Hashing for Multimedia Similarity Modeling and Large-Scale Retrieval

2017

Kai Li
University of Central Florida

HASHING FOR MULTIMEDIA SIMILARITY MODELING AND LARGE-SCALE RETRIEVAL

by

KAI LI
B.S. Huazhong University of Science and Technology, 2010
M.S. University of Central Florida, 2015

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2017

Major Professor: Kien A. Hua

ABSTRACT

In recent years, the amount of multimedia data such as images, texts, and videos have been growing rapidly on the Internet. Motivated by such trends, this thesis is dedicated to exploiting hashing-based solutions to reveal multimedia data correlations and support intra-media and inter-media similarity search among huge volumes of multimedia data.

We start by investigating a hashing-based solution for audio-visual similarity modeling and apply it to the audio-visual sound source localization problem. We show that synchronized signals in audio and visual modalities demonstrate similar temporal changing patterns in certain feature spaces. We propose to use a permutation-based random hashing technique to capture the temporal order dynamics of audio and visual features by hashing them along the temporal axis into a common Hamming space. In this way, the audio-visual correlation problem is transformed into a similarity search problem in the Hamming space. Our hashing-based audio-visual similarity modeling has shown superior performances in the localization and segmentation of sounding objects in videos.

The success of the permutation-based hashing method motivates us to generalize and formally define the supervised ranking-based hashing problem, and study its application to large-scale image retrieval. Specifically, we propose an effective supervised learning procedure to learn optimized ranking-based hash functions that can be used for large-scale similarity search. Compared with the randomized version, the optimized ranking-based hash codes are much more compact and discriminative. Moreover, it can be easily extended to kernel space to discover more complex ranking structures that cannot be revealed in linear subspaces. Experiments on large image datasets demonstrate the effectiveness of the proposed method for image retrieval.

We further studied the ranking-based hashing method for the cross-media similarity search problem. Specifically, we propose two optimization methods to jointly learn two groups of linear subspaces, one for each media type, so that features ranking orders in different linear subspaces maximally preserve the cross-media similarities. Additionally, we develop this ranking-based hashing method in the cross-media context into a flexible hashing framework with a more general solution. We have demonstrated through extensive experiments on several real-world datasets that the proposed cross-media hashing method can achieve superior cross-media retrieval performances against several state-of-the-art algorithms.

Lastly, to make better use of the supervisory label information, as well as to further improve the efficiency and accuracy of supervised hashing, we propose a novel multimedia discrete hashing framework that optimizes an instance-wise loss objective, as compared to the pairwise losses, using an efficient discrete optimization method. In addition, the proposed method decouples the binary codes learning and hash function learning into two separate stages, thus making the proposed method equally applicable for both single-media and cross-media search. Extensive experiments on both single-media and cross-media retrieval tasks demonstrate the effectiveness of the proposed method.

I would like to dedicate this dissertation to my wife and my parents for always being there and unconditionally support me, encourage me and congratulate me.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my academic adviser, Dr. Kien A. Hua for his continued guidance, strong support and valuable advising throughout my Ph.D. study. He introduces me into the world of research and teaches me scientific thinking, writing and presentation skills. He has always been supportive for my choice of research problems and been available whenever I need his advice. This dissertation would not have been possible without his guidance.

I would like to thank Dr. Guo-Jun Qi for his important insight and advice with my research. He is both a friend and and an adviser; our discussion and collaboration have led to important publications for my dissertation research.

I would like to thank Dr. Haiyan Hu and Dr. Morgan C. Wang, for their efforts in serving in my dissertation committee and providing valuable guidance and suggestions to my dissertation.

I would also like to thank the DSG members, Dr. Jun Ye, Dr. Faisal Amjad, Kutalmis Akpinar, Omar Nakhila, Trevor Ballard, Sansiri Tarnpradab, Naifan Zhuang, Yusuph Turgun, Kevin Joslyn, Hao Hu, Liheng Zhang, Affra Attiah, Fereshteh Jafariakinabad, for being good friends and making my Ph.D. study such an enjoyable and memorable experience.

Finally, I am especially thankful to my wife, Ronglu, and my parents, Baodi and Jinxiu, for offering unwavering love and wholehearted support. Their love and support have always been so important for my Ph.D. study, and for my life and career.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1 Problem Statement and Motivation

Thanks to the rapid advancement of information technologies, the last decade has witnessed unprecedented growth in multimedia content generated by all kinds of digital electronic devices, such as digital cameras, mobile phones, and tablets etc. The ubiquitous multimedia big data presents a number of challenges and opportunities for research and development of efficient storage, indexing, and retrieval techniques.

Hashing is recognized by many researchers as a promising solution to some of the aforementioned big data problems. Typically, hashing algorithms transform high-dimensional data representations into compact binary codes, and such a transformation enjoys several compelling benefits. Firstly, binary hash codes require significantly less storage compared with the high dimensional floating point representations. For instance, it only takes 8GB space to store 1 billion 64-bits hash codes, which can be easily loaded into the memory of a single PC; while if the data points are represented by 4096-dimensional CNN features, one would need 20TB memory to store the same amount of data. Secondly, the similarity between data points can be computed with Hamming distance through bit-wise "XOR" operation, which is the fastest atomic operations supported by modern computers. In fact, the Hamming distance computation is several orders of magnitude faster than that of the Euclidean distance between two vectors of the same dimension. Lastly, the binary hash codes can be naturally used as the index to build hash tables and support sub-linear or constant-time lookup, which offers even more aggressive speedup when one needs to search for similar items in massive-scale data repositories.

In light of these inherent speed and storage advantages, hashing, especially learning-based hashing, has attracted considerable research attentions during the past few years [42, 87, 58, 77, 10, 54]. Typically, learning-based hashing methods learn binary hash codes by preserving certain similarity structure among the training data items from a specific dataset. These hash codes can be optimized to be very compact and discriminative, thus achieving promising efficiency and accuracy for a number of similarity search tasks, including, but not limited to, image retrieval [58], video retrieval [101], and cross-media retrieval [10]. Moreover, hashing-based methods can not only be used for retrieval, they have also been applied to a range of other relevant computer vision and machine learning problems that can be implicitly modeled as the nearest neighbor search problem, such as classification [52], recommendation systems [100], face recognition [78], video analysis [95], object detection [82], etc.

Although hashing methods have achieved great success in many applications, the research on this topic is still far from full-fledged, in terms of both theory and application. One the one hand, there are many open problems in developing better hash learning algorithms for large-scale similarity search. For instance, since most of the existing hashing algorithms are restricted to a specific type of hash function based on binary space partitioning, it remains to be answered as to whether and how new types of hash functions could be exploited in the hash learning process. Some other open problems include how to design scalable optimization methods that can take full advantage of the large-scale training set, how to develop flexible learning methods that can be easily combined with different objective functions, and how to harness the recent advancement of deep learning techniques to learn more discriminative hash codes, etc. On the other hand, it also requires much research efforts to discover good applications of the existing hashing methods. In fact, the novel application of existing hashing methods can sometimes lead to good results in an established research problem. For instance, the hashing-based collaborative filtering method in [100] achieves much better performance in recommendation systems than conventional matrix factorization methods; the hashing-based deep compression algorithm in [19] greatly reduces the storage requirements of deep neural networks without sacrificing much generalization performance.

1.2 Our Contributions

Motivated by the great potential of hashing algorithms in different applications, as well as the limitations of the current research, we focus on both theoretical and application aspect of hashing research in this dissertation. Specifically, we summarize the major contributions of this dissertation as follows.

First, we propose a novel application of the random permutation-based hashing method to the audio-visual correlation problem. One of the challenges in multimodal video understanding concerns the localization and segmentation of sounding object through audio-visual correlation analysis, which has been mostly tackled in a controlled environment with multiple microphones. We address a more challenging version of this problem by considering general consumer videos taken with a single camera and microphone. We take a hashing approach towards this problem by transforming the correlation analysis problem into audio-visual temporal similarity modeling. Specifically, we decompose videos into collections of spatiotemporal objects and represent each spatiotemporal object by their temporal motion features. Meanwhile, we represent audio signals with the temporal energy features. Then we apply temporal hashing to the audio and visual features, thus mapping them to a common Hamming space and transforming the original problem into a nearest neighbor search problem. Our experiments demonstrate that this hashing-based approach achieves promising localization and segmentation performance in standard benchmark videos.

Second, we present a novel supervised hashing algorithm for single-media retrieval. Specifically, we explore a new type of hash function based on the ranking of feature subspaces, and it is referred to as ranking-based hashing hereinafter. The ranking-based hash function can be seen as a generalization of the random permutation hashing, which is closely related to rank correlation measures [64] that have been well-deemed as robust measures in many performance evaluation schemes. The ranking-based hash learning problem is formulated as the optimization of a highly non-convex and discontinuous objective function, which is then relaxed and solved by a simple yet effective iterative learning algorithm. We further embed such learning algorithm in a boosted sequential learning framework to learn multiple uncorrelated and informative hash codes. The overall learning procedure generates a sequence of optimal low-dimensional ranking subspaces where the similarities among data samples are maximally preserved. Experimental comparison with existing supervised hashing techniques on several large-scale image datasets demonstrates the effectiveness of the proposed hashing method for image retrieval.

Third, we further extend the ranking-based hashing method into a cross-media hashing framework, which transforms multimedia data into a common Hamming space to support cross-media retrieval. Specifically, we learn two groups of linear subspaces jointly, one for each modality, such that the ranking ordering in one subspace is maximally aligned with that of the other. We present two alternative approaches to solving such cross-media learning problem: one is based on the minimization of a piece-wise linear upper bound of the original objective function, and the other is based on continuous relaxation. We have also presented a more general formulation of the ranking-based hash learning problem in the cross-media context, and show that the learning problem in single-media is a degenerate version of this general formulation under certain conditions. We demonstrate through extensive experiments on real-world datasets that the proposed cross-media hashing algorithm achieves competitive performances compared with state-of-the-art techniques for a number of cross-media retrieval benchmarks.

Lastly, we present a novel multimedia hashing framework, termed as Label Preserving Multimedia Hashing (LPMH) for multimedia similarity search. In LPMH, a general optimization method is used to learn the joint binary codes of multiple media types by explicitly preserving the semantic label information. The proposed optimization strategy is not tied to any specific loss function, and can easily incorporate bit balance constraints to produce well-balanced binary codes. Specifically, our formulation leads to a set of Binary Integer Programming (BIP) problems that have exact solutions both with and without the bit balance constraints. These problems can be solved extremely fast and the solution can easily scale up to large-scale datasets. In the hash function learning stage, the boosted decision trees algorithm is utilized to learn multiple media-specific hash functions that can map heterogeneous data sources into a homogeneous Hamming space for cross-media retrieval. We have comprehensively evaluated the proposed method using a range of large-scale datasets in both single-media and cross-media retrieval tasks and show competitive results against state-of-the-art methods in both speed and accuracy.

1.3 Proposal Organization

The rest of this dissertation is organized as follows:

Chapter 2 presents the literature review, which consists of three parts: audio-visual correlation, single-media hashing, and cross-media hashing.

Chapter 3 presents a hashing-based approach for the audio-visual correlation problem in videos taken with a single camera and microphone.

Chapter 4 presents a supervised hashing method based on feature subspace rankings for image retrieval.

Chapter 5 presents the cross-media ranking hashing framework and studies its performance on cross-media retrieval.

Chapter 6 presents a general approach for solving semantics preserving binary hash codes for multimedia hashing and lays out a tentative experimental plan to carry out the proposed work for the remaining time of the dissertation.

CHAPTER 2: LITERATURE REVIEW

In this chapter, we present a brief review of the existing literature that is relevant to the problems investigated in this dissertation. This literature review is divided into three sections, discussing the related works in audio-visual correlation, single-media retrieval, and cross-media retrieval separately. Although retrieval problems have also been tackled with various different approaches, we only review hashing-based solutions as it is the focus of this dissertation. As for audio-visual correlation/localization, since our method is the first hashing-based method, the focus of the review will be on existing non-hashing methods.

2.1 Audio-visual Corrlelation

The existing work that localizes visual objects associated with audio signals using a single microphone approximately fall into two categories: pixel-level localization [39], [79], [5], [16] and object level localization and/or segmentation [34], [15], [61], [17]. In [39], Kdiron et al. propose to use Canonical Correlation Analysis (CCA) to find the image pixels that are most correlated with audio signals. The authors reveal the ill-posedness of CCA due to the high dimensionality of visual features and insufficient samples of stationary signals, and exploit the spatial sparsity of audiovisual events to seek sparse solution with L-1 norm. Sigg [79] further consolidates the problem of CCA in its original form and presents a reformulation that incorporates nonnegativity and sparsity constraints on the coefficients of projection directions. Through this reformulation, the author is able to locate sound sources in a test movie and separate the corresponding audio signals by filtering. Barzelay et al. [5] address the problem of audiovisual source separation in both modalities. They represent the audio and visual signals as audio and visual onsets. The audio and visual onsets measure the drastic change of audio and visual features respectively. The correlation of audio onset with each visual onset is evaluated using a simple coincidence-based measure. A major drawback of pixel-level correlation technqiues is that they are sensitive to visual noises and the localization results (i.e. isolated pixels) do not carry too much high-level semantic meaning. Therefore, those methods are generally not very useful for higher-level reasoning that builds upon the results of audiovisual correlation.

On the other hand, object-level correlation techniques aim at a higher level of abstraction that carries more semantic information about what makes the sound. Casanovas et al. [16] propose to use non-linear diffusion to focus on the audio source in visual domain. The diffusion process is controlled by a diffusion coeffecient based on an estimate of the synchrony between audio energy and motion in the video. The result of the diffusion process is an image region whose motion is most consistent with changes of audio energy. In [60], Liu et al. propose an algorithm to find quasi-stationary speaker faces using audiovisual correlation. The video in a time window is analyzed and the audio source is found by using Quadratic Mutual Information. The analyzed results are incorporated into a Graph-cut based image segmentation to extract the face region of the speaker. The same authors further extend their work to locate general non-stationary sound sources [61]. In this later work, a motion inconsistency measure of small spatial-temporal patches (ST-patch) centered at a pixel is used as the pixel's visual feature. A similar inconsistency measure is defined for audio energy as the audio feature. The audiovisual correlation is analyzed using the incremental Mutual Information, which is able to find each pixel's visual trajectory that best matches the audio energy change. Similar to [60], the results of such audiovisual analysis is fed into a segmentation algorithm to extract the sounding object. In [15], the video signals are first decomposed into a number of video atoms. The sound source is subsequently reconstructed by clustering the visual atoms that have a high audiovisual correlation. This technique tends to extract circular-shaped region irrespective of the original shape of the actual source, because all visual atoms within a radius are used in the reconstruction. The authors address this limitation in [17] by using video diffusion followed by a Graph cut segmentation procedure that keeps together pixels in regions with high audiovisual synchrony.

These techniques are similar in the sense that they first seek audiovisual correlation at a finer level (i.e. pixels or small atoms) and then apply clustering or segmentation afterwards to group pixels based on the synchrony. The limitation of such methods is that the extracted sounding object boundaries are quite irregular due to the noise in fine-grained synchronization analysis and the shape of the object hardly observes the shape of the actual object. A different method that reverses the above correlation-before-segmentation process is proposed in [34]. [34] first oversegments each video frame into small segments. K-means clustering is then applied to segments in the entire video such that each resultant spatial-temporal visual cluster represents an object. The velocity and acceleration of visual clusters and the audio Mel-frequency Cepstral Coefficients (MFCCs) are used as the audio and visual features respectively. Canonical Correlation Analysis (CCA) is finally used to identify the objects most correlated with the audio signal.

2.2 Single-media Hashing

Learning-based hashing can be generally classified as unsupervised or supervised. Unsupervised hashing aims to approximate metric similarity in the original feature space and earlier works on unsupervised hashing include Spectral Hashing (SH) [87], PCA Hashing [96], Kernelized Locality Sensitive Hashing (KLSH) [43], and Binary Reconstructive Embedding (BRE) [42]. Earlier works are mostly limited by the retrieval accuracy, and in order to improve the retrieval accuracy, quantization and graph-based methods are proposed. Quantization methods approximate kNN search by minimizing the quantization loss with respect to the original features. Representative works in this category includes Iterative Quantization (ITQ) [30], Optimized Tree Quantization (OTQ) [3], Sparse Composite Quantization (SQ) [104] and Sparse Projection Hashing (SPH) [92]. Graph-based methods exploit the graph structure of data similarities for hash learning. For example, Anchor Graph Hashing (AGH) [59] and Discrete Graph Hashing (DGH) [57] use anchor graphs to capture the neighborhood structure inherent in a given dataset and adopt a discrete optimization procedure to achieve nearly balanced and uncorrelated hash bits.

On the other hand, supervised hashing takes advantage of semantic labels to learn data-dependent hash functions. It has been shown that supervised hashing methods are able to learn more discriminative hash codes with the inclusion of the label information. For instance, Minimal Loss Hashing (MLH) [67] formulates the linear hash learning problem using structural SVM and minimizes the loss-adjusted upper bound of a hinge-like loss function defined on pairwise similarity labels. The resulting hash codes are shown to give superior performance. The same authors also extend this method to minimize loss functions defined on triplet similarities [68]. Similarly, Column Generation Hashing (CGH) [51] also learns hash functions based on triplet similarities. The difficulty in generating triplet similarity labels and the prohibitive training costs (i.e. O(n^3) triplets), however, limit the application of triplet similarity based hashing algorithm.

Neural networks and deep learning is also an emerging research area in supervised hashing. Specifically, end-to-end deep hashing methods based on Convolutional Neural Networks (CNN) [41] are being proposed lately. Although CNN-based hashing methods achives competitive performances, its focus is slightly different than traditional hashing methods. Specifically, the focus of traditional hashing research is to learn good hash functions with fixed feature representations, while end-to-end hashing focuses on how to combine representation learning and hash function learning in a synergic manner. Notable examples of deep hashing methods include CNN hashing (CNNH) [91], Deep Quantization Network (DQN) [14], Deep Semantic Ranking Hashing (DSRH) [107], Very Deep Supervised Hashing (VDSH) [106], Simultaneous Feature Learning and Hash Coding (SFHC) [45], etc.

Sequential learning algorithms that learn one bit at a time are found to be very effective in the hash learning task. For example, Boosting Similarity Sensitive Coding (BSSC) [76] and Forgiving Hash (FGH) [4] treat each hash bit as a week classifier and learn a series of hash functions in an AdaBoost framework. Sequential Projection Learning (SPLH) [86] fits the eigenvector solution into a boosting framework and uses pseudo labels in learning each hash bit. Supervised Hashing with Kernels (KSH) [58] sequentially learns compact hash codes by minimizing Hamming distances of similar pairs and maximizing that of dissimilar pairs simultaneously in the kernel space. The convergence properties of arbitrary sequential learning algorithms are theoretically proved in [29] and the Jensen Shannon divergence (JSD) sequential learning method is proposed with a multi-class classification formulation.

More recently, Two-Step Hashing (TSH) [53] decomposes the hash learning problem into the binary hash bit learning step and the boosted hash function learning step. TSH has outperformed many previous state-of-the-arts in a number of retrieval tasks. Other representative two-step methods include Latent Factor Hashing (LFH) [102], Column Sampling based Discrete Supervised Hashing COSDISH [38], Supervised Discrete Hashing (SDH) [77], DIscrete Supervised Hashing (DISH) [103], and Fast Hash (FastH) [52].

In sum, most of the existing learning-based hashing methods are restricted to a specific type of hash function (i.e. the sign function), while ranking-based hash functions are relatively under-researched. To the best of our knowledge, there has been no previous work explicitly studying ranking-based hash functions in the supervised hashing research.

2.3 Cross-media Hashing

Although hashing for single-media data has been extensively studied in the past decade, cross-modal [1] hashing starts to receive increasing attention only very recently.

Cross-Modal Similarity Sensitive Hashing (CMSSH) [10] and Cross-View Hashing (CVH) [44] are arguably the earliest works on this topic. CMSSH sequentially constructs two groups of linear hash functions and explicitly minimizes the distances between the Hamming space embeddings of data from different modalities. CVH extends the unimodal hashing method, Spectral Hashing (SH) [87], to consider both intra-view and inter-view similarities with a generalized eigen-

[1] We use 'single-/cross-modal' and 'single-/cross-media' interchangeably in this dissertation.

value formulation.

Several new methods were proposed soon after CMSSH and CVH. Iterative Multi-View Hashing (IMVH) [72] learns discriminative hash functions by solving a series of binary label assignment problems. Co-Regularized Hashing (CRH) [108] learns single-bit cross-modal hash functions by solving DC (i.e. difference of convex function) programs; and multiple bits are sequentially learned using boosting. The same authors of CRH also propose Multimodal Latent Binary Embedding (MLBE) [109], which takes a probabilistic generative approach and achieves competitive performance. However, the prohibitive computational costs for out-of-sample extensions limit the applications of MLBE to large-scale datasets.

In order to balance performance and computational costs, several new methods are proposed. For example, (PLMH) [97] extends MLBE to learn parameterized hash functions as the linear combination of a small set of anchor points. Similar ideas have also been exploited in Linear Cross-Modal Hashing (LCMH) [111], where a small set of cluster centroids are used in a similar fashion to the anchor points in PLMH. Semantic Correlation Maximization (SCM) [99] integrates semantic label information into a learning procedure with closed-form solutions and scales to large datasets by avoiding the explicit computation of pairwise similarity matrix. Inter-Media Hashing (IMH) [80] incorporates both labeled and unlabeled data to explore correlations among multiple media types from large-scale data sources.

More recently, Sparse Multi-Modal Hashing (SM^{2}H) [90] is proposed to obtain sparse codesets for data objects across different modalities through joint multi-modal dictionary learning. Latent Semantic Sparse Hashing (LSSH) [110] and Collective Matrix Factorization Hashing (CMFH) [26] use sparse coding and matrix factorization to capture the latent semantic features of different modalities. Supervised Matrix Factorization Hashing (SMFH) [56] also uses matrix factorization, however with the addition of graph-regularization. Semantics-Preserving Hashing (SePH) [54] transforms the similarity matrix into a joint probability distribution and approximates the distribution using nonlinear functions of the pairwise Hamming distance. Quantized Correlation Hashing (QCH) [89] considers both intra-modality quantization loss and inter-modality correlation in a single multi-modal objective function. The multi-modal objective function is further transformed to a unimodal formulation and optimized through an alternative procedure. Similar quantization-based cross-modal hashing algorithms include Cross-Modal Collaborative Quantization (CMCQ) [105] and Alternating Co-Quantization (ACQ) [33]. Semantic Topic Multimodal Hashing (STMH) [85] learns a common feature subspace from multimodal semantic concepts, and encode a hash bit by examining the existence of a concept.

Neural networks and deep learning have also been used in cross-modal hashing [112, 63, 84, 12, 11]. Specifically, end-to-end deep cross-modal hashing frameworks, such as Deep Cross-modal Hashing (DCMH) [35] are starting to receive attention lately. However, in contrast to the conventional hash learning research, where the focus is to learn good hash functions given certain feature representations, end-to-end hashing focuses on the seamless combination of feature learning and hash learning, in a way that the feature representations can be optimized for hash learning through error back-propagation. The competitive performance shown in [35] demonstrates the efficacy of combining hash learning and feature learning methods.

CHAPTER 3: HASHING FOR AUDIO-VISUAL CORRELATION ANALYSIS

3.1 Background

Multimodal information fusion concerns the study of integrating information from multiple sensory modalities that describe the same event. It has been shown for long that combining complementary information from different modalities can improve the performance of a system that uses only a single modality [27][7][36]. An intuitive explanation to this fact is that real-life events are inherently multimodal, while an unimodal system utilizes only a portion of the available information leaving the rest wasted. One particular research area that takes advantage of multimodal information fusion is audiovisual analysis that combines visual information with correlated audio information for improved performance in tasks such as event detection [22], speech recognition [81], video concept classification [37] etc.

Although audio and visual modality provides complementary information to each other, one should be aware that, they may also bring in noise. For example, combining unrelated audio background in the detection of a visual event may undermine rather than boost the performance of an audiovisual event detection system; an irrelevant talking face might distract the audiovisual speech recognition system from focusing on the right person whose speech is being interpreted. The above problems can be effectively alleviated if the system can correctly associate the audio signal with the right sound source in the video. The ability to associate sound with the right visual object is essential for many applications. For example, a pan-tilt camera can automatically follow the active speaker in video conferencing [18];the robot can correctly identify the target that is speaking to her during interaction with a human. Other applications include audio-video stream synchronization [34], audio source separation [88], lip reading [27], object tracking [6][113] etc.

In this chapter, we address the audiovisual correlation problem in general videos captured using a single microphone. The sound sources we deal with in this paper is not restricted to any specific type (e.g. a talking person), but rather it could be any object whose motion generates sound. For example, the sound source could be a musical instrument, a talking face, or even a bouncing basketball. Moreover, we aim to accurately segment out the sounding object (i.e. an image structure). Such audiovisual correlation problem is known to be hard for several reasons: there are always multiple distracting motions in the video occasionally synchronous with the soundtrack; signals captured by audio and visual sensors are essentially different in terms of spatial-temporal resolution and semantic meaning.

Most earlier research on sound source localization either rely on multiple microphones in controlled environment [88], or deal with quasi-stationary [16] or specific sound sources such as a talking face [60]. Some recent work [5][79][39] attempt to remove such restrictions and achieve sound source localization in general videos. However, few of them attempt segmentation of the audiovisual object, that is, they merely identify a number of pixels scattered around the object. We believe cross-modal localization of image structures is more semantically meaningful compared to identifying isolated pixels. To achieve this goal, we propose a novel method for simultaneous localization and segmentation of the visual object that is most correlated with the primary audio signals. We use a two-step video segmentation method to represent the entire video as a set of spatial-temporal (ST) region tracks, each representing the temporal evolution of an object. The motion features of those ST tracks are used as the visual features of the objects. The audio energy features are used for audio signals to capture the temporal variations. An ordering-based hashing approach, the Winner-Take-All Hash [93], is then applied to audio and visual features to generate the audio and visual hash codes. The similarities between the visual objects and audio signals are then computed as the Hamming distance between the hash codes. Note that a portion of this work, including the techniques and results, have previously been published by the author in conferences [50].

The remainder of this chapter is organized as follows. Section 3.2 gives an overview of the proposed method. Section 3.3 explains audio and visual features we used for correlation analysis. Section 3.4 describes our audiovisual correlation method. In Section 3.5, we present the experimental results on the test videos. Finally, this chapter is summarized in Section 3.6.

3.2 An Overview of the Method

This section provides a brief overview of the general steps of the proposed algorithm, leaving the details to be discussed in Section 3.3 and Section 3.4.

As illustrated in Fig. 3.1, the overall process involves two phases: audiovisual feature extraction and audiovisual correlation. In the audiovisual feature extraction phase, audio and visual descriptors are extracted independently. The audio descriptors we use is the smoothed audio energy. Briefly, this is obtained by applying short-term fourier transform to the raw audio signal followed by an integration over all frequencies at each time instant. The resultant audio energy descriptor is further smoothed using a Gaussian filter. The audio energy descriptor represents the change of audio signal strength over time, which we believe is caused by correlated motion in the visual domain. Such audio representation bridges the sampling rate gap between audio and visual signals, thus making it possible to perform audiovisual correlation analysis under the same temporal resolution.

