, 2016, pp. 2415–2421.
- [155] S. Chopra, R. Hadsell, and Y. LeCun, “Learning a similarity metric discriminatively, with application to face verification,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 539–546.
- [156] G. E. Hinton, “Training products of experts by minimizing contrastive divergence,” *Neural Computation*, vol. 14, no. 8, pp. 1771–1800, 2002.
- [157] Y.-G. Jiang, G. Ye, S.-F. Chang, D. Ellis, and A. C. Loui, “Consumer video understanding: A benchmark database and an evaluation of human and machine performance,” in *ACM International Conference on Multimedia*, 2011, pp. 29–37.
- [158] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, pp. 1–14, 2014.
- [159] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. Black, “Towards understanding action recognition,” in *IEEE International Conference on Computer Vision*, 2013, pp. 3192–3199.
- [160] G. Chéron, I. Laptev, and C. Schmid, “P-CNN: pose-based cnn features for action recognition,” in *IEEE International Conference on Computer Vision*, 2015, pp. 3218–3226.

- [161] G. Gkioxari and J. Malik, “Finding action tubes,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 759–768.
- [162] J. Wang, S. Kumar, and S.-F. Chang, “Semi-supervised hashing for scalable image retrieval,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 3424–3431.
- [163] M. Long, H. Zhu, J. Wang, and M. I. Jordan, “Unsupervised domain adaptation with residual transfer networks,” in *Advances in Neural Information Processing Systems*, 2016, pp. 136–144.
- [164] M. Long, Y. Cao, J. Wang, and M. I. Jordan, “Learning transferable features with deep adaptation networks,” in *International Conference on Machine Learning*, 2015, pp. 97–105.
- [165] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur, “Recurrent neural network based language model,” in *Annual Conference of the International Speech Communication Association*, 2010.
- [166] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [167] S. Ji, W. Xu, M. Yang, and K. Yu, “3D convolutional neural networks for human action recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, 2013.
- [168] J. Cheng, L. Dong, and M. Lapata, “Long short-term memory-networks for machine reading,” *Empirical Methods in Natural Language Processing*, 2016.
- [169] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, “Self-normalizing neural networks,” in *Advances in Neural Information Processing Systems*, 2017, pp. 972–981.
- [170] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and 0.5 mb model size,” *International Conference on Learning Representations*, 2017.
- [171] Y. Guo, G. Ding, J. Han, and Y. Gao, “SitNet: discrete similarity transfer network for zero-shot hashing,” in *International Joint Conference on Artificial Intelligence*, 2017, pp. 1767–1773.
- [172] H. Lai and Y. Pan, “Transductive zero-shot hashing via coarse-to-fine similarity mining,” *ACM International Conference on Multimedia Retrieval*, 2018.
- [173] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

- [174] S. Jin, “Deep saliency hashing,” *ACM International Conference on Multimedia*, 2018.
- [175] X. Zhang, H. Lai, and J. Feng, “Attention-aware deep adversarial hashing for cross-modal retrieval,” in *European Conference on Computer Vision*, 2018, pp. 591–606.
- [176] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh, “Vqa: Visual question answering,” in *IEEE International Conference on Computer Vision*, 2015, pp. 2425–2433.
- [177] K. Hegde, R. Agrawal, Y. Yao, and C. W. Fletcher, “Morph: Flexible acceleration for 3d cnn-based video understanding,” in *Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2018, pp. 933–946.
- [178] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, “Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks,” in *IEEE International Conference on Robotics and Automation*, 2017, pp. 1355–1361.