Compared with audio signals, visual signals are characterized by high spatial resolution. We bridge this gap by first using a region tracking algorithm to consistently label the same image structure (i.e. regions in different frames that corresponds to the same semantic object/part) across all video frames. Each label generated in this manner indexes a spatial-temporal region track (STRtrack). In this way, the whole video $I(x, y, t)$ is decomposed into a set of N STRtracks $\{I_1(x, y, t), I_2(x, y, t), \cdots, I_N(x, y, t)\}$, where $t = 1, 2, \cdots, T$ is the frame index, and x, y is pixel's coordinate in a frame. For the i^{th} STRtrack, its motion descriptor associated with the t^{th} frame $m_i(t)$ is then computed as the average acceleration of all pixels labeled as i in the frame.

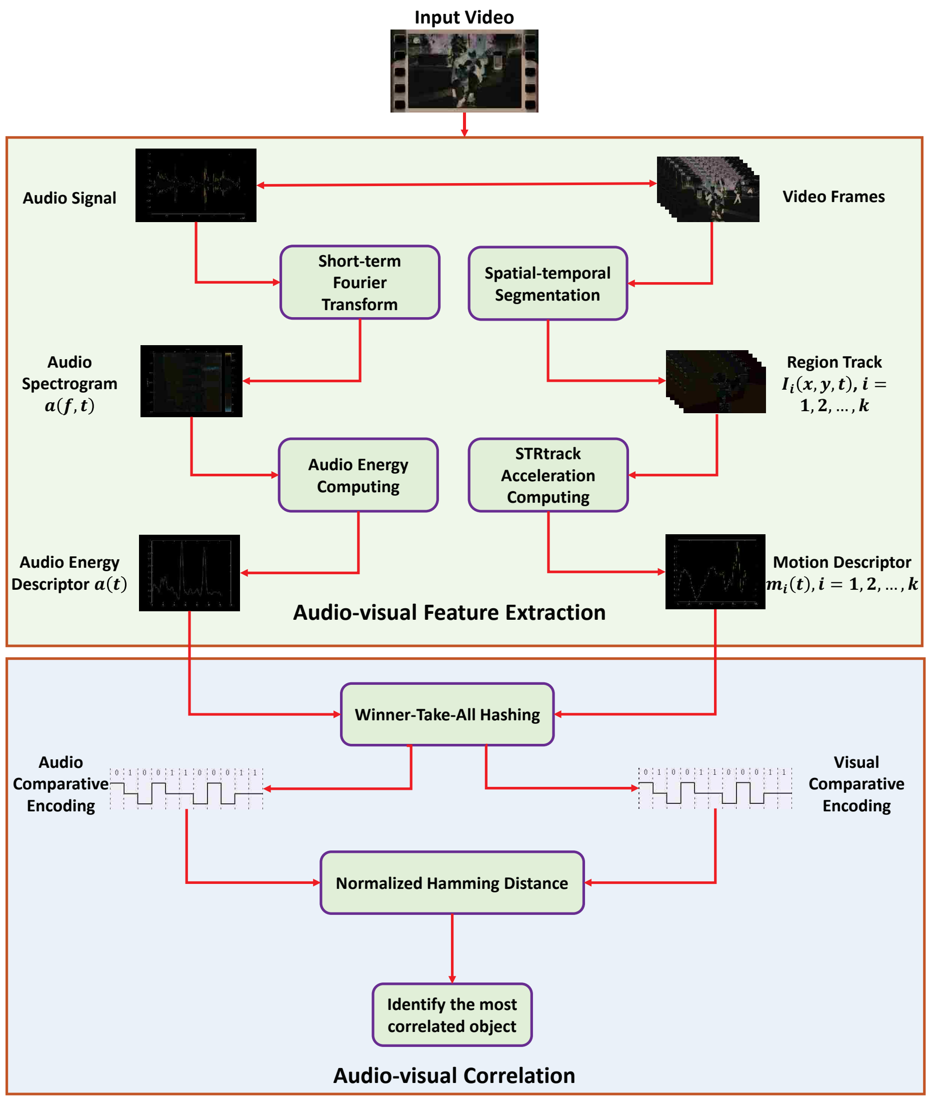

Input Video
Audio Signal
Video Frames
Short-term Fourier Transform
Spatial-temporal Segmentation
Audio Spectrogram $a(f,t)$
Region Track $I_i(x,y,t), i = 1,2,\ldots,k$
Audio Energy Computing
STRtrack Acceleration Computing
Audio Energy Descriptor $a(t)$
Motion Descriptor $m_i(t), i = 1,2,\ldots,k$
Audio-visual Feature Extraction
Winner-Take-All Hashing
Audio Comparative Encoding
Visual Comparative Encoding
Normalized Hamming Distance
Identify the most correlated object
Audio-visual Correlation

Figure 3.1: Block diagram of the proposed audiovisual source localization method. The audio and visual features are extracted independently. The audio spectrogram is computed by applying short-term fourier transform to the audio samples with proper framing. Next, audio power spectrum is integrated over all frequencies to get the audio energy. For the video frames, a two-step segmentation is applied to each frame, and a region tracking algorithm is used to obtain a consistent labeling of all videos. Regions sharing the same label form a spatial-temporal region track. Each region track is represented by a vector of its average acceleration in different frames. Both audio and visual features go through the same hashing that generates a single audio code for the audio signal and one visual code for each of the spatial temporal region track. The correlation between each region track and the audio signal is simply computed as the Hamming distance between each visual code and audio code.

Therefore, the video is compactly represented as a number of visual descriptors $\{m_1(t), \cdots, m_N(t)\}$.

The audiovisual correlation takes as input the resultant audio and visual descriptors and outputs the correlation scores χ_i for each STRtrack. We believe that the sounding object's motion pattern must be highly correlated to the audio energy in some feature space. We propose to capture this similarity using the Winner-Take-All (WTA) hashing [93]. The correlation between audio and visual modalities can be simply computed as the Hamming distance between the two hash codes.

3.3 Audio and Video Representation

The compact representation of audio and video signals are critical for the effectiveness of the proposed method. Our motivation for the proposed audio and video representation is to seek a compact yet informative representation that retains the most relevant information in both modalities and allows for audiovisual comparison in a common feature space.

3.3.1 Audio Representation

The audio signal $f(t)$ is transformed into the time frequency domain using the Short-term Fourier Transform [1] (STFT) that provides simultaneous time and frequency localization. The capability of STFT for analyzing time-varing, non-stationary signals makes it applicable for general sound sources.

In detail, the signal is first framed according to the frame rate of the video such that each audio frame corresponds to a video frame. The framing window size h is chosen such that neighboring windows overlap by 50% of the window size. Fourier transform is then applied to the sampling points within each window to get the spectrogram of the audio signal at that frame. Finally, the audio energy feature $a(t)$ is attained by integrating the spectrogram over all frequencies.

Formally, the above process can be represented as

$$a(t) = \int_0^\infty \int_0^T f(t').W(t'-t).e^{-j2\pi f t'} dt' df,$$

where T is the length of the audio, and the windowing function $W(t)$ is defined as

$$W(t) = \begin{cases} 1 & \text{if } |t| < h/2 \\ 0 & \text{otherwise.} \end{cases}$$

The resultant audio energy descriptor $a(t)$ is further filtered using a 1-D Gaussian kernel. Intuitively, the audio energy descriptor captures the changing patterns of audio signal strength in the same temporal resolution as the video frames.

3.3.2 Video Representation

The basic idea of the proposed video representation method is to extract and analyze the motion patterns of all objects present in a scene. We define an object to be an appearance-motion-coherent image structure. We aim to identify image structures corresponding to the same object across all video frames so as to analyze the object's motion patterns.

As illustrated in Fig. 3.2, we use a region tracking procedure to propagate first-frame segmentation labeling to all the rest frames. The procedure starts with a two-step segmentation for the first frame. In detail, we first apply Mean shift segmentation of the color image using LUV color features. If the resultant number of small regions is greater than a predefined threshold k, they are further clustered using location and motion features. The reason for such 2-step frame segmentation is to combine color and motion features for segmenting images into regions with highly coherent motion and appearance. To be specific, each region is represented as a 5-dimensional feature vector where the first two dimensions are the spatial centroid coordinates and the last three dimensions are the region's average LUV color values in the color-coded optical flow image. Formally, we compute the forward and backward dense optical flow $\mathbf{F}^{+}(x,y,t)$ and $\mathbf{F}^{-}(x,y,t)$ for each frame, where $\mathbf{F}^{+}(x,y,t)$ is the 2-D flow vector at pixel (x,y) estimated between frame t and $t+1$, and $\mathbf{F}^{-}(x,y,t)$ is that from t to $t-1$. The average flow is computed as

$$\mathbf{F}(x,y,t) = \frac{1}{2}(\mathbf{F}^{+}(x,y,t) - \mathbf{F}^{-}(x,y,t))).$$

Since spatial coordinates and the optical flow values are in different metric spaces, they need to be transformed into a into the LUV color feature space such that color differences can be measured by the Euclidean distance. In detail, the normalized optical flow (u_x, u_y) is converted to its polar representation (ρ, θ). Then the direction θ is used to access a discretized RGB color wheel adapted from [75]. The accessed RGB values are modulated with ρ to obtain the final RGB colors. The resultant RGB color image are further transformed into the LUV color space.

After the frame segmentation procedure, each region is represented using its color histogram. Specifically, the LUV color space is quantized into $n = 5^3 = 125$ bins and the number of pixels in each bin is counted. The regions in the first frame (i.e. their color histograms) are used as instances to initialize the same number of STRtracks. A new frame first goes through the same segmentation procedure as the first frame. Then for each region in the new frame, it's assigned to one of the track based on spatial proximity and appearance similarity. Specifically, the distance between an STRtrack and a new region is computed as the Euclidean distance between the centroid coordinates of the new region and that of the most recent instance of that STRtrack. The STRtracks whose distance are within a search radius are identified as the candidate matches. The appearance similarity is computed between a new region and it's candidate STRtracks. We define the appearance similarity between a region and an STRtrack as the cosine value of the angle between the color histogram of the region and the average color histogram of all instances of that STRtrack.

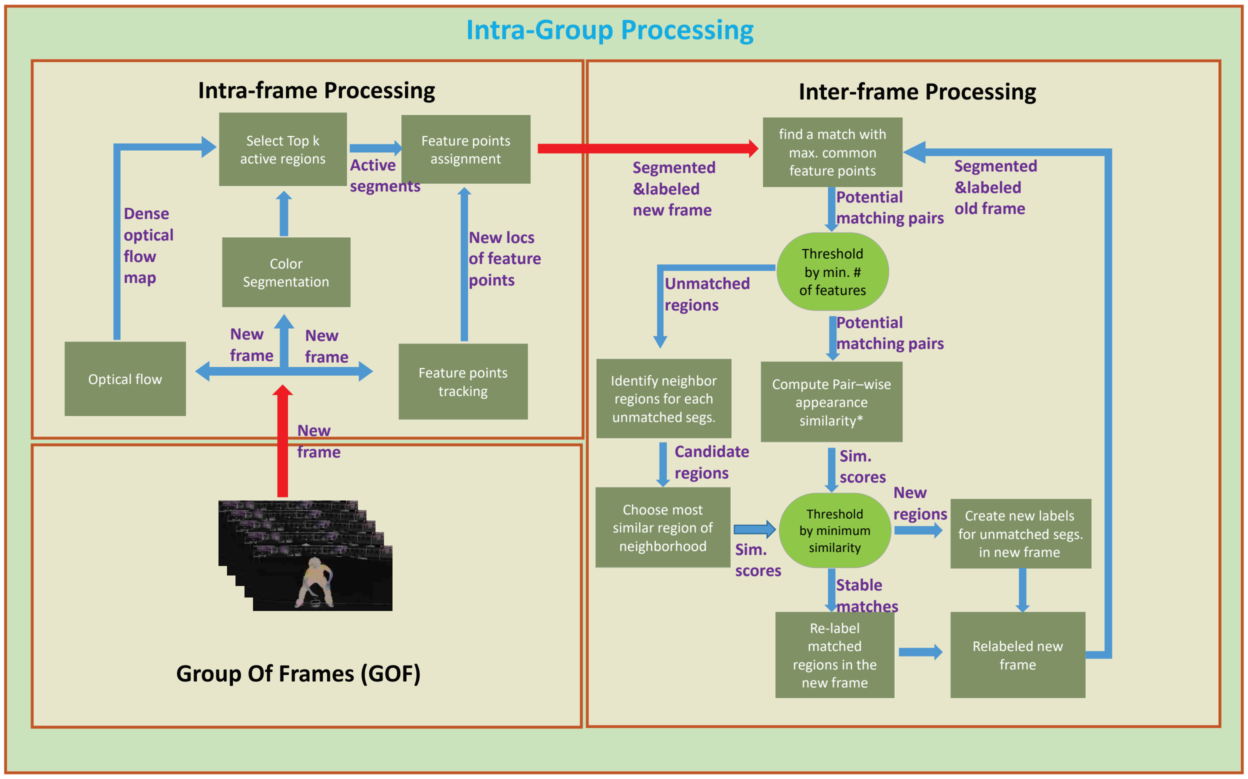

Intra-Group Processing
Intra-frame Processing
Select Top k active regions
Active segments
Feature points assignment
Dense optical flow map
Color Segmentation
New locs of feature points
New frame
New frame
Optical flow
Feature points tracking
New frame
Group Of Frames (GOF)
Inter-frame Processing
Segmented &labeled new frame
find a match with max. common feature points
Segmented &labeled old frame
Potential matching pairs
Threshold by min. # of features
Unmatched regions
Potential matching pairs
Identify neighbor regions for each unmatched segs.
Compute Pair–wise appearance similarity*
Candidate regions
Sim. scores
Choose most similar region of neighborhood
Sim. scores
Threshold by minimum similarity
New regions
Create new labels for unmatched segs. in new frame
Stable matches
Re-label matched regions in the new frame
Relabeled new frame

Figure 3.2: Block diagram of the proposed region tracking algorithm. The algorithm processes video in a streaming manner starting from the first frame. Optical flow and the color segmentation are performed in parallel for each new coming frame. Small regions of color segmentation are then clustered using optical flow information. The initial set of spatial-temporal region tracks are created using clustered regions in the first frame. Regions in later frames are compared with existing region tracks and are added to the right track based on location and appearance. At the same time, regions in the new frame are relabeled in accordance with their instances in previous frames.

It's reasonable to use the average color histogram because the same object typically doesn't change too much in appearance. And the average color histogram is more robust to outliers than the color histogram of a single instance (e.g. the most recent instance). Finally, if the similarity value of the most similar STRtrack is greater than a threshold, we simply assign the new region to that STRtrack (i.e. relabel the region using the STRtrack ID). Otherwise, we create a new STRtrack and add to it the new region as the first instance.

The result of the above region tracking algorithm is a number of STRtracks each representing the temporal evolution of an object. We then extract the motion descriptor for an STRtrack ST_i as its average acceleration. Specifically, the acceleration of a pixel (x, y) at frame t is defined as $\mathbf{g}(x, y, t) = \mathbf{F}^+(x, y, t) + \mathbf{F}^-(x, y, t)$, where $\mathbf{F}^+(x, y, t)$ and $\mathbf{F}^-(x, y, t)$ is the same as defined before. The motion descriptor of ST_i at time t is therefore

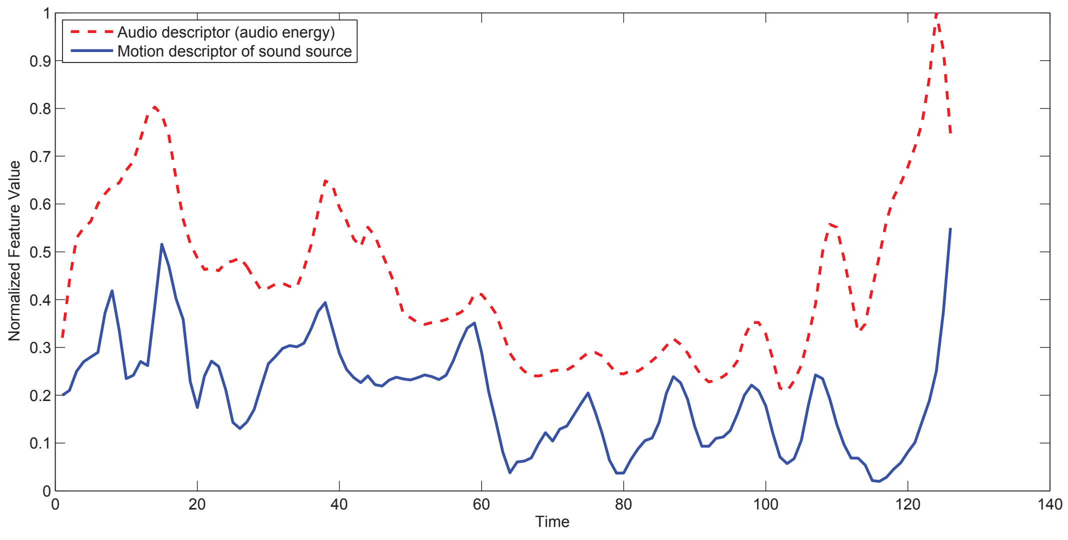
$$m_i(t) = \frac{1}{|ST_i^t|} \sum_{(x,y) \in ST_i^t} ||\mathbf{g}(x, y, t)||.$$

where $|ST_i^t|$ is the number of pixels of ST_i in the t^{th} frame. Similar to the audio descriptor, the motion descriptor $m_i(t)$ is smoothed using a Gaussian kernel to reduce the effect of visual noises. In addition, we pick the the top 15 $m_i(t)$s in terms of standard deviation as the candidates in order to filter out objects whose motion is random or minimal.

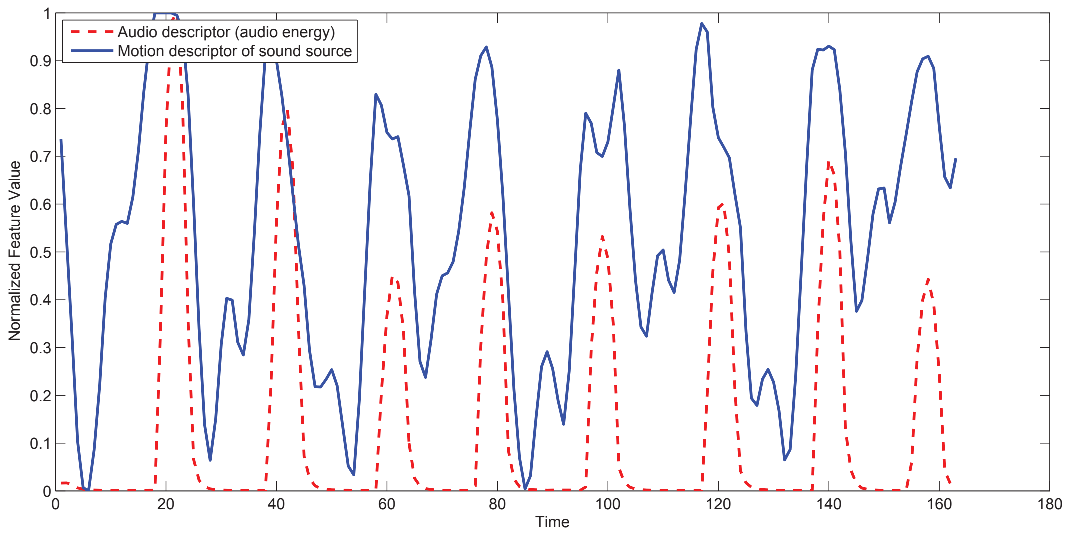
3.4 Audiovisual Correlation

Audiovisual correlation is analyzed between audio and visual descriptors obtained in the previous section. We have used the temporal functions $a(t)$ and $m_i(t)s$ to represent the audio and visual descriptors. Another way to look at those functions is to consider them as a vector in the T-Dimensional feature space, where T is the number of frames. Let $\mathbf{a}$ and $\mathbf{m_i}$ represent those feature vectors, they are formally defined as $\mathbf{a} = [a(1), a(2), \cdots, a(T)]^T$ and $\mathbf{m_i} = [m_i(1), m_i(2), \cdots, m_i(T)]^T, i = 1, 2, \cdots, k$.

Now as both modality are in a T-D vector space, an intuitive way to measure the synchrony χ_i between object i with the audio signal is to use Euclidean distance between the two corresponding feature points. However, this does not work because the precise values of variables captured by auditory and visual sensors carry essentially different semantic meanings. In other words, a larger distance does not necessarily indicate that the two signals are less synchronized than otherwise.



(a) Continuous Sound



(b) Discrete Sound

Figure 3.3: Examples of audio descriptors and the motion descriptors of the sound source. (a) is the intermediate results of the video *Violin Yanni*. The visual descriptor represents the motion of the violin player's left hand. (b) is the intermediate result of the video *Basketball*, in which the visual descriptor describes the motion of the basketball.

Correlation measures such as cosine distance does not work well either, especially for discrete sounds [1] generated by continuous motion (i.e. a bouncing basketball).

Fig.3.3 shows two examples of the audio descriptor of a sound and the motion descriptor of that sound source. As can be observed from the examples, the two descriptors do exhibit similar changing patterns (i.e. similar relative ordering of feature values at different time instant). However, the differences are even more if we look at the precise feature values, which explains why common metrics based on the precise feature values fail to capture the correlation.

We address the above problem by performing a non-linear transformation of the original feature into the ordinal space, using the Winner-Take-All (WTA) technique. Briefly, WTA hash is a ordinal space embedding technique that captures the partial ordering statistics of the original feature dimensions. Such ordinal space embedding techniques have shown success in other tasks such as dimension reduction [71], feature extraction [24] etc. However, they have not been explored in the area of multimodal information analysis. We extract the partial ordering information encoded by ordinal space embedding in an effort to bridge the semantic gap between features in audio and visual modality.

3.4.1 Winner-Take-All Hash

The WTA hash is a subfamily of hashing functions introduced in [93]. WTA hash is controlled by two parameters: the number of random permutations N and window size S. Fig. 3.4 illustrates the computation of WTA hash for a 5-dimensional feature vector with parameters $N = 3$ and $S = 3$. The index of the maximum entry among the first S entries of a permutation Θ is chosen as the hash code for that permutation. In our example, the first permutation $[2, 1, 5, 4, 3]$ reorder the feature vector as $[0.2, 0.5, 1.0, 0.30.7]$. The index of the maximum value among the first 2 en-

[1]Audio signals exhibiting clear intervals of silence is referred to as discrete sounds. An example is the sound produced by a bouncing basketball. While continuous sounds refer to audio signals without clear intervals of silence, such as that generated by violin.

tries $[0.2, 0.5]$, is taken as the encoding for that permutation. Note that N permutations generate a length-N hash code and each code entry is an integer value between 0 and $S - 1$.

Each WTA hash function encodes a partial ordering statistics of the feature dimensions and defines an ordinal embedding of the features in the rank correlation space. An intuitive way to understand such encoding is to consider the case when $S = 2$. Choosing the first 2 entries from a random permutation is essentially same as randomly choosing 2 entries from the feature vector. If the first entry is larger, the output code is 0, and it's 1 otherwise. When N is sufficiently large, the binary code for the feature vector encodes a complete pairwise ordering of different feature dimensions. Note that for a d-dimensional feature vector, the complete pairwise ordering can be encoded with a binary code of length $A_d^2 = d(d-1)$. Larger values of S also encodes the pairwise ordering, however with more emphasis on paring with the head of the S-sized subset of the feature dimensions.

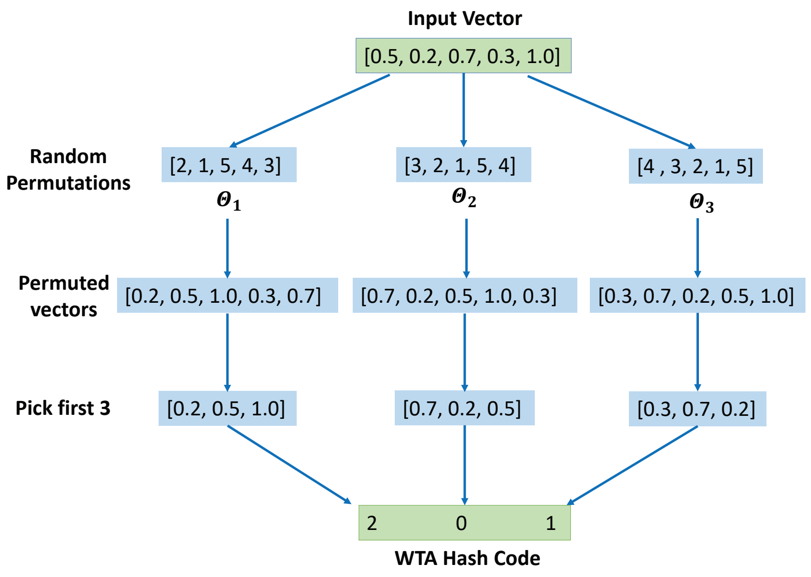


Figure 3.4: An example of WTA hash with 5-dimensional input vector X, $N = 3$ and $S = 3$. Θ_i is a permutation of the input vector X, and $X(\Theta_i)$ is the result of the permutation. $X(\Theta_i)$ is further windowed, resulting in a vector containing only its first S entries. Index of the maximum entry in the windowed $X(\Theta_i)$ is output as the WTA hash code for the given permutation.

In general, if two vectors are close in their original feature space, they must be close in the rank correlation space. However, the opposite is not true because differences in the numerical values that do not affect the ordering might have bigger effect for other metric spaces (e.g. Euclidean). This characteristic of rank correlation naturally provides certain degree of stability in face of perturbations in numerical values.

Another advantage of WTA hash that merits discussion is that compared with a complete ordering, multiple partial orderings offers another degree of resilience to noise and gives considerations to local support (with $S \geq 2$) on feature dimensions. Those advantages of WTA hash makes it perfect for measuring the synchrony between signals in different modalities, such as audio and visual in our case. Synchronized signals in audio and visual modalities demonstrate similar temporal changing patterns, which are precisely the orderings of feature dimensions when we represent the signals as vectors. Therefore, we use the same WTA hash function to encode audio and visual signals for similarity computation.

3.4.2 Correlation Analysis

Before measuring the synchrony between audio and visual signals, we first use WTA hash to encode the partial ordering statistics of audio descriptor $\mathbf{a}$ and visual descriptors $\mathbf{m_i}$. Note that same set of random permutations are used for both audio and visual descriptors. This encoding transforms features in two different modalities into the common rank correlation space for correlation analysis. Then the synchrony $\chi(\mathbf{a}, \mathbf{m_i})$ between a visual descriptor $\mathbf{m_i}$ and the audio, denoted as χ_i is simply

$$\chi_i = \frac{H_d(\mathbf{a}, \mathbf{m_i})}{Dim(\mathbf{x})},$$

where $H_d(\mathbf{a}, \mathbf{m_i})$ is the Hamming distance between them (i.e. the number of different entries). Finally, the object that corresponds to the sound source can be determined as

$$s = \arg\max_i \chi_i.$$

After identifying the sounding object, we create a confidence map and set the localization confidence for pixels belonging to this object in every frame to 1, while the rest to 0. Similar to [34], instead of dealing with binary localization confidence, the confidence map is convoluted with a Gaussian kernel in both spatial and temporal domain to generate a smooth confidence surface. However, rather than fixing the standard deviation of Gaussian kernel for all videos, we make it a function of the average velocity and object size. This is reasonable as objects with large motion has less temporal locality coherency, and the temporal Gaussian kernel should therefore use smaller standard deviation. Formally, let fr be the frame rate of a video, ma be the average acceleration of the sounding object, the standard deviation of the Gaussian kernel is computed as $\sigma_t = \rho_t \cdot fr/ma$, where ρ_t is a coefficient and we set it to 0.5 in the experiments. Similarly, the spatial Gaussian kernel size should be related with the size of an object, and we define it to be $\sigma_s = \rho_s \cdot sqrt(A)$, where A is the average area of the object and ρ_s is an coefficient set to 0.2 in our experiments.

3.5 Experiments

In this section, we show performances of the proposed algorithm on a number of challenging test videos. Specifically, we compared with the state-of-the-art method proposed in [34] both qualitatively and quantitatively.

We gather challenging test videos from previous research and Youtube. In detail, *Violin Yanni* is used in [34]. The audience clapping in *Violin Yanni* adds much noise to the audio; *Wooden Horse* is used in [17], [34] and [39]. The swaying wooden horse in this video is uncorrelated to the audio; *Guitar Lessons*, *Student News* and *Guitar Street* are downloaded from the Youtube website.

Table 3.1: Parameter settings of the proposed method.

Algorithm	**Parameter**	**Value**
Mean shift	Spatial bandwidth	13.0
	Range bandwidth	13.0
	Minimum area	400
Optical flow	Regularization weight	0.012
	Downsample ratio	0.75
	Width of the coarsest level	40
Frame segmentation	Maximum no. of segments	25
Region tracking	Searching radius	55
AV correlation	No. of permutations	2000
	Window size	5

In *Guitar Lessons*, the unintentional movement of the player head and body are all somewhat harmonious with the sound; *Guitar Street* contains lots of background noise and moving passengers in the street; In *Student News*, the left reporter's movement occasionally synchronizes with the right one's speech. In addition, we also created an additional test video *Basketball* with a distracting moving cushion, which is challenging as the motion of the sounding objects (i.e. basketball) is not constrained within a small range.

The proposed framework takes only a few parameters other than those required by Mean shift [21] and optical flow [55] component algorithms. The primary parameters used in the experiments are summarized in Table 3.1 unless otherwise specified. We also implemented the algorithm proposed in [34] for comparison. The three parameters for the QuickShift algorithm used in [34], namely, the trade-off between the color and spatial importance, the kernel size for density estimation and the maximum within-cluster distance is set to 0.25, 0.4, and 10 respectively. The number of top visual clusters in terms of velocity and acceleration standard deviation are both set to 5. The spatial-temporal smoothing of localization confidence uses the same settings as our algorithms for a fair comparison.

3.5.1 Qualitative Performance

We show the sample frames of localization results produced by the proposed method in Fig. 3.5. The ground truth and results of the algorithm in [34] are also shown for a visual comparison. We produce the ground truth data by manually labeling the most correlated moving object in each video frame using interactive segmentation. As can be observed from the results, our algorithm successfully identified the sound sources in all of the test videos, with good localization results.

Several factors contribute to the superior performance of our algorithm. First, our region tracking algorithm effectively identify the same object across all video frames, which lays the foundation for motion pattern analysis. Second, smoothing both the audio and visual descriptors in temporal domain before performing correlation analysis greatly reduces the feature noise and well captures the temporal changing pattern in both modalities. Third, the WTA hash technique we use provides further robustness to noise through rank correlation space encoding. The algorithm in [34] is able to roughly identify regions most correlated to the audio in most videos. However, the localization is less accurate than the proposed method. This is largely due to the fact that regions in the visual clusters produced by spatial-temporal clustering of small segments in [34] tend to under-segment the audiovisual object. In [34], small segments within a certain neighborhood around the sounding object are very likely to be grouped into the same cluster, resulting in a very coarse estimation of the most correlated object. We also note that, [34] fails to focus on the right sound source in *Guitar Street* and the first half of *Wooden Horse* as a result of dominance of distracting motion.

3.5.2 Quantitative Performance

We quantify the performance of the proposed sound source localization method using similar metrics as in [34]. Specifically, $precision$ and $recall$ are used to assess the spatial localization.

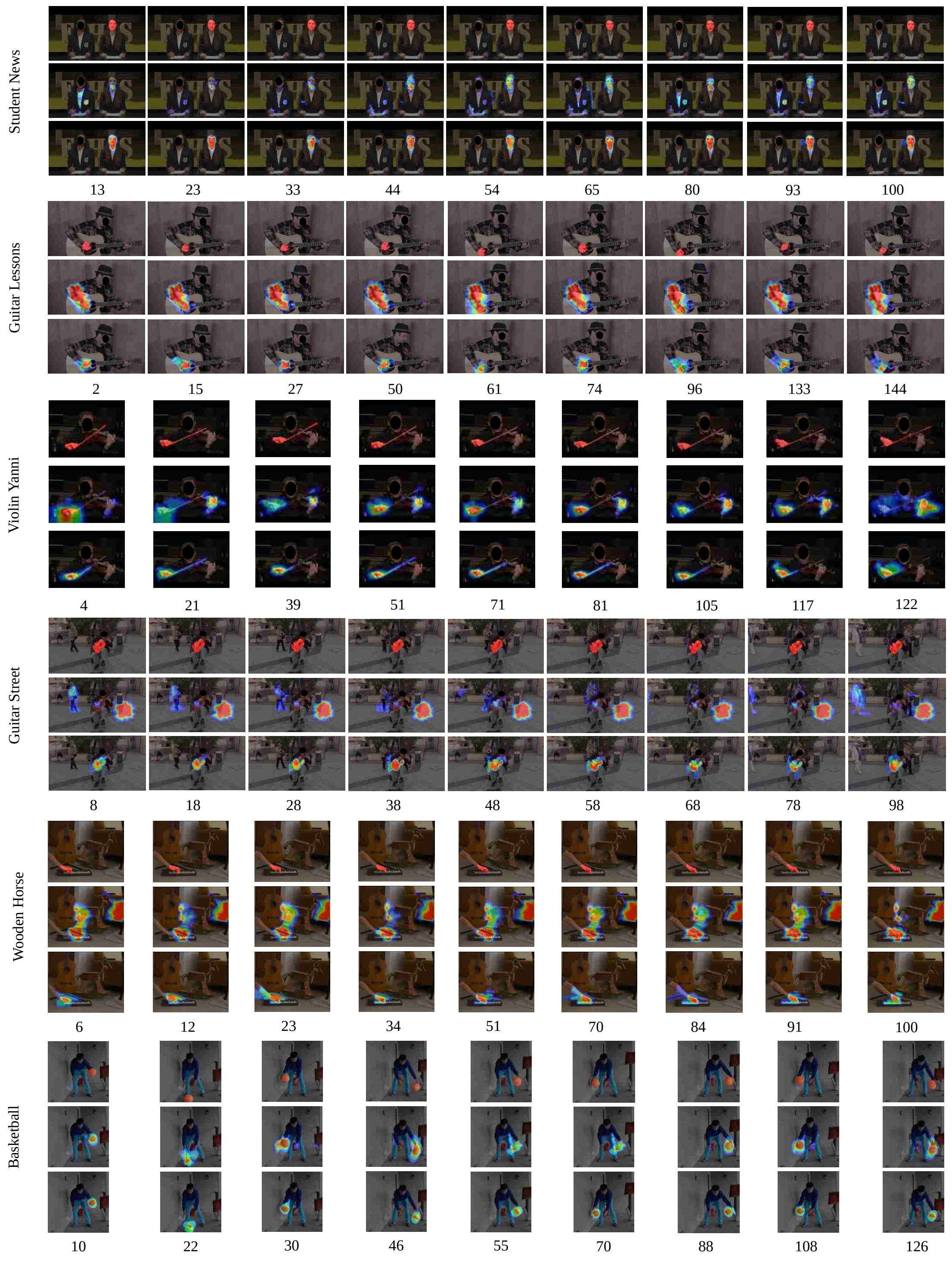
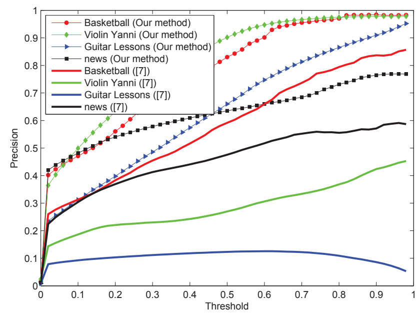


Figure 3.5: Sample frames of the audiovisual source localization and segmentation results. For each test video, the top row is the manually labeled ground truth.The second row is the results of the method proposed by [34]. The third row the results of our method. The frame number of each column is marked at the bottom of the sample frames.

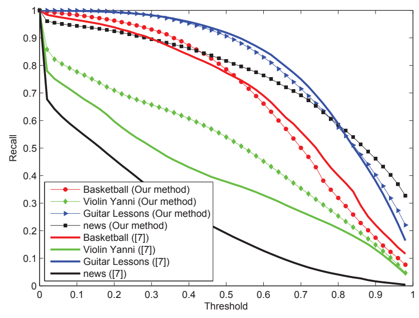
In detail, if the set of detected pixels is denoted as P, and the set of pixels in ground truth as T, then $precision$ and $recall$ are defined as $precision = \frac{|P \bigcap T|}{|P|}$ and $recall = \frac{|P \bigcap T|}{|T|}$, where $|\cdot|$ is the cardinality operation. The detected region (i.e. pixel set) P is controlled by a threshold applied to the smoothed localization confidence. The threshold is varied from 0 to 1 to get the precision-threshold and recall-threshold curve for each frame. Then those curves are averaged over all frames. In addition, the precision-recall curve is also plotted to show how $precision$ and $recall$ change against each other. Those results are shown in Fig. 3.6. Note that we only showed the quantitative results on videos that both methods have correctly localized the audio source in most of the frames. Therefore, *Guitar Street* and *Wooden Horse* are not included in the quantitative results.

Obviously, the proposed method is able to achieve higher precision consistently across all test videos under different threshold settings. The primary reason for the good precision of our method is that the proposed frame segmentation and region tracking algorithm can accurately segment the objects and reliably track the same object across the entire video. We also note that the recall performance gap between the two methods is much smaller compared to that of precision. In videos such as *Guitar Lessons* and *Basketball*, the recall of [34] is almost on par with our method. This is a natural effect of under-segmentation in [34], which sacrifices precision for recall. Our method, however, is able to achieve good results for both metrics. This is further consolidated by the precision-recall curve shown in Fig. 3.6c, where larger area under the curves means better performance.

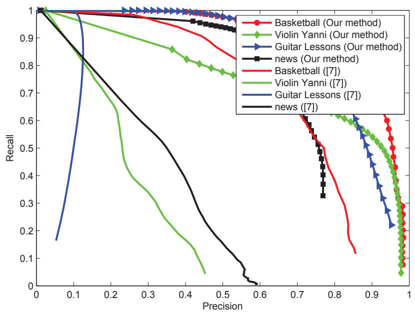
For temporal localization performance, we use the detection rate and hit ratio as the evaluation metric. Detection rate is defined as the ratio of number of frames in which the object is successfully detected to the total number of frames. A successful detection is characterized by a localization containing more than half of pixels of the actual audiovisual object (i.e. recall is greater than 0.5). Hit ratio is the ratio of accurately localized frames to the total number of frames. A localization is said to be accurate if more than half of the detected pixels are in groundtruth.



(a) Precision under varying threshold.



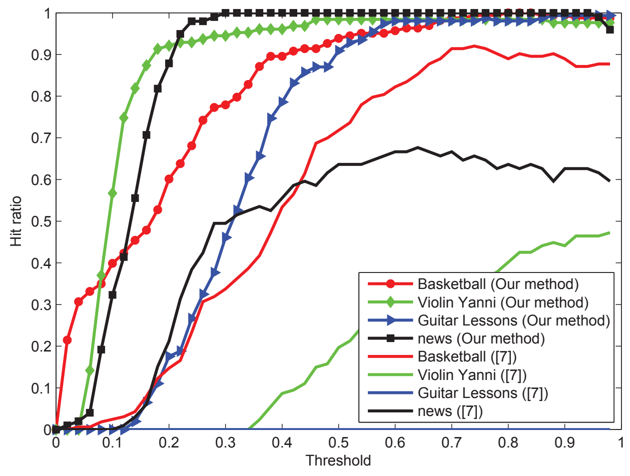
(b) Recall under varying threshold.



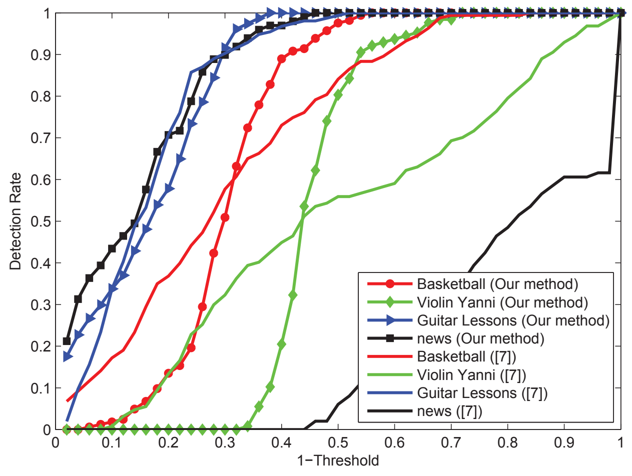
(c) Precision-Recall Curve.

Figure 3.6: Quantitative results of spatial localization performance.

Additionally, the same threshold as in spatial localization performance evaluation is used to generate a continuous curve. As shown in Fig. 3.7, the proposed method outperforms [34] in terms of hit ratio. As for detection rate, our method produces steep curves, rapidly increasing to the best performance above a certain threshold. In comparison, [34]'s detection rate increases slowly with changing threshold.



(a) Hit ratio under varying threshold.



(b) Detection rate under varying threshold

Figure 3.7: Quantitative results of temporal localizationperformance.

Table 3.2: Quantitative results of the proposed algorithm against that of [34]. The numbers are in percentile. “Prc”, “Hit” and “Ret” denotes precision, hit ratio and detection rate respectively. “Prc@Rec=0.5” denotes the precision when recall is set to 0.5, and the remaining notations are interpreted in a similar way. Better results are marked as bold.

Method	**Prc@Rec=0.5**	**Hit@Rec=0.5**	**Det@Rec=0.5**
	Basketball		
Compared method [34]	77.06	92.02	99.39
Our method	**95.54**	**98.87**	**100.0**
	Violin Yanni		
Compared method [34]	10.03	0.0	**100.0**
Our method	**92.42**	**98.43**	92.91
	Guitar Lessons		
Compared method [34]	77.06	92.02	99.39
Our method	**88.40**	**98.05**	**100.0**
	Student News		
Compared method [34]	34.80	6.06	0.0
Our method	**76.04**	**100.0**	**100.0**

Another view of the above results is shown in Table 3.2. It’s reasonable to look at the above performance metrics given a specific recall value. We set recall to 0.5 to evaluate the other performance metrics, as this represents a successful detection in the test. As shown in the table, our method again outperforms the state-of-the-art method. And interestingly, the performance gap difference between detection rate and hit ratio is similar to that in the precision and recall in the spatial localization performance. This is reasonable as detection rate is more closely related with recall and hit ratio is more related to precision based on the definition. [34] tends to perform worse in metrics related with recall than precision. An example video is *Violin Yanni*. Although the algorithm in [34] successfully detected the left hand of the violin player, which is the ground truth, the hit ratio is essentially zero because more than half of the localized region is false positive. However, for video *Student News* the hit ratio is better than the detection rate. This is because the localization results is a number of small scattering within the speaker’s face.

3.6 Summary

In this chapter, we address the problem of sound source localization and segmentation in general videos captured using a single microphone. The proposed approach uses a novel region tracking algorithm to represent the entire video as a number of spatial-temporal region tracks. The synchronization between each region track and the audio energy are analyzed efficiently with the Winner-Take-All hash. We have evaluated our approach with a number of challenging test videos and compared with the state-of-the-art, thus demonstrating its superior performance.

CHAPTER 4: HASHING FOR SINGLE-MEDIA RETRIEVAL

In this chapter, we study supervised hashing approaches for single-media retrieval. Specifically, we present a generalization of the random hashing approach that has been used to model audio-visual correlation in the previous chapter. Based on this generalization, we propose an efficient supervised learning algorithm to find optimal hash functions that can generate compact hash codes for semantic similarity search.

4.1 Overview

Existing hashing algorithms mostly learn binary codes by quantizing numeric projections [67, 59, 58] of high-dimensional features. In contrast, hashing schemes based on feature's ranking order (i.e. comparisons) are relatively under-researched. Ranking-based hashing, such as Winner-Take-All (WTA) [94] and Min-wise Hashing (MinHash) [9], ranks the random permutation of input features and uses the index of maximal/minimal feature dimensions to encode a compact representation of the input features. The benefit of ranking-based hashing lies in the fact that these algorithms are insensitive to the magnitude of features, and thus are more robust against many types of random noises universal in real applications [74, 94]. In addition, the magnitude-independence also makes the resultant hash codes scale-invariant, which is critical to compare and align the features from heterogeneous spaces, e.g., revealing the multi-modal correlations [50, 49].

Unfortunately, the existing ranking-based hashing is data-agnostic. In other words, the obtained hash codes are not learned by exploring the intrinsic structure of data distribution, making it suboptimal in its efficiency of coding the input features with compact codes of minimal length. For example, WTA encodes the data with the indices of the maximum dimensions chosen from a number of random permutations of input features. Although WTA has generated leading performances in many tasks [94, 50], it is constrained in the sense that it only ranks the existing features of input data, while incapable of combining multiple features to generate new feature subspaces to rank. A direct consequence of such limitation is that this sort of ranking-based hashing usually needs a very large number of permutations and rankings to generate useful codes, especially with a high dimensional input feature space [94].

To address this challenge, we abandon the use of ranking random permutations of existing features in ranking-based hashing algorithms. Instead, we propose to generate compact ranking-based hashing codes by learning a set of new subspaces and ranking the newly projected features in these subspaces. At each step, an input data is encoded by the index of the maximal value over the projected points onto these subspaces. The subspace projections are jointly optimized to generate the ranking indices that are most discriminative to the metric structure and/or the data labels. Then a vector of codes are iteratively generated to represent the input data from the maximal indices over a sequence of sets of subspaces.

This method generalizes ranking-based hashing from restricted random permutations to perform encoding by ranking a set of arbitrary subspaces learned by mixing multiple original features. This greatly extends its flexibility so that much shorter bits can be generated to encode input data, while retaining the benefits of noise insensitivity and scale invariance inherent in this type of algorithms. Part of this work have previously been published by the author in conference proceedings [47] and in journals [48].

The rest of this chapter is organized as follows. Section 4.2 formulates the optimization problem and describes the learning algorithm. Section 4.3 discusses the extension of the proposed algorithm to the kernel space. Experimental results and analysis are presented in Section 4.4, followed by the summary in Section 4.5.

4.2 Supervised Ranking Hash

4.2.1 The Limitations of WTA Hashing

As introduced in Section 3.4.1 of Chapter 3, WTA is specified by two parameters: the number of random permutations L and the window size K. Each permutation $\boldsymbol{\pi}$ reorders the elements of an input vector $\mathbf{x} \in \mathbb{R}^d$ to $\mathbf{x}_\pi$ in the order specified by $\boldsymbol{\pi}$. Then the index of the maximum dimension of the feature among the first K elements of $\mathbf{x}_\pi$ is used as the hash code. The limitations of WTA are twofold: (1) the entries of the input vectors are permuted in a random fashion before the comparison is applied to find the largest entry out of the first K ones; (2) the comparison and the ranking are restricted to be made between the original features. The random permutations are very inefficient to find the most discriminative entries to compare the similarity between the input vectors, and the restriction of the ranking to original features is too strong to generate the compact representations. We relax these two limitations in the development of the supervised hashing algorithm.

4.2.2 Problem Formulation

Rather than randomly permuting the input data vector $\mathbf{x}$, we project it onto a set of K one-dimensional subspaces. Then the input vector is encoded by the index of the subspace that generates the largest projected value. In other words, we have

$$h(\mathbf{x};\mathbf{W}) = \arg\max_{1\le k\le K} \mathbf{w}_k^T\mathbf{x}, \tag{4.1}$$

where $\mathbf{w}_k \in \mathbb{R}^d, 1 \le k \le K$ are vectors specifying the subspace projections, and $\mathbf{W} = [\mathbf{w}_1\mathbf{w}_2\cdots\mathbf{w}_K]^T$.

We use a linear projection to map an input vector into subspaces to form its hash code. At first glance, this idea is similar to the family of learning-based hashing algorithms based on linear projection [23, 67]. However, different from these existing algorithms, the proposed method instead ranks the obtained subspaces to encode each input vector with the index of the dimension with the maximum value. This makes the obtained hash codes highly nonlinear to the input vector, invariant to the scaling of the vector, as well as insensitive to the input noises to a larger degree than the linear hashing codes. We name this method Supervised Ranking Hash (SRH) to distinguish it from the other compared methods.

WTA can be seen as a special case of the SRH algorithm by restricting the projections onto K axis-aligned linear subspaces, i.e., $\mathbf{w}_k$ is set to a column vector $\mathbf{e}_k$ randomly chosen from an identity matrix $\mathbf{I}$ of size $d \times d$. SRH extends WTA by relaxing the axis aligned linear subspaces in (4.1) to arbitrary K-dimensional linear subspaces in $\mathbb{R}^d$. Such relaxation greatly increases the flexibility to learn a set of subspaces to optimize the hash codes resulting from the projections to these subspaces.

Now our objective boils down to learn hash functions characterized by the projections $\mathbf{W}$ as in Equation (4.1). Specifically, let $\mathcal{D}$ be the set of N d-dimensional data points $\{\mathbf{x}_i\}_{i=1}^N$ and let $\mathbf{S} = \{s_{ij}\}_{i,j}^N$ be the set of pair-wise similarity matrix satisfying $s_{ij} \in \{0, 1\}$, where $s_{ij} = 1$ means the pair $(\mathbf{x}_i, \mathbf{x}_j)$ is similar and vice verse. The pair-wise similarity matrix $\mathbf{S}$ can be obtained either from the nearest neighbors in a metric space or by semantic labels.

Given a similarity label s_{ij} for each training pair, we can define an error incurred by a hash function like (4.1) below

$$e(h_i, h_j, s_{ij}) = \begin{cases} \mathrm{I}(h_i \neq h_j), & s_{ij} = 1 \\ \lambda\,\mathrm{I}(h_i = h_j), & s_{ij} = 0 \end{cases} \tag{4.2}$$

where $\mathrm{I}(\cdot)$ is the indicator function outputting 1 when the condition holds and 0 otherwise, $h_{i(j)}$ is $h(\mathbf{x}_{i(j)}; \mathbf{W})$ for short, and λ is a hyper-parameter that controls the relative penalty of false positive with respect to false negative.

The learning objective is to find $\mathbf{W}$ to minimize the cumulative error over the training set:

$$E(\mathbf{W}) = \sum_{s_{ij} \in \mathbf{S}} e(h_i, h_j, s_{ij}) \tag{4.3}$$

Note that $\mathbf{W}$ factors into the above objective function because both h_i and h_j are functions of $\mathbf{W}$.

4.2.3 *Optimization*

4.2.3.1 Reformulation

Without loss of generality, the hash function in (4.1) is equivalent to the following formulation:

$$\begin{aligned} &\mathbf{h}(\mathbf{x}; \mathbf{W}) = \arg\max_{\mathbf{g}} \mathbf{g}^T \mathbf{W} \mathbf{x}, \\ &\text{subject to } \ \mathbf{g} \in \{0,1\}^K, \mathbf{1}^T \mathbf{g} = 1, \end{aligned} \tag{4.4}$$

which outputs an 1-of-K binary code $\mathbf{h}$ for an input feature vector $\mathbf{x}$. The constraints enforce that there must exist one and only one nonzero entry of 1 in the resultant hash code. It is easy to find the equivalence to the hashing function (4.1): the only nonzero element in $\mathbf{h}$ encodes the index of dimension with the maximum value in $\mathbf{W}\mathbf{x}$.

Following (4.4), (4.3) can also be rewritten accordingly, giving rise to the objective function in a matrix form,

$$\begin{aligned} E(\mathbf{W}) &= \sum_{s_{ij}=1} (1 - \mathbf{g}_i^T \mathbf{g}_j) + \sum_{s_{ij}=0} \lambda \mathbf{g}_i^T \mathbf{g}_j \\ &= \sum_{s_{ij} \in \mathbf{S}} (\lambda - (\lambda + 1) s_{ij}) \mathbf{g}_i^T \mathbf{g}_j + const. \\ &= \text{trace}\left(\mathbf{G}\mathbf{D}\mathbf{G}^T\right) + const. \end{aligned} \tag{4.5}$$

where $\mathbf{G} = [\mathbf{g}_1, \mathbf{g}_2, \cdots, \mathbf{g}_N]$ is the $K \times N$ hash code matrix with the constraints in (4.4) enforced for each column and $\mathbf{D}$ is a $N \times N$ matrix whose entries are defined as $d_{ij} = \lambda - (\lambda + 1) s_{ij}$.

4.2.3.2 Continuous Approximation

The objective function in (4.5) is straightforward to formulate, but hard to optimize because $\mathbf{G}$ is non-convex and highly discontinuous with respect to $\mathbf{W}$ due to the $\arg\max$ operations. Therefore, we seek a continuous relaxation of the original problem and solve the relaxed problem instead.

Specifically, Eq. (4.4) can be approximated with the softmax function,

$$\mathbf{h}(\mathbf{x}; \mathbf{W}) \approx \sigma(\mathbf{W}\mathbf{x}), \tag{4.6}$$

where $\sigma(\mathbf{z})$ is a K-dimensional softmax vector defined as

$$\sigma(\mathbf{z})_j = \frac{e^{\alpha \mathbf{z}_j}}{\sum_{k=1}^{K} e^{\alpha \mathbf{z}_k}} \quad \text{for} \quad j = 1, \cdots, K, \tag{4.7}$$

where α controls the smoothness of the approximation and z_j denotes the j^{th} entry of $\mathbf{z}$. The above relaxation is intuitive from a probabilistic perspective: the k^{th} entry of the softmax vector can be seen as the probability of the k^{th} dimension being the maximum. When $\alpha \to \infty$, the output of (4.6) converges to that of (4.4).

4.2.3.3 The Learning algorithm

With the softmax approximation, the objective function in (4.5) becomes a continuous function of $\mathbf{W}$. However, the non-convex nature of the problem remains unchanged. One can compute the negative gradient and use the standard gradient descent algorithms (e.g. L-BFGS) to find its local minima. But computing the gradient over the full training set is computationally expensive, and the sum of error form of the objective function indicates that this problem can be solved with the stochastic gradient descent (SGD) algorithm which uses a single pair or a mini-batch to approximate the expected gradient in an iterative learning procedure.

In practice, we use mini-batch based updates to reduce gradient variance and obtain more stable convergence. The batch update rules can be written in the following matrix form,

$$\mathbf{W} \leftarrow \mathbf{W} + \eta[\mathbf{G}\,\mathrm{diag}(\mathbf{G}_s^T\mathbf{G}) - \mathbf{G}_s \circ \mathbf{G}]\mathbf{X}^T, \tag{4.8}$$

where the operator $\mathrm{diag}(\cdot)$ outputs a diagonal matrix by only retaining the diagonal entries of the input square matrix, $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N]$ is the $d \times N$ training data matrix, $\mathbf{G} = [\mathbf{g}_1, \cdots, \mathbf{g}]$ is a $K \times N$ matrix containing the softmax vectors of each training sample, and $\mathbf{G}_s = \mathbf{GD}$.

Note that the softmax approximation is only used to learn the optimal K-dimensional ranking subspace $\mathbf{W}$. While in the hash encoding step, we still use (4.1) to output the hash code which can take a value between 0 and $K - 1$.

The above procedure for learning one hash function is illustrated in Algorithm 1. To generate L hash codes, we use the standard Adaboost algorithm to sequentially learn L hash functions. Similar boosting procedures have also been applied to previous works [52, 53]. Since boosting is not the contribution of this work, we therefore do not elaborate on it. Interested reader can refer to [8] for more details on this standard ensemble learning method.

4.3 Kernel Space Extension

It has been verified in various studies [58, 77] that kernel-based hash functions are more effective in capturing complex nonlinear structures of high-dimensional features. The simple structure of the ranking-based hash functions makes it easy to extend the ranking operations to kernel spaces. Specifically, instead of ranking the feature embedding in the linear subspaces, we consider the kernel space embedding

$$F(\mathbf{x}) = \mathbf{W}\phi(\mathbf{x}).$$

Algorithm 1: Supervised Ranking Hash

```
1: Input: data $\mathbf{X} \in \mathbb{R}^{d\times N}$, pairwise similarity matrix $\mathbf{S} \in \mathbb{R}^{N\times N}$, length of hash code $L$,
   subspace dimension $K$
2: Initialize: set weight matrix $\mathbf{\Omega} = [\omega_{ij}]$ to all ones.
3: for $l = 1$ to $L$ do
4:   Randomly initialize $\mathbf{W}_l$ from Gaussian distribution
5:   repeat
6:     Randomly select a training batch $\mathbf{X}_b$ and obtain the batchwise similarity matrix $\mathbf{S}_b$ and
       weight matrix $\mathbf{\Omega}_b$ accordingly.
7:     Compute $\mathbf{D}_b$ and $\mathbf{G}_b$ for the batch according to the definition of $\mathbf{D}$ and $\mathbf{G}$ respectively
8:     Set $\mathbf{G}_{bs} = \mathbf{G}_b(\mathbf{D}_b \circ \mathbf{\Omega}_b)$
9:     Update projection matrix $\mathbf{W}_l$ according to
       $$\mathbf{W}_l \leftarrow \mathbf{W}_l + \eta[\mathbf{G}_b \,\mathrm{diag}(\mathbf{G}_{bs}^T\mathbf{G}_b) - \mathbf{G}_{bs} \circ \mathbf{G}_b]\mathbf{X}_b^T$$
10:  until Convergence
11:  Compute ranking hash code for all samples using (4.1) and evaluate the weighted
     cumulative error
       $$\epsilon_l = \frac{\sum_{i,j} \omega_{ij}^{(l)} e(h_i, h_j, s_{ij})}{\sum_{i,j} \omega_{ij}^{(l)}}$$
12:  Evaluate the quantity
       $$\theta_l = \ln\left\{\frac{1-\epsilon_l}{\epsilon_l}\right\}$$
13:  Update the pair weighting coefficients using
       $$\omega_{ij}^{(l+1)} = \omega_{ij}^{(l)} \exp\{\theta_l e(h_i, h_j, s_{ij})\}$$
14:  Normalize $\omega_{ij}$'s such that $\sum_{i,j} \omega_{ij}^{(l+1)} = N^2$.
15: end for
```

Here $\phi(\mathbf{x})$ defines the kernel embedding and $\mathbf{W} = [\mathbf{w}_1\mathbf{w}_2\cdots\mathbf{w}_K]^T$ is the projection onto the a K-dimensional kernel space. Generally, any suitable embedding $\phi(\mathbf{x})$ are acceptable. Here we choose the simple yet powerful RBF kernel

$$\phi(\mathbf{x}) = [\exp(-\frac{\|\mathbf{x}-\mathbf{a}_1\|^2}{2\sigma^2}), \cdots, \exp(-\frac{\|\mathbf{x}-\mathbf{a}_m\|^2}{2\sigma^2})]^T$$

where $\{\mathbf{a}_i\}_{i=1}^m$ are the anchor points randomly selected from the training set and σ controls the kernel width.

Then the input vector is encoded by the index of the maximum value in the K-dimensional kernel space. In other words, we have

$$h(\mathbf{x}; \mathbf{W}) = \arg \max_{1 \le k \le K} \mathbf{w}_k^T \phi(\mathbf{x}), \tag{4.9}$$

Note that the general solution to the ranking-based hash function learning steps still holds with the kernel extension. The only difference in the learning steps is line 9 in Algorithm 1, which should be replaced with the following update rule in the kernel space

$$\mathbf{W}_l \leftarrow \mathbf{W}_l + \eta[\mathbf{G}_b \, \mathrm{diag}(\mathbf{G}_{b_s}^T \mathbf{G}_b) - \mathbf{G}_{bs} \circ \mathbf{G}_b]\phi(X_b)^T, \tag{4.10}$$

where the embedding function $\phi(\cdot)$ applies to each column of $\mathbf{X}_b$. The kernel extension extends the discriminative capabilities ranking-based hash functions and reveals ranking structures that can not be discovered in the linear subspaces, as will be evidenced by our experimental results.

4.4 Experiments

4.4.1 Dataset

To evaluate the proposed hashing method, we use three frequently used [28, 42, 67, 59] datasets with semantic annotations: Labelme, Peekaboom and NUSWIDE. Labelme is an object image collection which consists of 22,000 images represented as 512D Gist vectors designed for vision and learning tasks. This dataset also comes with a semantic affinity matrix which is based on segmentations and object labels. Peekaboom [83] is an semantically annotated image dataset containing 60,000 images, where the objects labels in each image are obtained through involuntary game play. The images are represented by 512D Gist features and the pairwise semantic similarity scores are also included; The NUSWIDE dataset [20] is a real-world image dataset that contains approximately 270,000 images downloaded from Flicker. Every image in this dataset is labeled as one or more of 81 concepts, and represented with 500-D bag-of-visual words (BOVW) feature.

4.4.2 Baseline Methods

We compare the proposed SRH against seven state-of-the-art hashing methods including Winner-Take-All hashing (WTA) [94], Supervised Hashing with Kernels (KSH) [58], Two-step Hashing (TSH) [53], Latent Factor Hashing (LFH) [102], Supervised Discrete Hashing (SDH) [77], Column Sampling based Discrete Supervised Hashing (COSDISH) [38], and Fast Hash (FastH) [52]. Those algorithms are considered as the most competitive hashing algorithms in the literature. For all of the baseline algorithms, we have obtained the source code from the original authors and we use them directly in the experiments.

4.4.3 Evaluation Metrics

We consider the performance of approximate nearest neighbor search using the widely adopted *top-k precision*, *mean Average Precision* (mAP), and *precision-recall* which are defined as follows:

Top-k precision The top-k precision is defined as the ratio of relevant items among the retrieved top k instances in terms of Hamming distance. This metric is averaged over all queries in our evaluation.

mAP The mean average precision is defined as

$$\text{mAP} = \frac{1}{Q}\sum_{q=1}^{Q}\frac{\sum_{r=1}^{R} P_q(r)\delta_q(r)}{\sum_{r=1}\delta_q(r)}, \tag{4.11}$$

where Q is the size of the query set, $P_q(r)$ denotes the top-k precision of the q^{th} query, and $\delta_q(r)$ indicates whether the k^{th} data item is relevant to the q^{th} query.

Precision-recall Precision-recall reflects the precision values at different recall levels and it's a good indicator of the overall performance of different algorithms. Typically, the area under the precision-recall curve is computed and a larger value indicates better performance.

4.4.4 Experiment Settings

For each of the three datasets, we randomly sample a separate set of 2000 samples as test queries and use the remaining as the database. Then we randomly pick 2000 samples from the database to construct similarity matrix for training. The learned hash functions are then applied to the database and query set to obtain the database and query hash codes. Such settings follow the practice in [42, 67, 58, 53], where a small portion of the database items are used for training to test the generalization capability of different algorithms. This also simulates real-world scenarios where the semantic annotations are scarce compared to the entire corpus of available data.

We follow previous work (e.g. [67], [59]) in labeling the groudtruth neighbor of each sample in those datasets. In detail, the groundtruth neighbors in Labelme are obtained by thresholding the semantic affinity matrix to make sure each sample has an average of 100 groundtruth neighbors. As for Peekaboom, the similarity scores can be directly utilized for training purposes. In NUSWIDE, an image may contain multiple tags, and the groundtruth neighbors are determined by examining whether a pair share at least one common tag. Pairwise similarity labels are sufficient for the training of all the algorithms except SDH. SDH is based on the classification framework and can only be trained with point-wise class labels, which are not available in Labelme and Peekaboom. As a result, we couldn't put SDH in the tests on those two datasets.

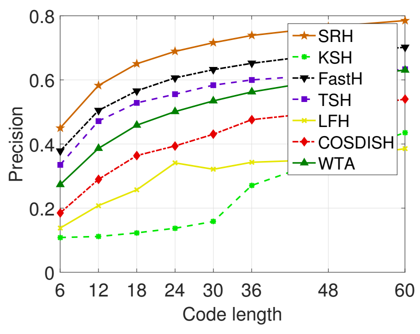
For all the baseline methods, we have used the suggested parameters provided by their authors. The proposed SRH takes a primary parameter K (i.e. subspace dimension), and a hyper-parameters λ (i.e. the penalty coefficients). They are selected by 5-fold cross-validation on a small held-out subset in the training set. Specifically, the parameter range for K and λ are $\{2, 4, 8, 16\}$ and $\{0.5, 1.0, 2.0\}$ respectively.

Table 4.1: mAP Comparison on Labelme. The results are obtained by training on 2000 samples randomly selected from the database. The learned hash functions are applied to both database and test set to generate database codes and test codes respectively. We couldn't train SDH on this dataset because there're no instance-wise labels. The proposed SRH is better than all the compared baselines at different code lengths.

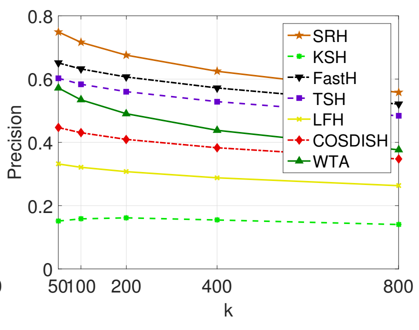
Method	6 bits	12 bits	30 bits	60 bits
WTA	0.2042	0.2679	0.3560	0.4414
KSH	0.1127	0.1134	0.1365	0.3023
LFH	0.1248	0.1855	0.2404	0.2803
TSH	0.3077	0.3782	0.4502	0.4798
COSDISH	0.1759	0.2288	0.3139	0.3906
FastH	0.3171	0.3971	0.4844	0.5361
SRH	**0.3379**	**0.4292**	**0.5370**	**0.6112**

Since SRH generates K-ary hash codes each requiring $\log_2 K$ binary bits to encode, therefore we only learn $L_b / \log_2 K$ SRH codes when comparing with other hash methods at L_b binary bits to ensure fairness. All of the experimental results are averaged over 5 independent runs.

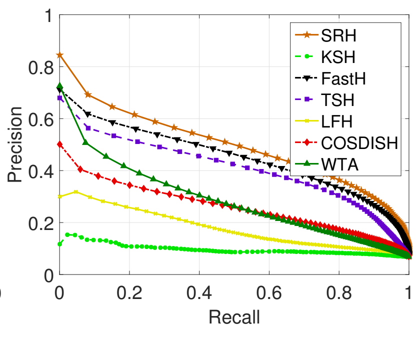
4.4.5 Results and Discussions



(a) Top 100 precision



(b) kNN accuracy



(c) Precision-recall

Figure 4.1: Test results on Labelme. 30 bits are used for (b) and (c).

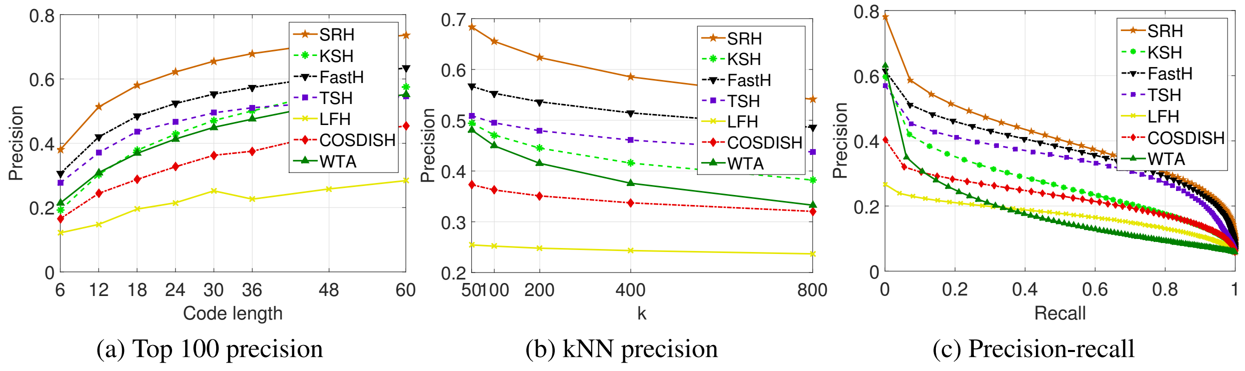


Figure 4.2: Test results on Peekaboom. 30 bits are used for (b) and (c).

4.4.5.1 Results on Labelme

The mAP of the proposed SRH and the baseline methods are shown in Table 4.1. Note that we aim to compare the performance of different methods in generating compact hash codes, therefore we consider hash codes up to 60 bits. It's easy to observe the overall leading performance of SRH across different benchmarks. Interestingly, we found that as a random ranking-based hashing algorithm, WTA performs surprisingly well in this test, even better than some competitive supervised hashing algorithms such as LFH and COSDISH. This can be explained by the fact that similarity labels obtained through semantic affinity matrix are noisy, which makes it difficult for most supervised algorithms to learn high quality binary-partitioning-based hash functions. The good performance of WTA verifies the efficacy of ranking-based hashing methods against prevalent feature noises. The proposed SRH further improves over WTA as a result of the effective ranking-based supervised learning framework. In particular, we found that the representative kernel hashing algorithm KSH performs very bad in this dataset. This is partially due to the above facts, and additionally, the powerful kernel embedding makes it more easily overfitted to the noisy training data. The superior performance of RSH demonstrates the effectiveness of the ranking-based hashing framework in handling semantic similarity search in real-world image datasets.

Table 4.2: mAP Comparison on Peekaboom. The results are obtained by training on 2000 samples randomly selected from the database. The learned hash functions are applied to both database and test set to generate database codes and test codes respectively. Results of SDH are not available on this dataset because the instance-wise class labels are not available. The proposed SRH achieves the best performance on this dataset.

Method	6 bits	12 bits	30 bits	60 bits
WTA	0.1635	0.2131	0.3069	0.3786
KSH	0.1829	0.2492	0.3513	0.4121
LFH	0.1218	0.1450	0.2282	0.2495
TSH	0.2663	0.3363	0.4162	0.4475
COSDISH	0.1597	0.2199	0.2999	0.3640
FastH	0.2885	0.3649	0.4572	0.5134
SRH	**0.3273**	**0.3991**	**0.5112**	**0.5815**

4.4.5.2 Results on Peekaboom

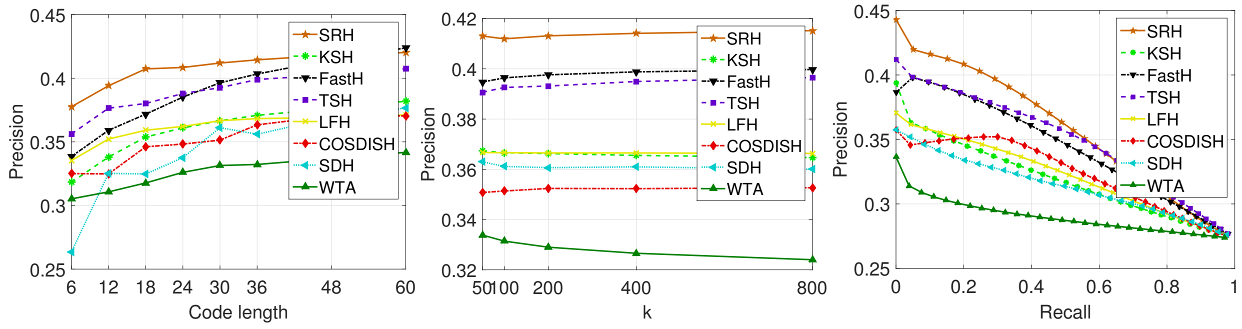
The test results on Peekaboom are shown in Table 4.2 and Figure 4.2. The relative performances of different algorithms are generally consistent with those of Labelme, with SRH leading in different metrics. Similarly, WTA continues to demonstrate competitive performance even without any training involved. Such results are reasonable because the semantic similarity scores that come with Peekaboom is obtained through involuntary game play which inevitably contain certain degree of noises similar to that in Labelme. The precision-recall curve in Figure 4.2c shows more details of the precision at different recall levels and larger area under the curve indicates better performance. Therefore, the results are in accordance with the mAP. Figure 4.2a and Figure 4.2b show the top-k precision of kNN search from different perspectives, and the results demonstrate the effectiveness of SRH in kNN search tasks.

4.4.5.3 Results on NUSWIDE

NUSWIDE is the largest datasets among the three and the full annotations based on semantic content make it the *de facto* benchmark dataset for testing semantic retrieval algorithms. The results of our tests in NUSWIDE are shown in Table 4.3 and Figure 4.3. However, different from previous datasets where the similarity labels contain certain degree of noises, the image labels here are obtained in a strictly-scrutinized and systematic way and are therefore much less noisy. As a result, the supervised hashing algorithms demonstrate significant performance gains over the random hashing algorithm. Among the supervised algorithms, SRH maintains the best overall performance and the advantages are especially noteworthy at short code length. We find that FastH is also very competitive in this test. However, as we will see the next section, the training time of FastH is significantly higher than SRH. In all, the experimental results in NUSWIDE demonstrate that SRH can also take advantage of high quality semantic labels and easily handle large-scale datasets.

Table 4.3: mAP Comparison on NUSWIDE. The results are obtained by training on 2000 samples randomly selected from the database. The learned hash functions are applied to both database and test set to generate database codes and test codes respectively. The proposed SRH continue to demonstrate competitive performances on this large-scale dataset.

Method	6 bits	12 bits	30 bits	60 bits
WTA	0.3001	0.3044	0.3212	0.3305
KSH	0.3195	0.3391	0.3646	0.3743
LFH	0.3368	0.3523	0.3665	0.3711
TSH	0.3588	0.3778	0.3995	0.4129
COSDISH	0.3274	0.3283	0.3554	0.3735
SDH	0.2734	0.3163	0.3587	0.3731
FastH	0.3424	0.3655	0.4012	0.4241
SRH	**0.3781**	**0.3957**	**0.4172**	**0.4323**

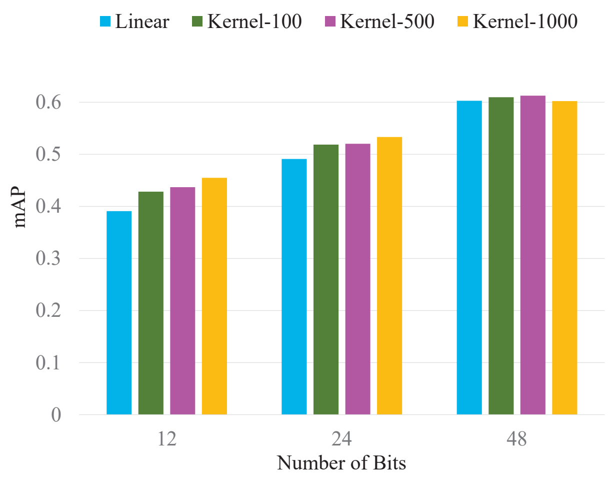


(a) Top 100 precision (b) kNN accuracy (c) Precision-recall

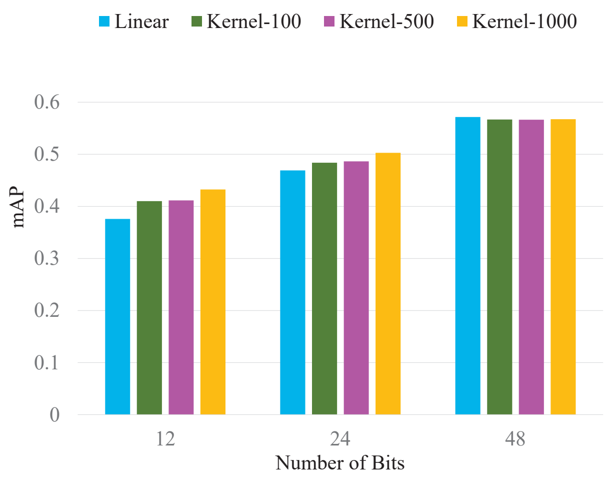
Figure 4.3: Test results on NUSWIDE. 30 bits are used for (b) and (c).

4.4.6 Evaluation of the Kernel Extension

We have evaluated the kernel extension of the proposed SRH algorithm on the Labelme and Peekaboom datasets, and the results are shown in Figure 4.4. We have used the non-linear version of SRH as the baseline, and it is shown as 'Linear' in the plot. For the kernel version, we vary the number of anchor points from 100 to 1000, and they are denoted as 'Kernel-100', 'Kernel-500' and 'Kernel-1000' respectively. As can be observed from the results, the kernel extension can further boost the performance of the proposed ranking-based hashing method, especially when the hash codes are short. In fact, the better performance of Kernelized SRH is expected because the the kernel embeddings can capture nonlinear ranking structures that can not be revealed in linear subspaces. Additionally, we find that the performances of kernelized SRH generally increase with more anchor points, which is consistent with findings in previous kernel methods [58, 77]. However, since more anchor points also result in higher computational overheads, one should find the balance between performance and computational costs in practice.



(a) Labelme

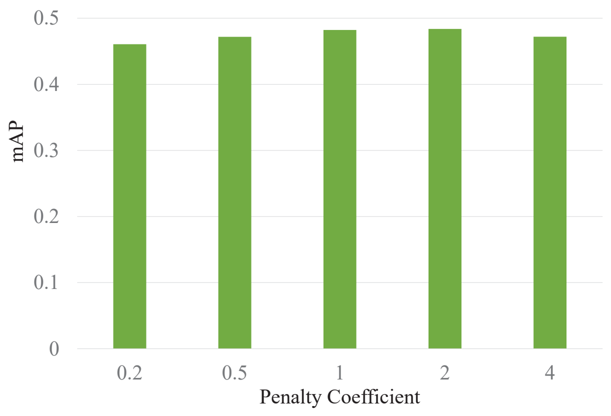


(b) Peekaboom

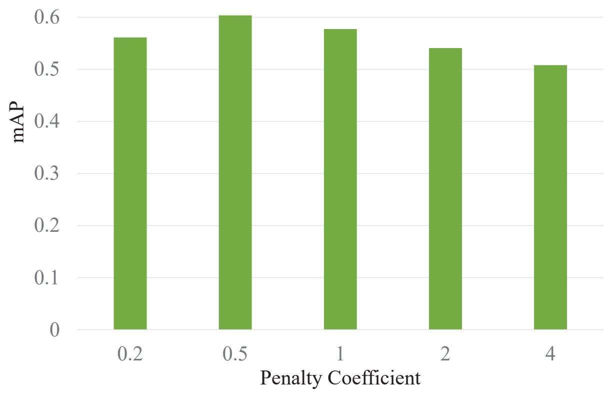
Figure 4.4: Evaluation of the kernel space extension. The subspace dimension is set to 4 for all the variants of SRH. "Linear" is the default SRH. "Kernel-100", "Kernel-500", and "Kernel-1000" represent the kernelized SRH with 100, 500 and 1000 anchors respectively. The kernel version of SRH generally performs better than the default SRH with linear subspace rankings.

4.4.7 The Effect of Penalty Coefficients

Here we study the effect of the only hyper-parameter, the penalty coefficient λ, on the performances of SRH. In this experiment, we fix the code length and subspace dimension (i.e. $L_b = 24, K = 4$), and vary λ within $\{0.2, 0.5, 1, 2, 4\}$. For different options of λ, we compute the mAP and report the results in Figure 4.5. We note that λ has different effects on the performances of SRH on different datasets. For instance, the performance of SRH on Labelme is very resilient to different options of λ; while on Peekaboom, SRH is more sensitive to different λ, showing more variations in the mAP values. In addition, the best option of λ is different on the two tested datasets, with $\lambda = 2$ and $\lambda = 0.5$ for Labelme and Peekaboom respectively. Therefore, the best practice is to use cross-validation to obtain the optimal λ.



(a) Labelme



(b) Peekaboom

Figure 4.5: Study of the hyper-parameter λ. The length of hash code fixed to 24 in this experiment. The effect of λ is different on different datasets. One should use cross-validation to choose the best λ.

4.5 Summary

In this chapter, we first present an alternative view of the WTA hashing methods used in Chapter 3. Based on this new perspective, we develop a generalized ranking-based hashing method, referred to as Supervised Ranking Hash (SRH), and study its application in semantic image retrieval. Our experiments in several datasets demonstrate the effectiveness of SRH in generating compact hash codes for semantic similarity search.

CHAPTER 5: HASHING FOR CROSS-MEDIA RETRIEVAL

This chapter presents the research on hashing approaches for the cross-media retrieval problem. Specifically, we continue to generalize the ranking-based hashing method and apply it to the cross-media scenarios, where one may search for relevant content using queries from a different media type. We will present alternative methods to solve the cross-media hash learning problem and discuss the ranking-based hashing in a more general context.

5.1 Overview

The majority of existing hashing research, including the SRH method discussed in Chapter 4, are designed for single-media data; that is, data can only be queried by an example of the same modality. In reality, however, when an user submits a query, he or she may want the system to bring back relevant content in different modalities. For example, when a "dog" image is submitted as a query, it is expected that system returns some "dog"-related media objects with different modalities, such as the sound of dog barking, the textual descriptions of dog characteristics, and the video showing dog running. On the other hand, even data of the same media type may be represented by different types of features (e.g. an image can be represented by Bag-of-Words, GIST descriptors, CNN features, etc.), and it is sometimes necessary to explore the correlations among distinct feature representations. These applications call for hashing techniques to enable similarity search using data from a different modality, which is the primary focus of cross-media hashing research.

Cross-media hashing is a challenging problem because data from different modalities typically have distinct representations with incomparable space structures and dimensionalities. Existing algorithms [112, 98, 63, 70, 65, 89] generally follow two steps: first, features from different modalities are mapped into a common feature space to minimize some cross-correlation error; second, hash codes are generated by binary partitioning of the feature space obtained through linear or nonlinear transformation of the original features. Different hashing techniques usually differ in the first step, where different error functions are defined. As for the second step, they can be similarly represented as the binary embedding: $h(\mathbf{x}; \mathbf{w}) = sign(F_{\mathbf{w}}(\mathbf{x}))$, where $\mathbf{x}$ is the input vector, $\mathbf{w}$ is the solution to the optimization problem in the first step and $F_{\mathbf{w}}(\cdot)$ is a feature transformation function parameterized by $\mathbf{w}$.

In comparison, the proposed cross-media hashing technique is based on the ranking hash framework, which exploits the relative ordering of feature dimensions. In this research, we extend the study of ranking-based hash functions to cater to cross-modal retrieval. Specifically, we learn two groups of linear subspaces jointly, one for each modality, such that the ranking ordering in one subspace is maximally aligned with that of the other.

The rest of this chapter is organized as follows. Section 5.2 formulates the cross-modal ranking subspace learning problem. Section 5.3 presents the solution to the hash function learning problem. Section 5.4 discusses how the proposed framework can easily accommodate different loss functions. Section 5.5 presents the experimental results, followed by the summary in Section 5.6.

5.2 Problem Definition

Suppose we have the data sets from two modalities $\mathcal{X}$ and $\mathcal{Y}$. Let $\mathcal{D}_{\mathcal{X}}$ be a set of $d_{\mathcal{X}}$-dimensional data points $\{\mathbf{x}_i\}_{i=1}^{N_{\mathcal{X}}}$ from modality $\mathcal{X}$ and $\mathcal{D}_{\mathcal{Y}}$ be a set of $d_{\mathcal{Y}}$-dimensional data points $\{\mathbf{y}_i\}_{i=1}^{N_{\mathcal{Y}}}$ from modality $\mathcal{Y}$. In addition, we have a set of inter-modality similarity labels $\mathbf{S} = \{s_{ij}\} \in \{1, 0\}^{N_{\mathcal{X}} \times N_{\mathcal{Y}}}$ indicating whether the cross-modal pair $(\mathbf{x}_i, \mathbf{y}_j)$ describe the same concept or not. Our objective is to learn two sets of hash functions $H_* = \{h_*^{(l)}\}_{l=1}^{L}$ with '$*$' being a place holder for $\mathcal{X}$ or $\mathcal{Y}$, so that data from both modalities can be projected into a common Hamming space.

We consider the same type of hash function as defined in (4.1). For notation convenience, we rewrite the ranking-based hash function for two modalities as follows

$$\begin{aligned} h_{\mathcal{X}}(\mathbf{x};\mathbf{W}_{\mathcal{X}}) &= \arg\max\nolimits_{1\le k\le K} \mathbf{w}_{\mathcal{X}k}^T\mathbf{x}, \\ h_{\mathcal{Y}}(\mathbf{y};\mathbf{W}_{\mathcal{Y}}) &= \arg\max\nolimits_{1\le k\le K} \mathbf{w}_{\mathcal{Y}k}^T\mathbf{y}, \end{aligned} \tag{5.1}$$

where $\mathbf{W}_{\mathcal{X}} \in \mathbb{R}^{K\times d_{\mathcal{X}}}$ and $\mathbf{W}_{\mathcal{Y}} \in \mathbb{R}^{K\times d_{\mathcal{Y}}}$ define two K-dimensional linear subspaces in modality $\mathcal{X}$ and $\mathcal{Y}$ respectively.

For each cross-modal training pair $(\mathbf{x}_i, \mathbf{y}_j)$ and their similarity label s_{ij}, one may define a similar empirical loss term $\ell(h_{\mathcal{X}}^i, h_{\mathcal{Y}}^j, s_{ij})$ as Equation (4.2). Here we look at a more general form of loss function

$$\ell(h_{\mathcal{X}}^i, h_{\mathcal{Y}}^j, s_{ij}) \triangleq d(I(h_{\mathcal{X}}^i = h_{\mathcal{Y}}^j), s_{ij}), \tag{5.2}$$

where $d(a,b)$ could be any proper loss function defined with respect to a prediction a and the target b, and $\mathrm{I}(\cdot)$ is the binary indicator function. It is not hard to see that the empirical loss in (4.2) is a special case of the above loss function. Intuitively, the loss function in (5.2) is dependent on the relationship rather than the value of the pair of hash codes. This is reasonable because we only need the hash codes to preserve the similar or dissimilar relationship among pairs. Here we do not restrict ourselves to any specific choice of $d(\cdot,\cdot)$ in order to arrive at a more general solution.

The overall learning objective is to find $\mathbf{W}_{\mathcal{X}}$ and $\mathbf{W}_{\mathcal{Y}}$ that minimize the aggregate loss over all the training pairs,

$$\mathcal{L}(\mathbf{W}_{\mathcal{X}}, \mathbf{W}_{\mathcal{Y}}) = \sum_{s_{ij}\in\mathcal{S}} \ell(h_{\mathcal{X}}^i, h_{\mathcal{Y}}^j, s_{ij}). \tag{5.3}$$

Note that $\mathbf{W}_{\mathcal{X}}$ and $\mathbf{W}_{\mathcal{Y}}$ factor into the above objective function because $h_{\mathcal{X}}^i$ and $h_{\mathcal{Y}}^j$ are functions of $\mathbf{W}_{\mathcal{X}}$ and $\mathbf{W}_{\mathcal{Y}}$.

5.3 Optimization

In this subsection, we introduce two alternative optimization techniques to solve the cross-modal hash function learning problem in (5.3). For mathematical convenience, in the following discussion we use the vectorized notation of (5.1) as defined in (4.4): $\mathbf{h}^i_{\mathcal{X}} \equiv \mathbf{h}_{\mathcal{X}}(\mathbf{x}_i; \mathbf{W}_{\mathcal{X}})$, $\mathbf{h}^j_{\mathcal{Y}} \equiv \mathbf{h}_{\mathcal{Y}}(\mathbf{y}_j; \mathbf{W}_{\mathcal{Y}})$.

5.3.1 Upper Bound Minimization-based Solution

In fact, the loss term of a cross-modal pair can be upper bounded by

$$\begin{aligned}
&\ell(\mathbf{h}^i_{\mathcal{X}}, \mathbf{h}^j_{\mathcal{Y}}, s_{ij}) \leq \\
&\max_{\mathbf{g}^{ij}_{\mathcal{X}}, \mathbf{g}^{ij}_{\mathcal{Y}}} [\ell(\mathbf{g}^{ij}_{\mathcal{X}}, \mathbf{g}^{ij}_{\mathcal{Y}}, s_{ij}) + (\mathbf{g}^{ij}_{\mathcal{X}})^T \mathbf{W}_{\mathcal{X}} \mathbf{x}_i + (\mathbf{g}^{ij}_{\mathcal{Y}})^T \mathbf{W}_{\mathcal{Y}} \mathbf{y}_j] \\
&\quad - (\mathbf{h}^i_{\mathcal{X}})^T \mathbf{W}_{\mathcal{X}} \mathbf{x}_i - (\mathbf{h}^j_{\mathcal{X}})^T \mathbf{W}_{\mathcal{Y}} \mathbf{y}_j
\end{aligned} \tag{5.4}$$

This upper bound directly follows from the following inequality

$$\begin{aligned}
&\max_{\mathbf{g}^{ij}_{\mathcal{X}}, \mathbf{g}^{ij}_{\mathcal{Y}}} [\ell(\mathbf{g}^{ij}_{\mathcal{X}}, \mathbf{g}^{ij}_{\mathcal{Y}}, s_{ij}) + (\mathbf{g}^{ij}_{\mathcal{X}})^T \mathbf{W}_{\mathcal{X}} \mathbf{x}_i + (\mathbf{g}^{ij}_{\mathcal{Y}})^T \mathbf{W}_{\mathcal{Y}} \mathbf{y}_j] \\
&\quad \geq \ell(\mathbf{h}^i_{\mathcal{X}}, \mathbf{h}^j_{\mathcal{Y}}, s_{ij}) + (\mathbf{h}^i_{\mathcal{X}})^T \mathbf{W}_{\mathcal{X}} \mathbf{x}_i + (\mathbf{h}^j_{\mathcal{X}})^T \mathbf{W}_{\mathcal{Y}} \mathbf{y}_j
\end{aligned}$$

Note that the above $\max$ should be taken with the one-hot constraints in (4.4). We do not write them underneath the $\max$ function to avoid notational clutter.

Thus, our objective boils down to minimizing the following function with respect to $\mathbf{W}_{\mathcal{X}}$ and $\mathbf{W}_{\mathcal{Y}}$

$$\begin{aligned}
&\Theta(\mathbf{W}_{\mathcal{X}}, \mathbf{W}_{\mathcal{Y}}) = \\
&\sum_{s_{ij} \in \mathcal{S}} \Big\{ \max_{\mathbf{g}^{ij}_{\mathcal{X}}, \mathbf{g}^{ij}_{\mathcal{Y}}} [\ell(\mathbf{g}^{ij}_{\mathcal{X}}, \mathbf{g}^{ij}_{\mathcal{Y}}, s_{ij}) + (\mathbf{g}^{ij}_{\mathcal{X}})^T \mathbf{W}_{\mathcal{X}} \mathbf{x}_i \\
&\quad + (\mathbf{g}^{ij}_{\mathcal{Y}})^T \mathbf{W}_{\mathcal{Y}} \mathbf{y}_j] - (\mathbf{h}^i_{\mathcal{X}})^T \mathbf{W}_{\mathcal{X}} \mathbf{x}_i - (\mathbf{h}^j_{\mathcal{X}})^T \mathbf{W}_{\mathcal{Y}} \mathbf{y}_j \Big\}
\end{aligned} \tag{5.5}$$

Note that the upper bound in (5.5) is convex-concave and piece-wise linear with respect to $\mathbf{W}_{\mathcal{X}}$ and $\mathbf{W}_{\mathcal{Y}}$. It is not differentiable because both the $\max$ term and ($\mathbf{h}^i_{\mathcal{X}}$, $\mathbf{h}^j_{\mathcal{Y}}$) depend on those projections. Also note that $\mathbf{W}_{\mathcal{X}}$ and $\mathbf{W}_{\mathcal{Y}}$ are not independent of each other because the cross-modal influence is propagated through the $\max$ term.

Our perceptron-like learning algorithm involves the following alternating optimization steps.

First, consider $\mathbf{W}_{\mathcal{X}}$ and $\mathbf{W}_{\mathcal{Y}}$ are fixed. We need to solve the $\max$ problem of the augmented error in the square brackets: $\ell(\mathbf{g}^{ij}_{\mathcal{X}}, \mathbf{g}^{ij}_{\mathcal{Y}}, s_{ij}) + (\mathbf{g}^{ij}_{\mathcal{X}})^T \mathbf{W}_{\mathcal{X}} \mathbf{x}_i + (\mathbf{g}^{ij}_{\mathcal{Y}})^T \mathbf{W}_{\mathcal{Y}} \mathbf{y}_j$. This discrete-optimization admits a global optimal solution. Specifically, it is not hard to see that the solution corresponds to the maximum entry in the following matrix with index p and q

$$m_{pq} = \begin{cases} \bar{x}_i^{(p)} + \bar{y}_j^{(q)} + d(1, s_{ij}) & \text{if } p = q \\ \bar{x}_i^{(p)} + \bar{y}_j^{(q)} + d(0, s_{ij}) & \text{otherwise} \end{cases} \tag{5.6}$$

where $1 \leq p, q \leq K$ and $\bar{x}_i^{(p)}$ and $\bar{y}_j^{(q)}$ denote the p^{th} and q^{th} dimension of $\mathbf{W}_{\mathcal{X}}\mathbf{x}_i$ and $\mathbf{W}_{\mathcal{Y}}\mathbf{y}_j$, respectively. Assuming that the entry at (p^*, q^*) of the matrix attains the maximum value, the maxima of the augmented error, denoted by $(\widehat{\mathbf{g}}^{ij}_{\mathcal{X}}, \widehat{\mathbf{g}}^{ij}_{\mathcal{Y}})$, are 1-of-$K$ binary vectors with the p^*th and the q^*th dimension set to 1. On the other hand, $(\mathbf{h}^i_{\mathcal{X}}, \mathbf{h}^j_{\mathcal{Y}})$ are the hashing codes selecting the maximal entries in the projected vectors $\mathbf{W}_{\mathcal{X}}\mathbf{x}_i$ and $\mathbf{W}_{\mathcal{Y}}\mathbf{y}_j$.

Now considering that $(\widehat{\mathbf{g}}^{ij}_{\mathcal{X}}, \widehat{\mathbf{g}}^{ij}_{\mathcal{Y}})$ and $(\mathbf{h}^i_{\mathcal{X}}, \mathbf{h}^j_{\mathcal{Y}})$ are fixed, $\Theta(\mathbf{W}_{\mathcal{X}}, \mathbf{W}_{\mathcal{Y}})$ becomes a linear function of $\mathbf{W}_{\mathcal{X}}$ and $\mathbf{W}_{\mathcal{Y}}$ and one update the weight matrices with the following perceptron-like learning rule:

$$\begin{aligned} \mathbf{W}_{\mathcal{X}} &\leftarrow \mathbf{W}_{\mathcal{X}} + \eta(\mathbf{h}^i_{\mathcal{X}} - \widehat{\mathbf{g}}^{ij}_{\mathcal{X}})\mathbf{x}_i^T \\ \mathbf{W}_{\mathcal{Y}} &\leftarrow \mathbf{W}_{\mathcal{Y}} + \eta(\mathbf{h}^j_{\mathcal{Y}} - \widehat{\mathbf{g}}^{ij}_{\mathcal{Y}})\mathbf{y}_j^T, \end{aligned} \tag{5.7}$$

where η is the step size.

5.3.2 Softmax Approximation-based Solution

The softmax approximation method in Chapter 4 can be used in a similar way here. Now we present a more detailed and general interpretation on the implications of such approximation in the context of cross-media hashing.

Let ($\mathbf{p}_i$, $\mathbf{q}_j$) be the softmax vectors for the cross-modal pair $(\mathbf{x}_i, \mathbf{y}_j)$ (i.e. $\mathbf{p}_i \triangleq \sigma(\mathbf{W}_{\mathcal{X}}\mathbf{x}_i)$ and $\mathbf{q}_j \triangleq \sigma(\mathbf{W}_{\mathcal{Y}}\mathbf{y}_j)$, where $\sigma(\cdot)$ is defined in Equation (4.6)). Indeed, $h^i_{\mathcal{X}}$ and $h^i_{\mathcal{Y}}$ can be regarded as two *independent* discrete random variables with the probability distribution

$$\begin{aligned} P(h^i_{\mathcal{X}} = k|\mathbf{W}_{\mathcal{X}}) &= p_{ik} \\ P(h^j_{\mathcal{Y}} = k|\mathbf{W}_{\mathcal{Y}}) &= q_{jk}, \end{aligned} \tag{5.8}$$

where $k \in \{0, \cdots, K-1\}$, and p_{ik} and q_{jk} are the k^{th} dimension of $\mathbf{p}_i$ and $\mathbf{q}_j$ respectively. The probability that two hash codes take the same value, denoted as π_{ij}, can be computed as

$$\begin{aligned} \pi_{ij} \triangleq\ & P(h^i_{\mathcal{X}} = h^j_{\mathcal{Y}}|\mathbf{W}_{\mathcal{X}}, \mathbf{W}_{\mathcal{Y}}) \\ &= \textstyle\sum_{k=1}^{K} P(h^i_{\mathcal{X}} = k|\mathbf{W}_{\mathcal{X}})P(h^j_{\mathcal{Y}} = k|\mathbf{W}_{\mathcal{Y}}) \\ &= \textstyle\sum_{k=1}^{K} p_{ik}q_{jk} = \mathbf{p}_i^T\mathbf{q}_j \end{aligned} \tag{5.9}$$

Based on (5.9) and (5.2), one can compute the expected loss for a cross-modal pair as

$$\begin{aligned} \mathbb{E}[\ell_{ij}] &= P(h^i_{\mathcal{X}} \neq h^j_{\mathcal{Y}}|\mathbf{W}_{\mathcal{X}}, \mathbf{W}_{\mathcal{Y}})d(0, s_{ij}) + P(h^i_{\mathcal{X}} \neq h^j_{\mathcal{Y}}|\mathbf{W}_{\mathcal{X}}, \mathbf{W}_{\mathcal{Y}})d(1, s_{ij}) \\ &= \big[d(1, s_{ij}) - d(0, s_{ij})\big]\pi_{ij} + d(0, s_{ij}), \end{aligned} \tag{5.10}$$

where ℓ_{ij} is short for $\ell(h^i_{\mathcal{X}}, h^j_{\mathcal{Y}}, s_{ij})$. Note that the expectation is dependent on the model coefficients $\mathbf{W}_{\mathcal{X}}$, $\mathbf{W}_{\mathcal{Y}}$ and the softmax smoothness α; we omit these to avoid notational clutter. We can also deduce the expectation of the overall objective, denoted as $\tilde{\mathcal{L}_\alpha}(\mathbf{W}_{\mathcal{X}}, \mathbf{W}_{\mathcal{Y}})$ (i.e. $\tilde{\mathcal{L}_\alpha}(\mathbf{W}_{\mathcal{X}}, \mathbf{W}_{\mathcal{Y}}) \triangleq$

$\mathbb{E}[\mathcal{L}(\mathbf{W}_{\mathcal{X}}, \mathbf{W}_{\mathcal{Y}})]$):

$$\begin{aligned}\tilde{\mathcal{L}_\alpha}(\mathbf{W}_{\mathcal{X}}, \mathbf{W}_{\mathcal{Y}}) &= \sum_{s_{ij}\in\mathbf{S}} \Big\{ \big[d(1, s_{ij}) - d(0, s_{ij})\big]\pi_{ij} + d(0, s_{ij}) \Big\} \\ &= \sum_{s_{ij}\in\mathbf{S}} a_{ij}\mathbf{p}_i^T\mathbf{q}_j + const. \\ &= \text{trace}\left(\mathbf{P}\mathbf{A}\mathbf{Q}^T\right) + const,\end{aligned} \tag{5.11}$$

where $\mathbf{P} = [\mathbf{p}_1 \ \cdots \ \mathbf{p}_{N_{\mathcal{X}}}]$ and $\mathbf{Q} = [\mathbf{q}_1 \ \cdots \ \mathbf{q}_{N_{\mathcal{Y}}}]$ are K-by-N matrices with softmax vectors in each column, and the entries of the $N_{\mathcal{X}}$-by-$N_{\mathcal{Y}}$ matrix $\mathbf{A}$ are defined as

$$a_{ij} = d(1, s_{ij}) - d(0, s_{ij}). \tag{5.12}$$

Interestingly, we find that the form of the overall objective in (5.11) is remarkably similar to what we have obtained in the single-media context (i.e. Equation (4.5)). Note that these two equations are derived in different contexts and with different assumptions. The resemblance is worthy of discussion here to provide some insight on the proposed hashing framework. Actually, (4.5) can be regarded as a special case of (5.11) by enforcing three conditions: 1) $\mathcal{X} = \mathcal{Y}$; 2) $\alpha \to \infty$; and 3) define $d(a, b)$ according to (4.2).

The generalized formulation in (5.11) allows us to devise more universal solutions for problems of this form. Specifically, we find that the objective functions is the linear combination of π_{ij}'s, therefore we only compute the derivatives of π_{ij} as follows

$$\begin{aligned}\frac{\partial \pi_{ij}}{\partial \mathbf{W}_{\mathcal{X}}} &= [\mathbf{p}_i \circ \mathbf{q}_j - (\mathbf{p}_i^T\mathbf{q}_j)\mathbf{p}_i]\mathbf{x}_i^T \\ \frac{\partial \pi_{ij}}{\partial \mathbf{W}_{\mathcal{Y}}} &= [\mathbf{q}_j \circ \mathbf{p}_i - (\mathbf{q}_j^T\mathbf{p}_i)\mathbf{q}_j]\mathbf{y}_j^T,\end{aligned} \tag{5.13}$$

where '$\circ$' stands for the element-wise Hadamard product. The above gradients can be used to update the weights when data samples are presented in a streaming fashion.

Algorithm 2: Linear Subspace Ranking Hashing

```
   Input: Data $\mathbf{X}$, $\mathbf{Y}$ and cross-modal similarity labels $\mathbf{S}$.
   Output: Linear projections $\mathbf{W}_\mathcal{X}$ and $\mathbf{W}_\mathcal{Y}$.
1  Initialization: Set $\mathbf{W}_\mathcal{X}$ and $\mathbf{W}_\mathcal{Y}$ to random values from Gaussian distribution
2  repeat
3      Randomly select a training batch $\mathbf{X}_b, \mathbf{Y}_b$ and obtain the batchwise similarity labels
         $\mathbf{S}_b$ accordingly
4      Compute $\mathbf{A}$ based on the choice of loss function
5      Compute $\mathbf{P}$ and $\mathbf{Q}$ by applying the softmax function to each column of $\mathbf{W}_\mathcal{X}\mathbf{X}_b$
         and $\mathbf{W}_\mathcal{Y}\mathbf{Y}_b$
6      Set $\mathbf{Q}_s = \mathbf{Q}\mathbf{A}^T$, $\mathbf{P}_s = \mathbf{P}\mathbf{A}^T$
7      Update projection matrix $\mathbf{W}_\mathcal{X}$ and $\mathbf{W}_\mathcal{Y}$ according to equation (5.7)
8  until Convergence
```

In practice, mini-batches are used in the learning process. In detail, let $\mathbf{X}_b$ and $\mathbf{Y}_b$ be two mini-batches randomly sampled from $D_\mathcal{X}$ and $D_\mathcal{Y}$ respectively. By summing pairwise gradients over the batches, we obtain the following equations for weights update

$$\begin{aligned}\mathbf{W}_\mathcal{X} &\leftarrow \mathbf{W}_\mathcal{X} - \eta[\mathbf{P} \circ \mathbf{Q}_s - \mathbf{P}\,\mathrm{diag}(\mathbf{Q}_s^T\mathbf{P})]\mathbf{X}_b^T \\ \mathbf{W}_\mathcal{Y} &\leftarrow \mathbf{W}_\mathcal{Y} - \eta[\mathbf{Q} \circ \mathbf{P}_s - \mathbf{Q}\,\mathrm{diag}(\mathbf{P}_s^T\mathbf{Q})]\mathbf{Y}_b^T,\end{aligned} \tag{5.14}$$

where the 'diag' operator outputs a diagonal matrix by retaining the diagonal entries of a square input matrix, $\mathbf{Q}_s = \mathbf{Q}\mathbf{A}^T$, $\mathbf{P}_s = \mathbf{P}\mathbf{A}$ and η is the learning rate. Note that the notations of $\mathbf{P}$, $\mathbf{A}$ and $\mathbf{Q}$ are the same as in (5.11) except that they are defined over the mini-batch. The above learning procedures for one pair of hash functions are summarized in Algorithm 2.

5.4 Discussion of different loss functions

Unlike most other cross-modal hashing algorithms, where the loss functions are deeply coupled to the problem formulations and optimization process, our ranking-based hash learning framework can easily accommodate different loss functions. Our discussion of the learning algorithm does not assumed any specific $d(a, b)$. Here we discuss a few commonly used loss functions.

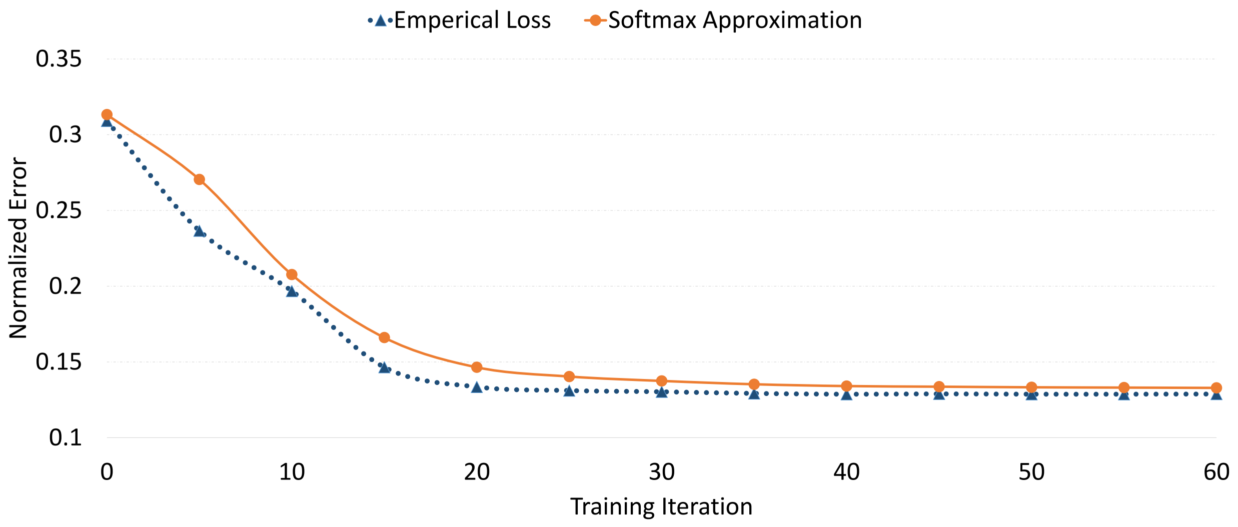


Figure 5.1: An example of the empirical loss (5.3) and its softmax approximation as a function of the training iteration number. The update rules in (5.14) are used with mini-batch size set to 500. The algorithm converges in less then 50 iterations.

LSRH-L1 If we consider $\lambda = 1$, the empirical loss in (4.2) is equivalent to the L_1 loss.

$$d_{l1}(\pi_{ij}, s_{ij}) = |\pi_{ij} - s_{ij}|. \tag{5.15}$$

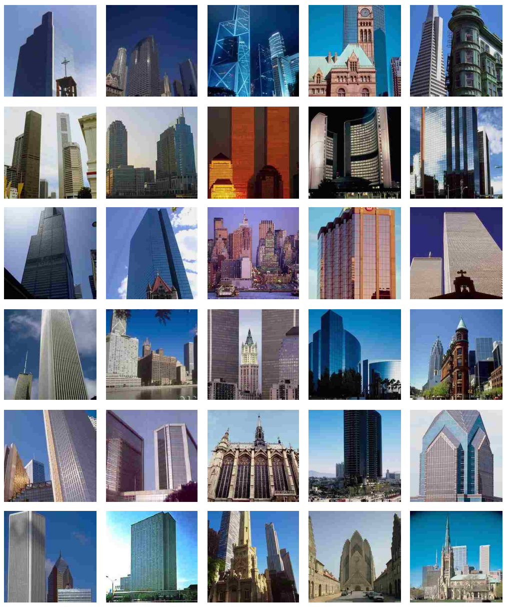
It's straightforward to incorporate λ into the formulation. This is the default loss function used in LSRH.

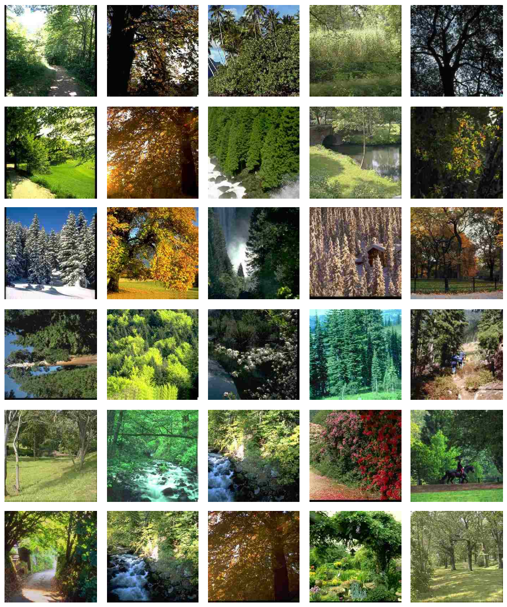
LSRH-L2 The L2 loss punishes the prediction's deviation from target by imposing a squared error

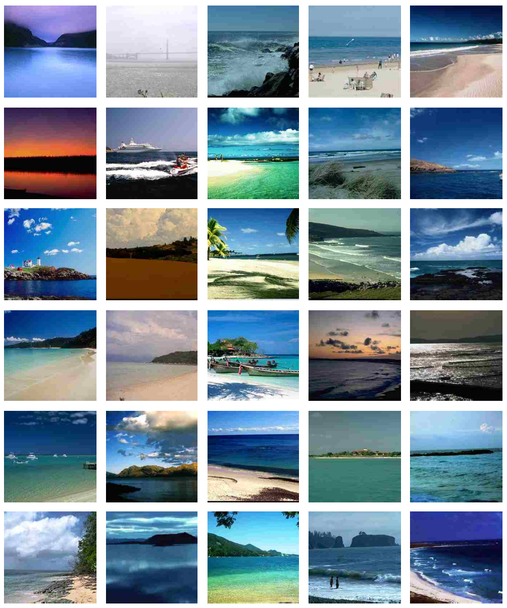
$$d_{l2}(\pi_{ij}, s_{ij}) = (\pi_{ij} - s_{ij})^2. \tag{5.16}$$

LSRH-Exp The exponential loss function is defined as

$$d_{ex}(\pi_{ij}, s_{ij}) = \exp\{-[2(\pi_{ij} - 0.5)][2(s_{ij} - 0.5)]\}. \tag{5.17}$$


(a) building


(b) tree


(c) sea

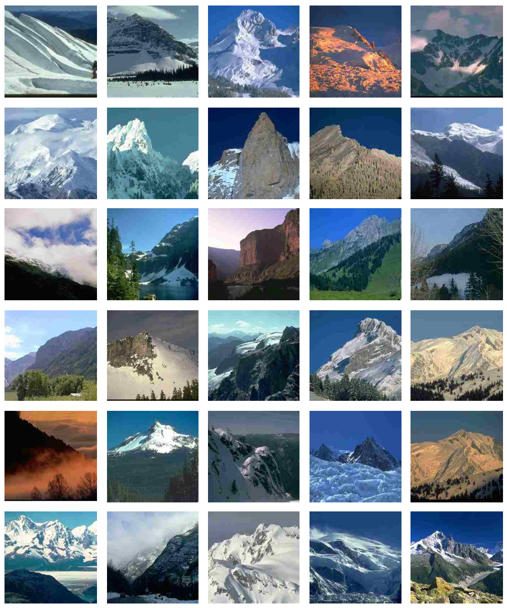

(d) mountain

Figure 5.2: Example of textual queries on the labelme image database. From (a) to (b), the query keywords are 'building', 'tree', 'sea' and 'mountain' respectively. The code length of LSRH is set to 30 bits and the top 30 results are shown. Images in the grid are ordered from left to right and top to bottom based on the Hamming distance of their hash codes to the hash code of the textual query.

Algorithm 3: Sequential Learning of Multiple Codes

```
  Input: Data $\mathbf{X}$, $\mathbf{Y}$ and cross-modal similarity labels $\mathbf{S}$.
  Output: Projections $\{\mathbf{W}_{\mathcal{X}}^{(l)}\}_{l=1}^{L}$ and $\{\mathbf{W}_{\mathcal{Y}}^{(l)}\}_{l=1}^{L}$.
1 Initialization: Set the weight $\omega_{ij}$ of all pairs to one
2 for $l = 1$ to $L$ do
3     Obtain $\mathbf{W}_{\mathcal{X}}^{(l)}$ and $\mathbf{W}_{\mathcal{Y}}^{(l)}$ using based on upper-bound minimization or softmax
        approximation
4     Compute hash codes for all samples using (5.1)
5     Set $\epsilon_l = \mathcal{L}(\mathbf{W}_{\mathcal{X}}, \mathbf{W}_{\mathcal{Y}})/(N_{\mathcal{X}} \cdot N_{\mathcal{Y}})$
6     Evaluate the quantity $\sigma = \ln(1/\epsilon_l - 1)$
7     Update the weighting coefficients using $\omega_{ij}^{(l+1)} = \omega_{ij}^{(l)} \exp[\sigma \cdot \ell(h_{\mathcal{X}}^{i}, h_{\mathcal{Y}}^{j}, s_{ij})]$
8     Normalize $\omega_{ij}$'s such that $\sum_{i,j} \omega_{ij}^{(l+1)} = N_{\mathcal{X}} \cdot N_{\mathcal{Y}}$
9 end
```

Here we have applied the mapping from $\pi_{ij} \to 2(\pi_{ij} - 0.5)$ to transform the range of the values from $[0, 1]$ to $[-1, 1]$. Similar operations are applied to s_{ij}.

LSRH-Hinge The hinge loss uses different punishment rules for similar and dissimilar pairs

$$d_{hg}(\pi_{ij}, s_{ij}) = \begin{cases} \max(0.5 - \pi_{ij}, 0), & s_{ij} = 1 \\ \lambda(\pi_{ij} - 0), & s_{ij} = 0. \end{cases} \tag{5.18}$$

Note that the Hinge loss pushes the similarity scores of similar pairs to be at least 0.5. This is reasonable as similar pairs are only required to have a sufficiently large similarity score, but not necessarily the maximum.

Although these loss functions seem very different, all of the optimization steps in Algorithm

2 remain the same except for line 4, which can be substituted with the following equations:

$$\begin{aligned}
\mathbf{A}_{l1} &= \lambda\mathbf{E} - (\lambda + 1)\mathbf{S} \\
\mathbf{A}_{l2} &= 2(\Pi - \mathbf{S}) \\
\mathbf{A}_{ex} &= 2(\mathbf{E} - 2\mathbf{S}) \circ d_{ex}(\Pi, \mathbf{S}) \\
\mathbf{A}_{hg} &= (\mathbf{E} - \mathbf{S}) + \mathbf{S} \circ [\mathrm{I}(\Pi > 0.5) - \mathbf{E}].
\end{aligned} \tag{5.19}$$

sky,building,mountain,tree,ground,person ,plant,window,door ,staircase,pot plant,bell,person walking,road,wall,curb,sidewalk, poster,ground grass,pane ,stone ball,building ,truck,car,sign,cars side ,window ,door,tree trunk,streetlight

sky,hill,sea water,rocks,field,water sea,bridge,tree,mountain,beach,ship,shrub,rock,ocean water,buildings,building ,trees,plain,palm tree,palm tree ,field grass,boat,person sitting,building,sand beach,

Tree trunk, branch,leaves,sky,sun,trees, land,brush,tree leaves,mountain,tree,house ,ground grass,stone,water river,brushes,slope,waterfall,ground,stones,undergrowth,sticks,path,rocks,chimney

mountain,clouds,trees,fog bank,rock,snowy mountain,mountain pass,sky,tree,branch,rocky mountain,land,road,snow,housecrop,waterfall,refuge,skier crop,skier,person walking,forest

sky,building,tree,building ,skyscraper,car,buildings,river water ,road,parking place,boat crop,dock, streetlight,poster,tower,clock,palmtree,plants,fountain,hall,groundgrass,boat,bridge, buildings crop

sky,mountain,river water,plain,rocks,building,path,field,tree crop,snowy mountain,tree,pole crop,building ,trees,ground,hill,fog bank,person walking,desert field,brushes,valley,stone

sky,building,mountain,buildings, road,car , window ,column,arcade,house ,house,ground grass,car rear,handrail,porch,parquing meter ,car side rear,tree,person walking,building

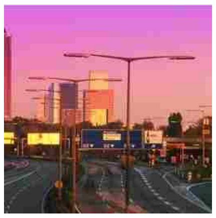
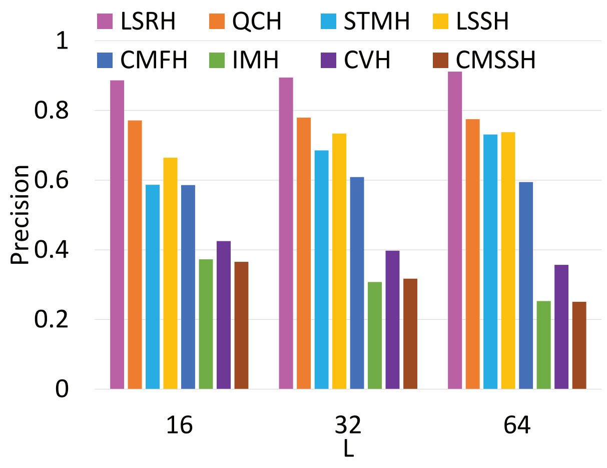

sky,building,tree,buildings, road,car,streetlight,car frontal,park,mountain,house ,house,ground grass,car rear,handrail,window,window ,porch,parquing meter ,carsiderear,trees,sidewalk,bus,van ,bus ,tower,toll gate,

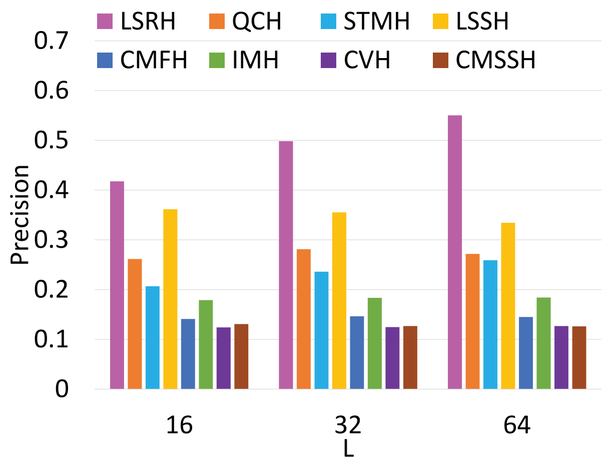
Figure 5.3: Example of image annotations by using images to query tags. The code length of LSRH is set to 30 bits and approximately 30 of the most relevant tags are shown. The ground truth for each image is shown in green. The order of the tags is based on the Hamming distance between hash codes.

Here Π denotes the matrix formed by π_{ij}, $d_{ex}(\cdot,\cdot)$ and $\mathrm{I}(\cdot)$ apply to the input matrix in an element-wise fashion, and $\mathbf{E}$ is a matrix of all ones.

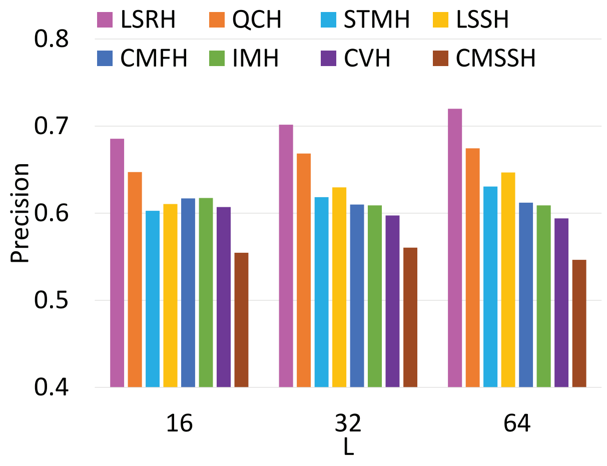
Recall that we have only discussed the learning procedures for one pair of hash functions so far. To learn multiple hash codes, we adopt a similar sequential learning method (i.e. Adaboost) as in Chapter 4, and the steps are summarized in Algorithm 3.



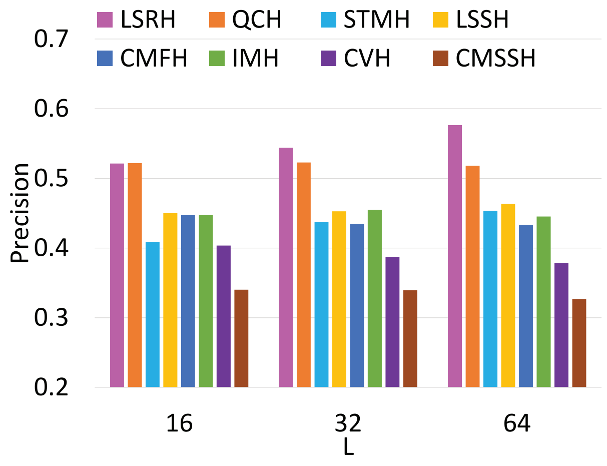
(a) Labelme



(b) Wikipedia



(c) MIRFlickr

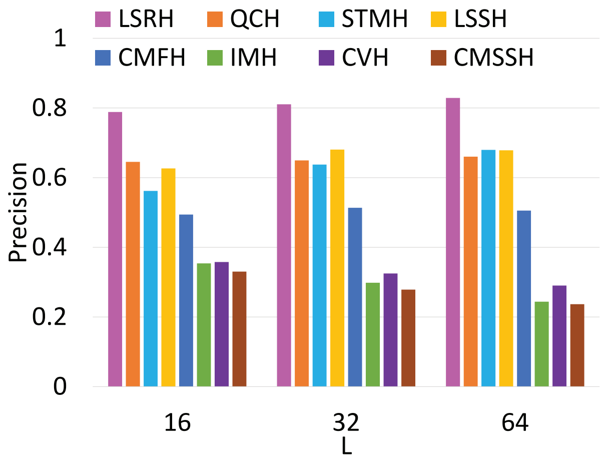


(d) NUSWIDE

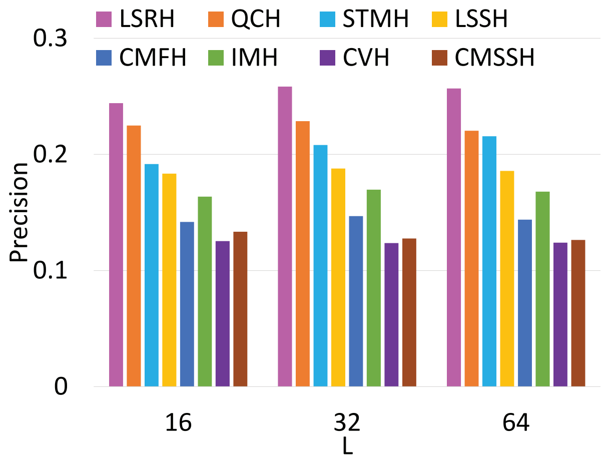
Figure 5.4: Top-100 precision of text-query-image on all datasets, with the hash code varying from 16 bits to 64 bits.

Table 5.1: Cross-modal mAP results of the proposed LSRH and compared baselines on all of the benchmark datasets. The length of the hash code is varied from 16 bits to 64 bits and the mAP of the top 50 neighbors are reported (i.e. R = 50). The best results are shown in bold. Our LSRH outperforms all baselines in almost all datasets and benchmarks.

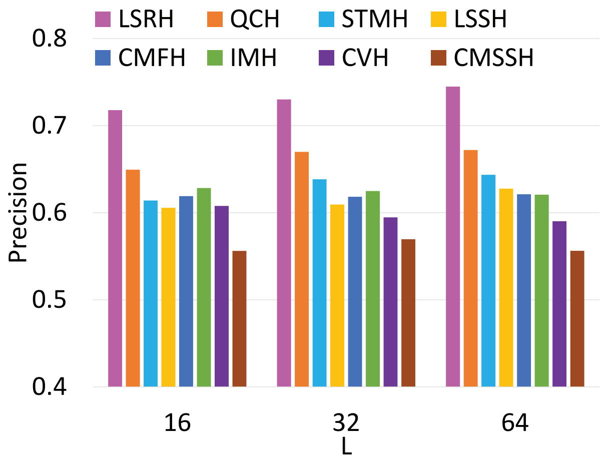
	Text query image				**Image query text**			
Method	Labelme	Wiki	MIRFlickr	NUS	Labelme	Wiki	MIRFlickr	NUS
				16 bits				
CVH	0.5630	0.1931	0.6434	0.4466	0.4574	0.1930	0.6381	0.4529
CMSSH	0.4369	0.1802	0.5997	0.4080	0.3857	0.1930	0.6381	0.4529
IMH	0.4958	0.2642	0.6406	0.4950	0.4447	0.2290	0.6615	0.4657
LSSH	0.7408	0.5002	0.6430	0.5013	0.6977	0.2284	0.6368	0.5201
CMFH	0.6938	0.2174	0.6510	0.4960	0.5835	0.2045	0.6528	0.4648
QCH	0.8151	0.3420	0.6602	**0.5562**	0.6727	0.2582	0.6595	0.5295
STMH	0.6487	0.2924	0.6315	0.4459	0.6098	0.2366	0.6387	0.5165
LSRH	**0.8883**	**0.5459**	**0.7108**	0.5525	**0.8048**	**0.2707**	**0.7395**	**0.5450**
				32 bits				
CVH	0.5555	0.1982	0.6368	0.4395	0.4191	0.1865	0.6301	0.4356
CMSSH	0.3760	0.1768	0.5688	0.3927	0.3229	0.1749	0.6069	0.4833
IMH	0.4202	0.2703	0.6416	0.4975	0.3943	0.2331	0.6576	0.4815
LSSH	0.7913	0.5220	0.6707	0.5066	0.7317	0.2355	0.6421	0.5318
CMFH	0.7285	0.2265	0.6444	0.4831	0.6159	0.2161	0.6542	0.4249
QCH	0.8314	0.3908	0.6899	0.5584	0.6788	0.2568	0.7048	0.5463
STMH	0.7484	0.3251	0.6500	0.4797	0.6885	0.2505	0.6684	0.5617
LSRH	**0.8989**	**0.6626**	**0.7223**	**0.5701**	**0.8211**	**0.2816**	**0.7529**	**0.5724**
				64 bits				
CVH	0.5481	0.2083	0.6291	0.4351	0.3946	0.1885	0.6381	0.4250
CMSSH	0.3153	0.1918	0.5835	0.3822	0.3022	0.1702	0.5790	0.4731
IMH	0.3679	0.2781	0.6383	0.4885	0.3401	0.2275	0.6554	0.4903
LSSH	0.8085	0.5168	0.6908	0.5299	0.7147	0.2422	0.6616	0.5372
CMFH	0.7330	0.2290	0.6461	0.4824	0.6187	0.2148	0.6590	0.4087
QCH	0.8246	0.3839	0.6988	0.5565	0.6899	0.2510	0.7033	0.5531
STMH	0.7874	0.3772	0.6596	0.4955	0.7310	0.2616	0.6750	0.5696
LSRH	**0.9153**	**0.7258**	**0.7450**	**0.6068**	**0.8376**	**0.2914**	**0.7682**	**0.6012**



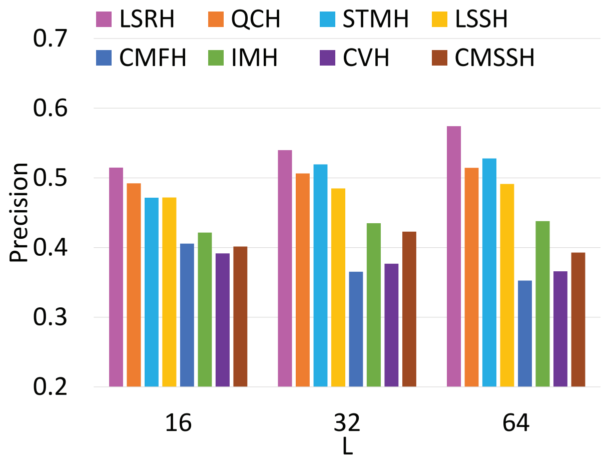
(a) Labelme



(b) Wikipedia



(c) MIRFlickr



(d) NUSWIDE

Figure 5.5: Top-100 precision of image-query-text on all datasets, with the hash code varying from 16 bits to 64 bits.

5.5 Experiments

5.5.1 Datasets

To evaluate the proposed algorithm, we choose four widely-used multimodal datasets: Wikipedia [26, 99], Labelme [110], MIRFLICKR [54, 26] and NUS-WIDE [110, 89, 99, 26].

The statistics of those datasets are shown in Table 6.1 and following are brief descriptions of each dataset.

Wiki. The wiki [73] dataset, crawled from Wikipedia's "featured articles", consists of $2,866$ documents which are image-text pairs and annotated with semantic labels of 10 categories. Each image in this dataset is represented as a 128-D bag-of-SIFT feature vector. For text documents, we extract the 1000-D tf-idf features over the most representative words.

LabelMe.[1] The LabelMe dataset [69] consists of 2688 images annotated by the objects' textual tags contained in them, such as "forest" and "mountain". Tags that occurs less than 3 times are discarded, resulting in 245 unique remaining tags. Each image is labeled as one of eight unique outdoor scenes, such as coast, forest and highway. Each image in this dataset is represented by a 512-D GIST vector and the corresponding textual tags are represented by the index vectors of selected tags.

MIRFLICKR.[2] The MIRFLICKR dataset [32] contains 25,000 images with associated textual tags. Each image-text pair is associated with one or more of 24 semantic labels. Tags that appear less than 20 times are first removed and then instances without tags or annotations are removed, resulting in 16738 instances remaining. Images in this dataset are described by 150-D edge histograms and the texts are represented as 500-D feature vectors obtained by applying PCA to the binary tagging vector. Instances are considered as similar if they share at least one common label.

NUS-WIDE.[3] The NUS-WIDE dataset [20] is a real-world image dataset with $269,648$ images. Each image has a number of textual tags and is labeled with one or more image concepts out of 81 concepts. We select the $186,577$ image-tag pairs belonging to the 10 largest concepts. In this dataset, the images are represented by 500-D bag-of-visual-words (BOVW) and the image

[1]http://people.csail.mit.edu/torralba/code/spatialenvelope/

[2]http://press.liacs.nl/mirflickr/

[3]http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm

tags are represented by 1000-D tag occurrence feature vectors.

Table 5.2: Statistics of benchmark datasets

	Features		**Classes**	**Size**	**Queries**
Dataset	Image	Text			
Labelme	512	245	8	2688	573
Wikipedia	128	1000	10	2866	693
MIRFlickr	150	500	24	16738	836
NUSWIDE	500	1000	10	186577	2000

5.5.2 Baselines

We have compared the proposed LSRH with seven well-known cross-modal hashing methods: Cross-view Hashing (CVH) [44], Cross-modal Similarity Sensitive Hashing (CMSSH) [10], Inter-media Hashing (IMH) [80], Latent Semantic Sparse Hashing (LSSH) [110], Collective Matrix Factorization Hashing (CMFH) [26], Semantic Topic Multimodal Hashing (STMH) [85] and Quantization Correlation Hashing (QCH) [89]. Those algorithms have been briefly introduced in Section 2.1 and are considered to be the current state-of-the-arts in cross-modal hash learning. The parameters for all the compared algorithms are either default ones or chosen according to the suggestions of the original papers to give the best performance.

5.5.3 Experiment settings

We follow previous work [110, 26, 85, 89] in choosing the training set and query set. In detail, for Labelme and Wikipedia, 20% of the data points are randomly selected as the query set, and the remaining data are used as the training set and retrieval database. For MIRFlickr and NUSWIDE, we randomly select approximately 5% and 1% of the dataset as queries respectively. The remaining data are used as the database for cross-modal retrieval. Moreover, we randomly select 5000 image-text pairs from the database for hash learning and apply the learned hash functions to the entire database to generate the hash codes. Such practice has been widely used in hash learning research [44, 10, 110, 54, 89, 85, 26] because it simulates real-world scenarios where the labeled data are limited compared to the entire data corpus, and this can well-demonstrate the out-of-sample extension capability of different hashing methods.

By default, LSRH uses the λ-parameterized L1 loss and the softmax approximation-based learning algorithm. Therefore, LSRH takes a single primary parameter: the subspace dimension K; and two hyper-parameters λ and α, which are the penalty coefficient and the softness of the softmax. We choose these parameters by using 5-fold cross-validation on a held-out subset in the training set. Specifically, we use linear search in log scale for K, and fix it to 4 for all the experiments. The effect of subspace dimension K will be discussed in detail in Section 5.5.5.4. As for λ and α, we use linear search over $\{0.5, 1.0, 2.0\}$ and $\{0.5, 0.8, 1.0\}$ respectively. In contrast to the binary hashing algorithms, our hashing code is K-ary. Therefore, we set $L = \lfloor N_b / \lceil \log_2 K \rceil \rfloor$ when comparing with other binary hash codes at N_b bits to ensure fairness.

We evaluate the retrieval performance of both text-query-image and image-query-text. Specifically, we follow the widely used metrics [80, 108, 72, 110, 89]: mean Average Precision (mAP), top-k precision and precision-recall for both retrieval tasks. All the experimental results are averaged over 5 independent runs.

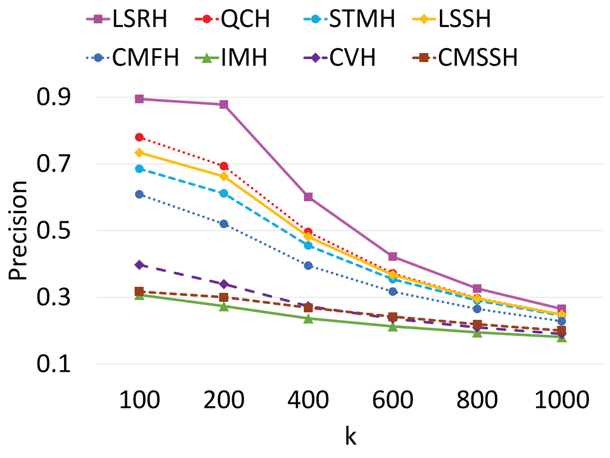
5.5.4 Comparison with baselines

5.5.4.1 Performance results

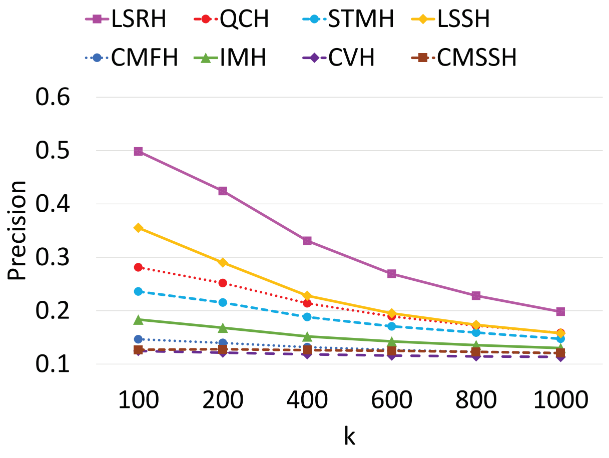
We vary the hash code from 16 bits to 64 bits and record the mAPs of LSRH and baselines on all the benchmark datasets in Table 5.1. We can observe that LSRH is competitive to or outperforms all the compared methods across different datasets and code lengths. In fact, the average performance advantage of LSRH is more than 5% across the four datasets in our experiments.

Table 5.3: Results of top-100 precision, precision-recall and training/testing time at 32 bits. The precision-recall values are computed as the area under the precision-recall curve.

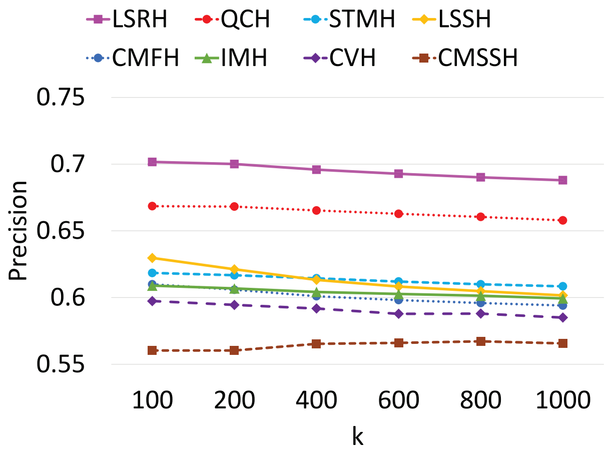
		Time (s)		**Precision (top 100)**			**Precesion-Recall**		
Method	**#Train**	Train	Test	T→I	I→T	Average	T→I	I→T	Average
				Labelme					
CVH	2151	1.3	0.005	0.3974	0.3250	0.3612	0.2917	0.2462	0.2690
CMSSH	2151	481.0	0.005	0.3172	0.2785	0.2979	0.2359	0.2311	0.2335
IMH	2151	5.5	0.008	0.3076	0.2984	0.3030	0.2270	0.2218	0.2244
LSSH	2151	263.4	14.9	0.7015	0.6805	0.6910	0.6234	0.5602	0.5918
CMFH	2151	10.8	0.005	0.5810	0.5136	0.5473	0.4821	0.3963	0.4392
QCH	2151	143.2	0.005	0.7796	0.6494	0.7145	0.5803	0.5452	0.5628
STMH	2151	3.7	0.2	0.6614	0.6375	0.6323	0.5727	0.5417	0.5572
LSRH	2151	13.7	0.009	**0.8944**	**0.8107**	**0.8526**	**0.8756**	**0.8485**	**0.8626**
				Wikipedia					
CVH	2173	0.08	0.005	0.1246	0.1237	0.1242	0.1147	0.1148	0.1148
CMSSH	2173	623.5	0.005	0.1268	0.1279	0.1207	0.1208	0.1205	0.1160
IMH	2173	5.7	0.01	0.1834	0.1697	0.1766	0.1430	0.1385	0.1408
LSSH	2173	136.3	8.8	0.3554	0.1878	0.2716	0.2470	0.1443	0.1957
CMFH	2173	9.3	0.006	0.1466	0.1469	0.1468	0.1240	0.1247	0.1244
QCH	2173	107.8	0.005	0.2813	0.2286	0.2550	0.2094	0.1882	0.1988
STMH	2173	6.3	0.5	0.2360	0.2081	0.2221	0.1863	0.1669	0.1766
LSRH	2173	14.2	0.02	**0.4983**	**0.2584**	**0.3784**	**0.3902**	**0.2325**	**0.3114**
				MIRFlickr					
CVH	5000	0.05	0.02	0.5974	0.5947	0.5961	0.5720	0.5716	0.5718
CMSSH	5000	350.1	0.02	0.5604	0.5695	0.5650	0.5580	0.5575	0.5578
IMH	5000	41.2	0.04	0.6090	0.6249	0.6170	0.5808	0.5809	0.5809
LSSH	5000	163.7	36.1	0.6297	0.6093	0.6195	0.5784	0.5759	0.5772
CMFH	5000	12.0	0.02	0.6100	0.6183	0.6142	0.5784	0.5787	0.5786
QCH	5000	228.7	0.02	0.6685	0.6698	0.6692	0.6011	0.6026	0.6019
STMH	5000	7.9	1.6	0.6185	0.6384	0.6285	0.5857	0.5842	0.5850
LSRH	5000	26.6	0.03	**0.7016**	**0.7301**	**0.7159**	**0.6688**	**0.6844**	**0.6766**
				NUSWIDE					
CVH	5000	0.8	0.4	0.3874	0.3768	0.3821	0.3462	0.3421	0.3445
CMSSH	5000	1084.4	0.4	0.3395	0.4229	0.3812	0.3211	0.3252	0.3418
IMH	5000	42.1	0.7	0.4549	0.4349	0.4449	0.3794	0.3729	0.3769
LSSH	5000	289.3	435.9	0.4527	0.4848	0.4688	0.3611	0.3510	0.3576
CMFH	5000	30.9	0.5	0.4348	0.3652	0.4000	0.3574	0.3483	0.3639
QCH	5000	586.6	0.4	0.5227	0.5064	0.5146	0.4341	0.4292	0.4312
STMH	5000	15.0	38.6	0.4374	0.5194	0.4784	0.3705	0.3549	0.3570
LSRH	5000	32.5	2.6	**0.5440**	**0.5398**	**0.5419**	**0.5132**	**0.5118**	**0.5125**



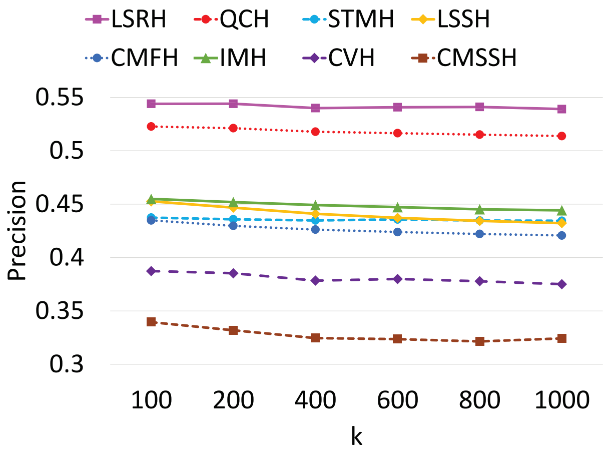
(a) Labelme



(b) Wikipedia



(c) MIRFlickr

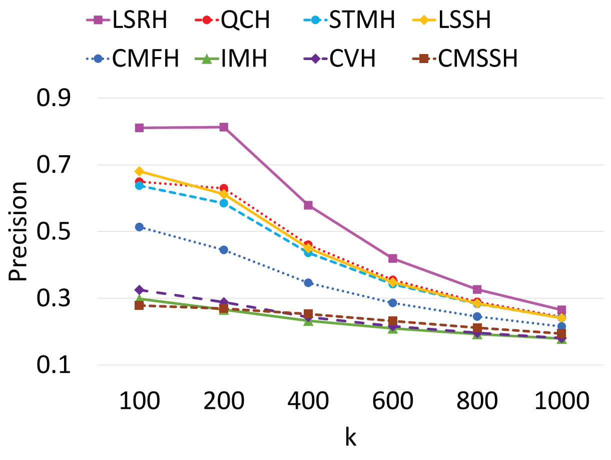


(d) NUSWIDE

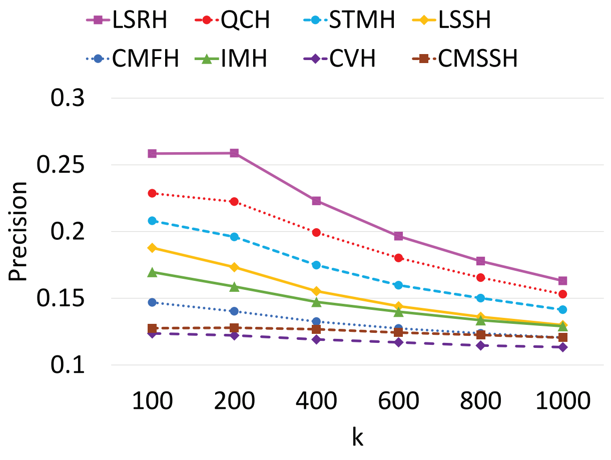
Figure 5.6: Top-k precision of text-query-image with 32-bit hash code and k varies from 100 to 1000.

Furthermore, when it comes to individual benchmarks, LSRH can beat the best baseline by up to 40%, for example, in the 64-bit text-query-image task on Wikipedia, while most of the baselines are only competitive in some benchmarks or datasets. For instance, STMH is very competitive in the image-query-text task on NUSWIDE, but not as competitive in the text-query-image task on the same dataset; LSSH performs very well in the text-query-image task on Wikipedia, but is not equally good in the the image-query-text task compared to some other baselines.

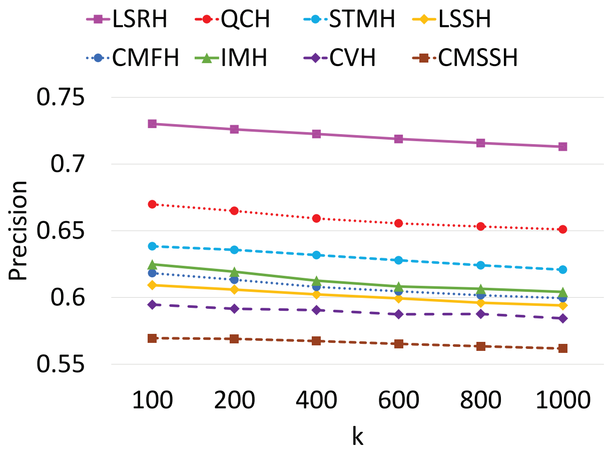
We find that the performance difference between text-query-image and image-query-text tasks are very close in most of the datasets except for Wikipedia. Such observation is consistent with the results reported in previous research [110, 89, 85]. As revealed in [110], there is a significant semantic gap between the images and the descriptive documents in the Wikipedia dataset.



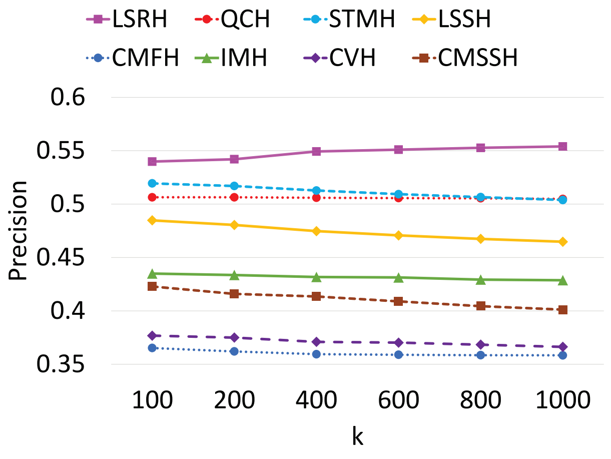
(a) Labelme



(b) Wikipedia

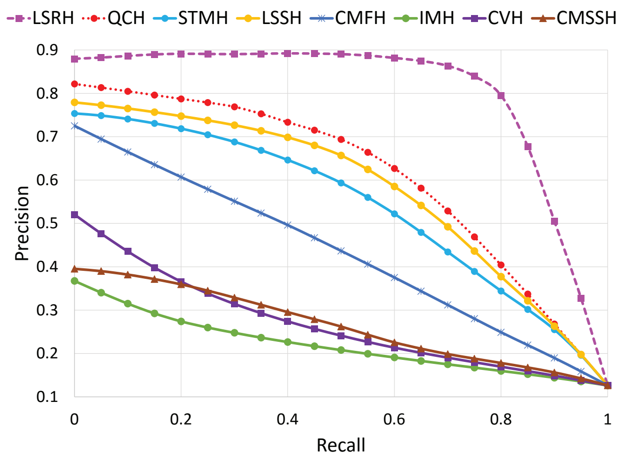


(c) MIRFlickr

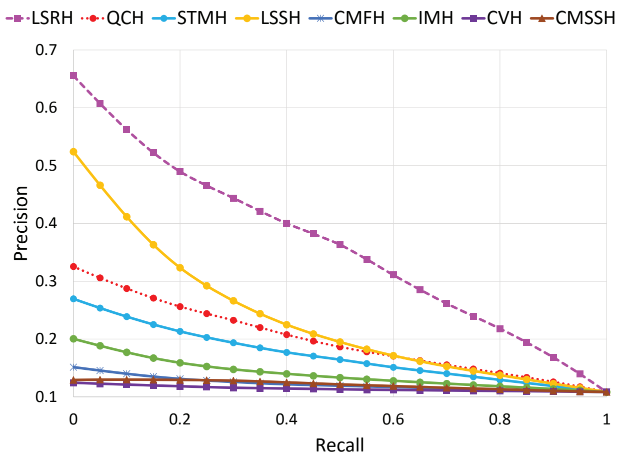


(d) NUSWIDE

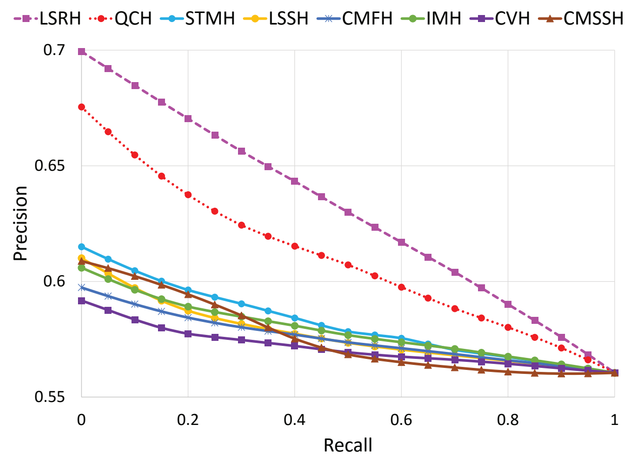
Figure 5.7: Top-k precision of image-query-query with 32-bit hash code and k varies from 100 to 1000.



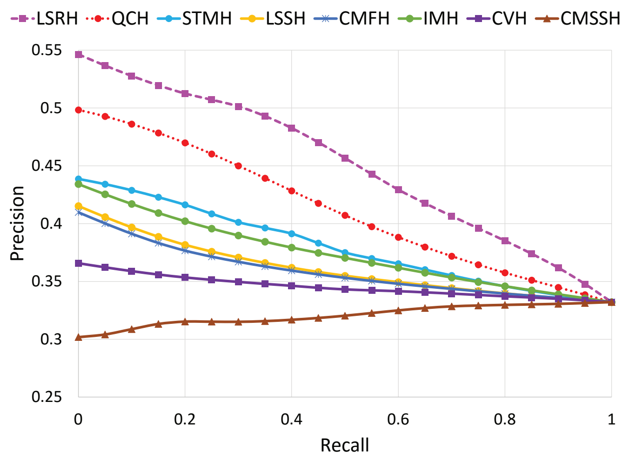
(a) Labelme



(b) Wikipedia



(c) MIRFlickr

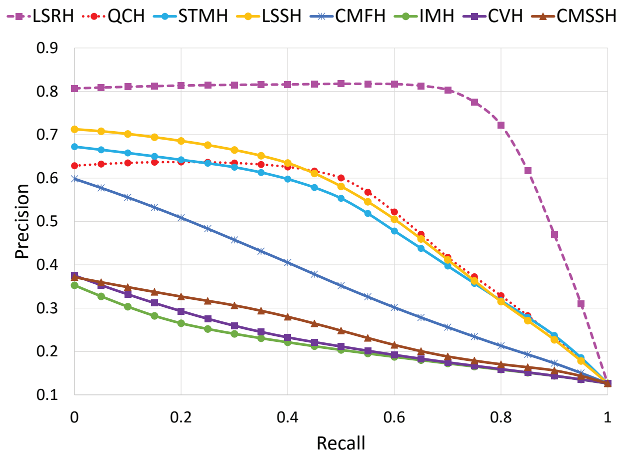


(d) NUSWIDE

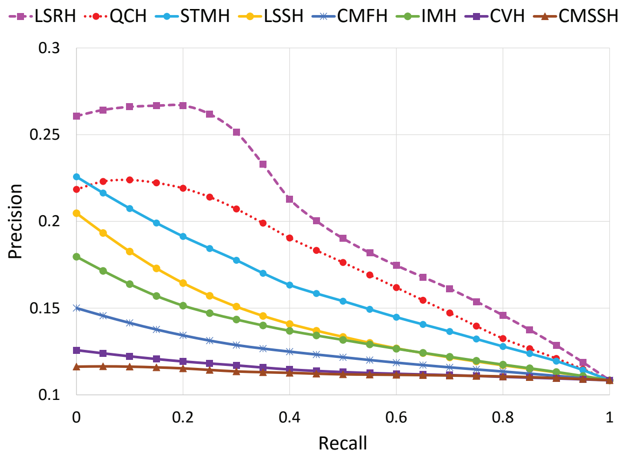
Figure 5.8: Precision-recall curves of text-query-image with 32-bit hash code. Larger area under the curve indicates better performance. LSRH achieves the best performance.

The texts in Wikipedia are much better than the images in describing the semantic concept, thus leading to lower mAPs when images are used to query against the text database.

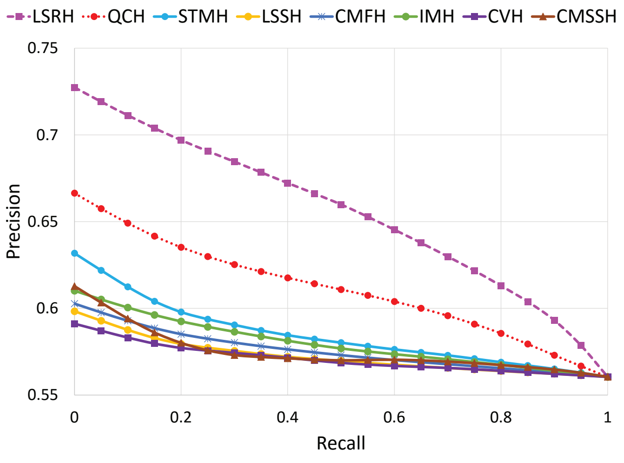
Another interesting finding is that the performance of LSRH monotonically increases when the hash code becomes longer, while the performances of some of the baselines such as CVH, CMSSH and IMH do not increase and sometimes even drop with the increase of code length.



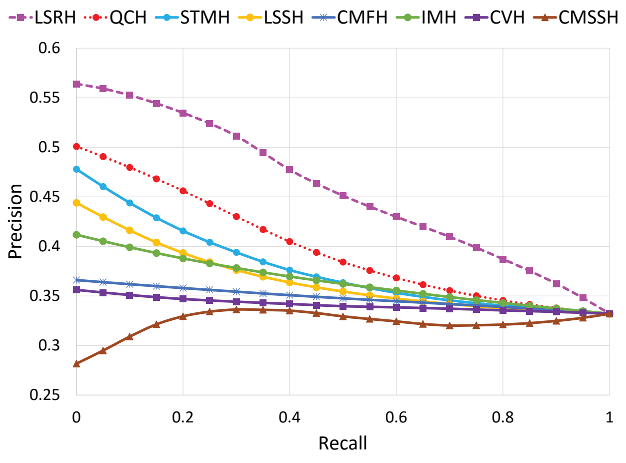
(a) Labelme



(b) Wikipedia



(c) MIRFlickr



(d) NUSWIDE

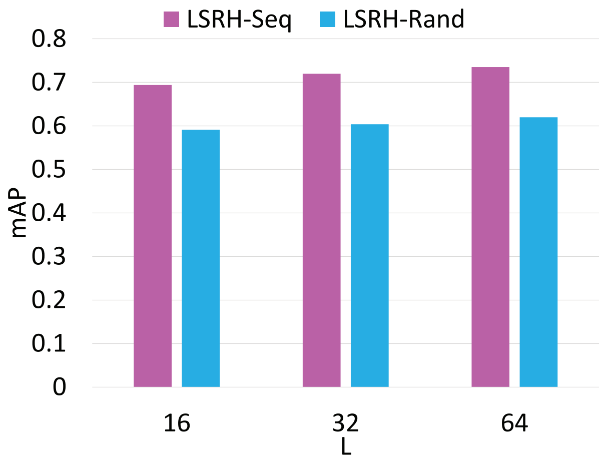
Figure 5.9: Precision-recall curves of image-query-text with 32-bit hash code. Larger area under the curve indicates better performance. LSRH achieves the best performance.

This has also been observed by [26, 99]. In fact, those methods are similar in the sense that they all solve certain eigen-decomposition problems with orthogonality constraints to reduce bit correlations. As a result, most discriminative information is contained within the first few bits. As the code becomes longer, the hash code will be gradually dominated by indiscriminative hash bits, which do not contribute to the retrieval performance.

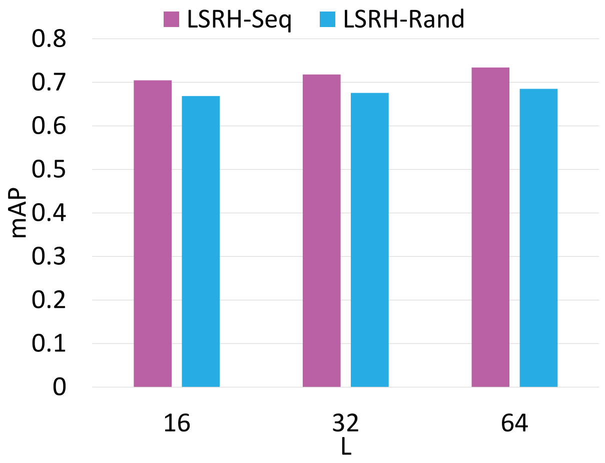
In addition to mAP, we also report the performances of different methods in terms of K-nearest neighbor precision and precision-recall. The results for those benchmarks on all the datasets are shown in Table 5.3, Figure 5.6, Figure 5.7, Figure 5.8 and Figure 5.9. Note that the precision-recall values in Table 6.5 are computed as the area under the precision-recall curves, and larger values mean better overall performance. From Table 6.5, we can observe that the relative performances of different methods are generally consistent with that of mAP. Specifically, LSRH consistently outperforms all the baselines across different datasets in both top-k precision and precision-recall, and the average performance gap between LSRH and the best baselines on the four datasets are 35%, 48%, 10% and 12% respectively.

In addition to the retrieval performance metrics, we have also shown the training and testing time for different algorithms under the same system settings in Table 5.3. The results are shown in seconds. We note that the proposed LSRH can be trained very fast compared to most of the baseline methods; while some of the competitive baselines such as LSSH and QCH take much longer to train. In terms of testing time, linear hashing algorithms are clear winners since the hash encoding stage only involves simple linear transformations followed by zero-thresholding. LSRH also falls into the linear hashing category since the ranking operation is performed upon linear projections. The close testing time of LSRH compared to most of the other methods confirms the efficiency of ranking-based hash encoding. We note that LSSH and STMH are two exceptions in the experiments with much longer testing time. This is caused by large-matrix inverse computations or nonlinear transformations in the hash encoding step, thus making them less effective in real-world applications that require online hash encoding. The moderate training and testing time further confirms the effectiveness of the proposed LSRH.

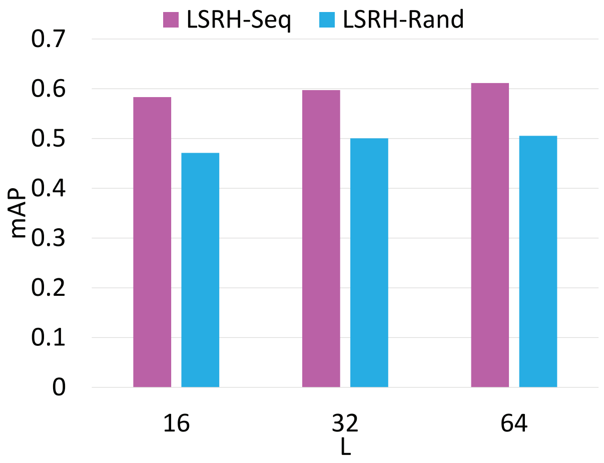
Overall, the proposed LSRH consistently achieves superior performance against the baselines in different metrics and datasets, with only moderate training and testing time. Such good performance of LSRH can be attributed to several reasons. Firstly, the ranking-based hash functions can be very useful in preserving the cross-modal similarities. Second, the proposed hash learning procedures are both efficient and effective in learning the ranking-based hash functions. Third, the ranking-structure of cross-modal data exploited by the proposed hashing framework is very useful in bridging the semantic gap between different modalities.



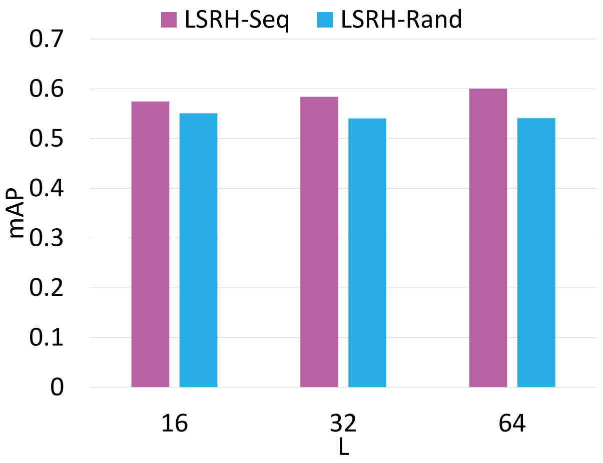
(a) MIRFlickr (tex-query-image)



(b) MIRFlickr (image-query-text)



(c) NUSWIDE (tex-query-image)



(d) NUSWIDE (image-query-text)

Figure 5.10: Comparison of different strategies in generating multiple hash codes. The results are obtained with 32-bit hash code on MIRFlickr and NUSWIDE.

5.5.4.2 Scalability Study

In order to test the scalability of the proposed hash learning method, we profile its training time under varying training sizes and compare with the baseline methods. Specifically, we vary the training size from 10^6 to 10^8 pairs on two of the larger datasets: MIRFlickr and NUSWIDE. All of the profiled algorithms run on the same system with Intel Xeon E5-2680 CPU @ 2.5 GHz and 128 GB of memory. The results of this test are summarized in Table 5.4. Note that all of the compared algorithms are implemented using MATLAB, and are therefore comparable under the same test settings. We can observe that LSRH can be trained significantly faster than the most competitive baselines such as LSSH and QCH. Additionally, the training time of LSRH only increases moderately with the increase in training size. The short training time and good scalability with large training sets demonstrate the effectiveness of our learning procedures.

Table 5.4: Training time of 32-bit hash code on MIRFlickr and NUSWIDE. The training size is varied from 10^6 to 10^8 pairs and the results are in seconds.

	Training size (pairs)					
Method	10^6	10^7	10^8	10^6	10^7	10^8
	MIRFlickr			NUSWIDE		
CVH	0.04	0.04	0.07	0.8	1.1	1.1
CMSSH	197.8	219.2	220.1	508.6	504.9	508.6
IMH	0.5	5.0	119.4	0.6	5.3	119.7
LSSH	536.7	415.7	610.6	383.0	396.7	711.7
CMFH	1.2	6.2	28.6	3.1	10.6	53.8
QCH	46.8	134.5	366.1	89.4	234.0	603.3
STMH	6.8	16.1	36.5	10.9	26.0	66.6
LSRH	4.5	9.6	33.2	6.7	11.6	34.7

5.5.5 Algorithm Analysis

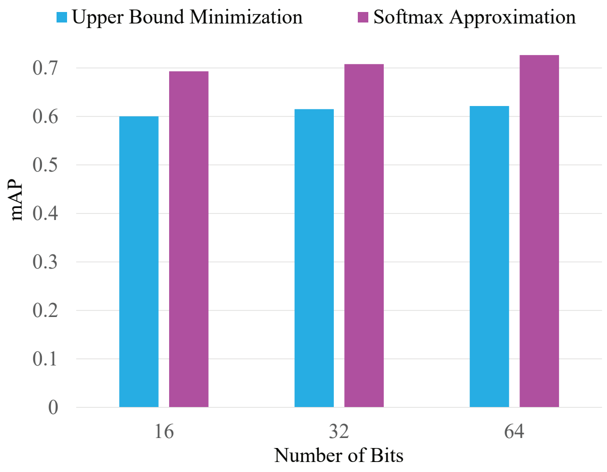
5.5.5.1 Effect of sequential learning

In this section, we study the role of boosting in learning multiple hash codes. Specifically, we consider a randomized version of LSRH, denoted as LSRH-Rand, where each hash code is learned independently with random initialization. To differentiate from LSRH-Rand, we denote the sequential version of LSRH as LSRH-Seq. The results of this experiment are shown in Figure 5.10. We note that LSRH-Seq demonstrates consistent performance boost over LSRH-Rand, with approximately 10% lead on average. The performance gap is mainly due to the code redundancy in LSRH-Rand. Specifically, in LSRH-Rand, multiple independent random initializations may correspond to the same local minima of the objective function; as a result, there exist redundant hash functions that compromise the amount of discriminative information contained in the resultant hash codes. On the other hand, LSRH-Seq learns each new code by taking advantage of the information from previous ones and focus on a different subset of the training data. Therefore, LSRH-Seq codes contain more discriminative information than LSRH-Rand codes of the same length.

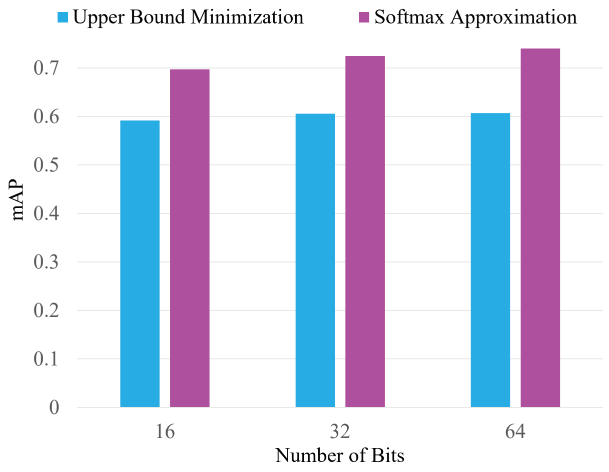
5.5.5.2 Different optimization Schemes

To learn the optimal cross-modal ranking subspaces, we have proposed two alternative optimization algorithms, that is, the upper bound minimization-based solution and softmax approximation-based solution. Here we compare the performances of these two solutions and briefly discuss about them. The results of this experiment are shown in Figure 5.11. As can be observed from the figure, the softmax approximation-based solution achieves better performances throughout this test. The inferior performance of the upper bound minimization-based solution is caused by the fact that the upper bound is not a tight bound, meaning that although the upper bound is strictly decreasing during the learning process, the actual training loss may not follow the upper bound exactly.

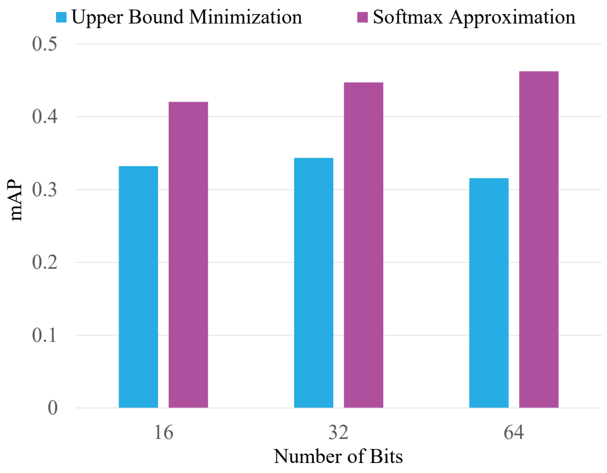
Therefore, the solution found by minimizing the upper bound may not be sufficiently optimized. In comparison, the softmax approximation is more close to the actual objective function, and the resultant solution is more optimal.



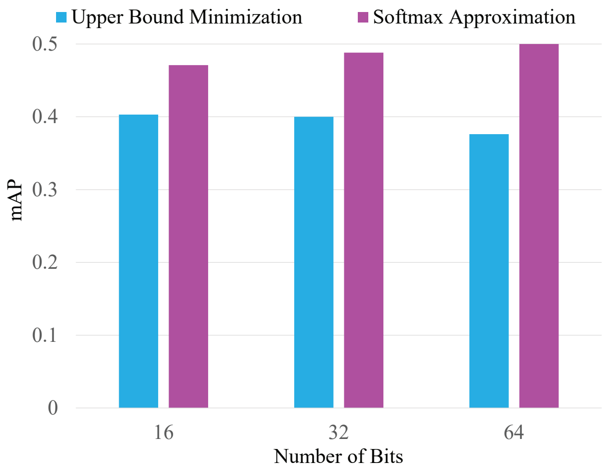
(a) MIRFlickr (tex-query-image)



(b) MIRFlickr (image-query-text)



(c) NUSWIDE (tex-query-image)



(d) NUSWIDE (image-query-text)

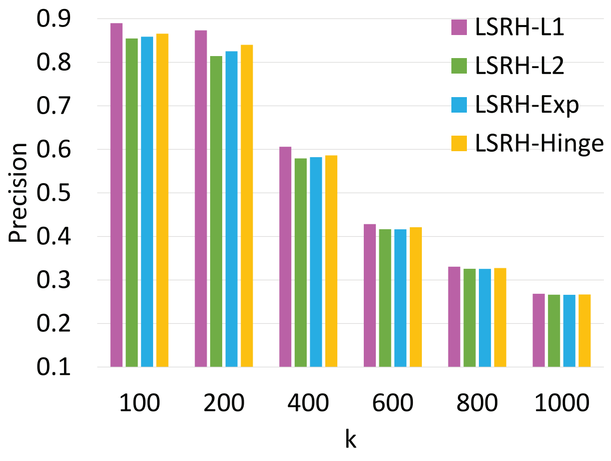
Figure 5.11: Comparison of different solutions. This experiment is performed on MIRFlickr and NUSWIDE, with 2000 and 3000 training samples respectively. The reported results are mAP over the top 50 returned neighbors.

5.5.5.3 Different loss functions

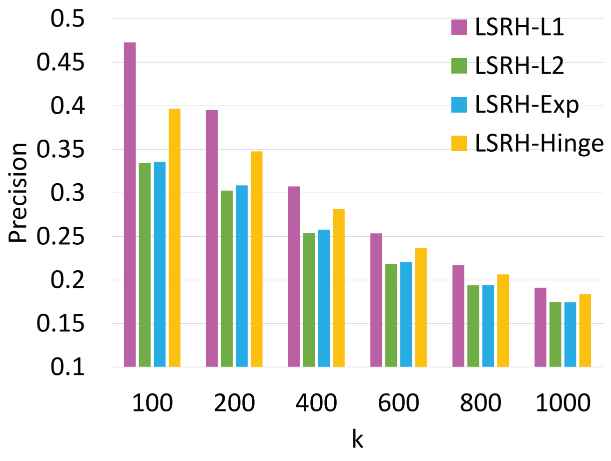
The proposed ranking-hash framework is able to incorporate different types of loss functions with minimal modification to the hash learning procedures. Here we compare the the performance of four different loss functions. Specifically, the loss functions included in this experiment are L1, L2, exponential and hinge loss. Details of these loss functions have been explained in Section 5.4. The results are illustrated in Table 5.5, Figure 5.12 and Figure 5.13. We observe that the default L1 loss function usually achieves the best performance. This may be explained by the fact that the L1 loss directly follows from the empirical loss, which is closely related to the performance metrics used in the similarity search. On the other hand, the performance values of different loss functions are very close in most test cases. The slight differences in the performances are caused by different ways of assigning penalties to the predictions. In general, the ability to accommodate different loss functions in a unified hashing framework greatly extends the flexibility of LSRH.

Table 5.5: mAP and top-100 precision of LSRH using different loss functions. The hash code length is set to 32 bits

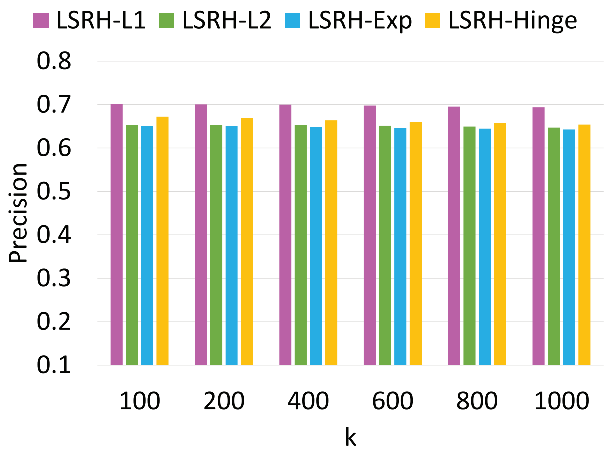
	Text query image				**Image query text**			
Loss	Labelme	Wiki	MIRFlickr	NUS	Labelme	Wiki	MIRFlickr	NUS
				mAP				
L1	**0.8931**	**0.6168**	**0.7230**	**0.5855**	**0.8167**	**0.2849**	**0.7680**	0.5483
L2	0.8664	0.4360	0.6774	0.5314	0.8136	0.2813	0.6712	0.5412
Exp	0.8569	0.4367	0.6730	0.5777	0.8005	0.2874	0.6963	**0.5997**
Hinge	0.8597	0.5292	0.6995	0.5849	0.8027	0.2771	0.7474	0.5507
				Precision (top 100)				
L1	**0.8898**	**0.4730**	**0.7008**	**0.5525**	**0.8108**	0.2569	**0.7392**	0.5198
L2	0.8547	0.3343	0.6526	0.4948	0.7806	0.2518	0.6421	0.5193
Exp	0.8587	0.3357	0.6506	0.5474	0.7854	**0.2580**	0.6718	**0.5660**
Hinge	0.8656	0.3966	0.6721	0.5315	0.7981	0.2479	0.7120	0.5177



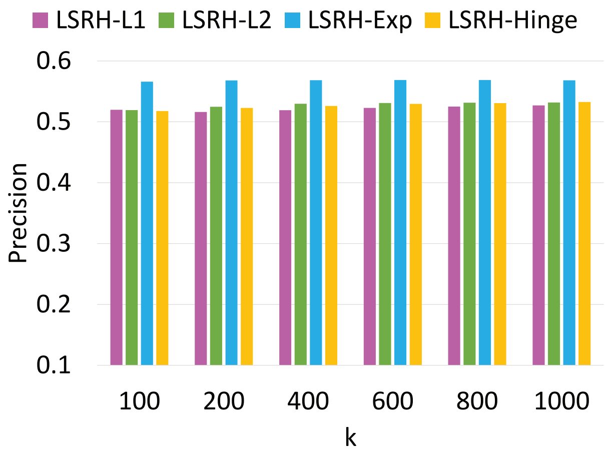
(a) Labelme



(b) Wikipedia



(c) MIRFlickr

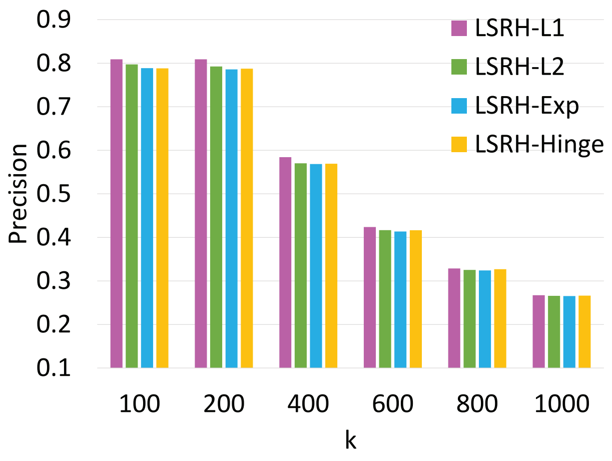


(d) NUSWIDE

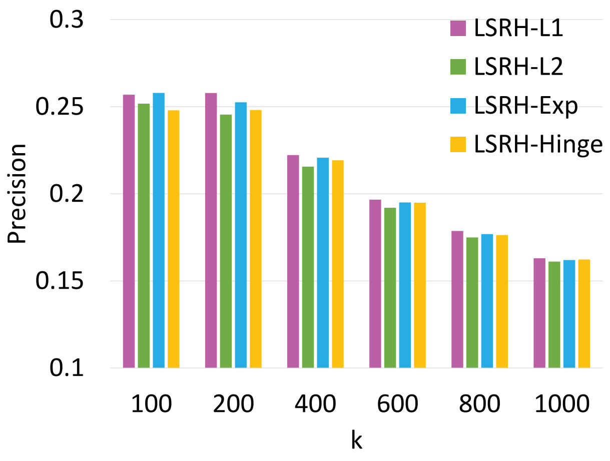
Figure 5.12: Text-query-image top-k precision of LSRH trained with different loss functions. The length of the hash code is set to 32 bits and k varies from 100 to 1000.

5.5.5.4 Effect of subspace dimension

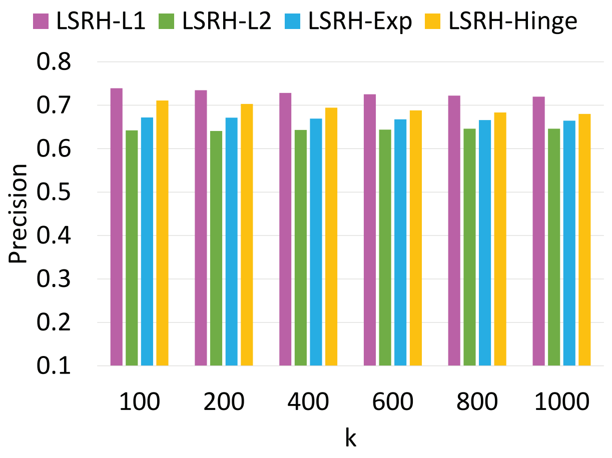
Here we study the performance of LSRH with different subspace dimension K. In this experiment, we vary K from 2 up to 32 in linear scale of $\log_2 K$ (i.e. $K = 2^1, \cdots, 2^5$). Note that the choice of K is not restricted to the powers of two, and the reason for our settings here is to make sure that there exists a bijective mapping between each K-ary code and a binary hash code. Recall that we train $\lfloor N_b / \lceil \log_2 K \rceil \rfloor$ LSRH codes when evaluating the performance at N_b bits. Here we choose N_b to be 60 bits since it is the common multiple of 1 to 5. This way, we can make sure that the same amount of information is contained in each LSRH codeword irrespective of the choice of K. The results of this experiment are shown in Figure 5.14. We note from the test results this experiment that the effect of K is not the same on different datasets and retrieval tasks.



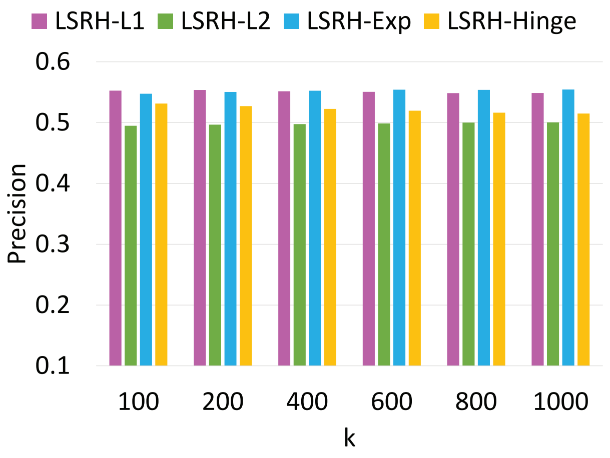
(a) Labelme



(b) Wikipedia

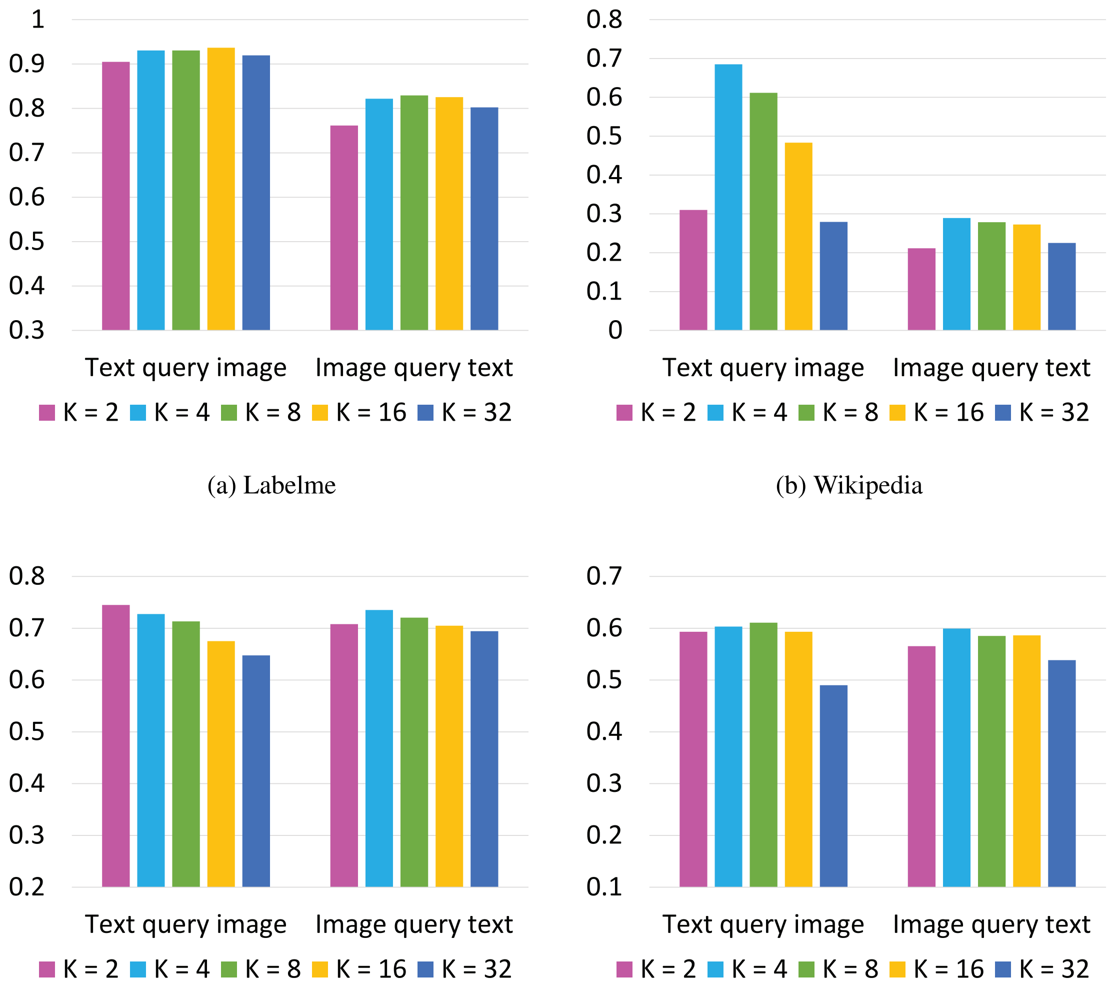


(c) MIRFlickr



(d) NUSWIDE

Figure 5.13: Image-query-text top-k precision of LSRH trained with different loss functions. The length of the hash code is set to 32 bits and k varies from 100 to 1000.



(a) Labelme (b) Wikipedia

(c) MIRFlickr (d) NUSWIDE

Figure 5.14: The mAP with 60-bit hash code under different subspace dimensions.

Such distinctions indicate that the ranking-structures inherent in different datasets are different and the best practice is to use cross-validation to choose the optimal subspace dimension. Overall, we find that $K = 4$ is an all-around good choice across different benchmarks and datasets.

5.6 Summary

In this chapter, we further develop the ranking-based hashing method into a more flexible cross-media hashing framework. We present two alternative methods to solve the cross-media hash function learning problem and show its connection to the single-media hashing problem under the same learning framework. We carry out extensive experiments on widely used multimodal datasets and compare with a range of state-of-the-art cross-modal hashing methods. The experimental results demonstrate the superiority of our method in generating highly discriminative and compact hash codes for cross-modal retrieval tasks.

CHAPTER 6: LABEL PRESERVING DISCRETE MULTIMEDIA HASHING

6.1 Overview

In learning-based hashing, one typically needs to solve optimization problems which either have binary constraints or involve discontinuous and non-convex components (e.g. the sign function). Many existing algorithms try to solve a continuous version of the original problem by either relaxing the discrete constraints [96, 99, 59, 80] or finding continuous approximations [42, 58, 47, 54]. For example, our solution to the ranking-based hashing problems is based on a continuous approximation to the discrete objective function.

A different class of learning algorithms attempt to solve the discrete-constrained optimization problems directly, leading to the family of discrete hashing algorithms, and such initiatives are recently found to be more effective than continuous approximations because they can avoid quantization errors introduced in the approximation or relaxations. Representative works include Discrete Graph Hashing (DGH) [57], FastHash [52] and Column Sampling Discrete Hashing (COSDISH), etc. The quality of the hash codes generated by those algorithms are found to be better than that of relaxed solutions [57, 52]. Nevertheless, most of those algorithms have high computational costs because they are designed to preserve pairwise similarities which often lead to NP-hard Binary Quadratic Programming (BQP) problems. The complexity of the approximate solutions to those BQP problems are typically at least $O(n^2)$, making them not scalable to large-scale datasets. Moreover, most of the discrete methods have been proposed for single-media hashing and are not directly applicable for multimedia hashing.

To address the aforementioned limitations, we propose a scalable discrete hashing framework for multimedia data, termed as Label Preserving Multimedia Hashing (LPMH) hereinafter. Instead of dealing with the pairwise affinities among training samples, we explicitly optimize the binary hash codes to preserve the instance-wise semantic labels. Such an initiative can be highly computationally efficient, as the effective size of the training set is kept at $O(n)$, in contrast with the $O(n^2)$ training pairs in the pairwise case.

Additionally, the proposed LPMH adopts a flexible two-stage learning framework for joint binary codes and media-specific hash functions. Specifically, LPMH first learns the binary codes by iteratively solving a series of unconstrained Binary Integer Programming (BIP) subproblems. Unlike most of the existing methods, which are tightly coupled with certain objective functions, the proposed discrete optimization method is a unified solution to different types of loss functions. For out-of-sample extension, the joint binary codes are used as the common labels to coordinate the learning of multiple media-specific hash functions. To sum up, we major contributions of the proposed work are summarized as follows:

- We propose a general approach to solve the classification-based binary code inference problem. We formulate the binary code optimization problem as a series of binary integer programming subproblems, and show that they have a simple and unified analytic solution irrespective of the type of loss function used.

- Our formulation can be easily combined with bit balance constraints and we provide a simple yet effective solution to the constrained optimization problem with linear time complexity. Additionally, we give proofs of the optimality of the proposed solution.

- As the first work that combines classification-based discrete optimization with the two-stage learning framework, the proposed method is both scalable and flexible. In particular, the method is equally optimized for single-media and cross-media similarity search.

- We extensively evaluate the proposed algorithm in both single-media and cross-media retrieval tasks, and we have separately compared with the state-of-the-arts of both settings.

The experimental results indicate that our algorithm compares favorably against the state-of-the-arts across a number of large-scale datasets and multiple retrieval benchmarks.

The remaining of the chapter is organized as follows. The proposed Label Preserving Multimedia Hashing approach is introduced and discussed in detail in Section 6.2. We extensively evaluate our algorithm and discuss the experimental results in Section 6.3, followed by the conclusion in Section 6.4.

6.2 Label Preserving Multimedia Hashing

6.2.1 Problem Definition

Suppose we have a training set of N instances, denoted as $\mathcal{S} = \{s_1, s_2, \cdots, s_N\}$ with s_n being the n^{th} instance. We consider the case that an instance s_n can be associated with one or multiple media types and denote the feature vector of the m^{th} media type as $\mathbf{x}_n^m \in \mathbb{R}^{d_m}$, where $1 \leq m \leq M$ and M is the number of media types. We assume the training set belong to C different classes and each instance has a class label, denoted as an indicator vector $\mathbf{t}_n \in \{0,1\}^C$, where a non-zero entry indicates the instance belongs to the corresponding class. Our objective is to learn the L-bit label-preserving binary codes $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_N] \in \{1, -1\}^{L \times N}$, as well as a set of media-specific hash functions $\mathcal{H} = \{h^1(\mathbf{x}), h^2(\mathbf{x}), \cdots, h^M(\mathbf{x})\}$ such that new data samples from heterogeneous media types can be mapped to a common Hamming space. We explicitly decouple the binary code inference and the hash function learning stages using a two-step hashing framework, as will be explained in detail in this section.

6.2.2 Binary Code Optimization

As indicated by recent studies [52, 106, 77], high-quality binary codes should also be good feature representations for classification tasks. Therefore, we explicitly preserve the semantic label information in the binary code learning stage. Formally, the general label-preserving binary code learning problem can be written as

$$\begin{aligned} &\min_{\mathbf{B},f} \sum_{n}^{N} \mathcal{L}(f(\mathbf{b}_n), \mathbf{t}_n) + \Omega(f) + \Omega(\mathbf{B}), \\ &\text{s.t. } \mathbf{B} \in \{-1, 1\}^{L \times N} \end{aligned} \tag{6.1}$$

where $\mathcal{L}(\mathbf{y}, \mathbf{t})$ could be any proper loss function defined with respect to a prediction $\mathbf{y}$ and a target label $\mathbf{t}$, $f(\cdot)$ is a decision function that maps an input to a decision output with the same dimensionality as the label vector, $\Omega(f)$ is the regularization term for the decision function, and $\Omega(\mathbf{B})$ is the regularizer for the binary codes. Note that one of the most common constraint for binary hashing is the bit balance constraint, which requires each bit position to have equal number of 1's and 0's to maximize the information entropy carried by the binary bits. Such constraint leads to the regularizer $\Omega(\mathbf{B}) = ||\mathbf{B1}||_1$, where $\mathbf{1}$ is the vector of all ones. We first solve the problem without the constraint on $\mathbf{B}$, and then we derive the solution to the constrained optimization problem.

Generally speaking, a better feature representation would need a simpler classifier to achieve the same level of classification performance. In order for the binary codes to be the best for classification, we choose $f(\cdot)$ to be the simplest decision function; that is, the linear function

$$f(\mathbf{x}) = \mathbf{W}^T \mathbf{x}, \tag{6.2}$$

where $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_C] \in \mathbb{R}^{L \times C}$ is the linear model coefficient matrix. Therefore, we focus on the following optimization problem

$$\begin{aligned} &\min_{\mathbf{B},\mathbf{W}} \frac{1}{N} \sum_{n}^{N} \mathcal{L}(\mathbf{W}^T \mathbf{b}_n, \mathbf{t}_n) + \frac{\mu}{2} ||\mathbf{W}||_F^2, \\ &\text{s.t. } \mathbf{b}_n \in \{-1, 1\}^L, n = 1, 2, \cdots, N, \end{aligned} \tag{6.3}$$

where μ is the regularization coefficient and $||\cdot||_F$ is the Frobenius norm. Note that we have used the averaging form of the loss term to make the reguarlization coefficient independent of the training size.

With both continuous and discrete decision variables, (6.3) is a Mixed Integer Programming (MIP) problem that is typically highly non-convex and difficult to solve. The recent work [77] has a similar formulation as (6.3). However, [77] introduces the binary codes as auxiliary variables to replace the binary hash functions and has an additional code-fitting term which makes the optimization process more complex and specific to the choice of $\mathcal{L}(\cdot,\cdot)$. In the following, we show that the explicit optimization of the binary codes in (6.3) can be much more efficient and leads to more general solutions.

Specifically, we solve $\mathbf{W}$ and $\mathbf{B}$ alternately by fixing the other. First consider fixing the binary codes $\mathbf{B}$. The optimization with respect to $\mathbf{W}$ becomes a continuous optimization problem that can be conveniently solved using well-established Stochastic Gradient Descent (SGD) (for differentiable loss functions) or subgradient descent (for non-differentiable loss functions) methods, and the update rule for the weight matrix is

$$\mathbf{W} \leftarrow \mathbf{W} - \eta(\nabla\mathcal{L}_n(\mathbf{W}) + \mu\mathbf{W}) \tag{6.4}$$

where $\mathcal{L}_n(\mathbf{W})$ is short for $\mathcal{L}(\mathbf{W}^T\mathbf{b}_n, \mathbf{t}_n)$, '$\nabla$' is the gradient/subgradient operator, and η is the step size. Note that (6.4) can also be replaced with a batch update rule where the update directions are averaged over the gradients/subgradients of a batch of samples.

Now consider fixing $\mathbf{W}$. Problem (6.3) then becomes a binary integer programming (BIP) problem, which is still hard to solve given the $O(2^{L\times N})$ solution space. Motivated by [52], we propose to iteratively solve (6.3) each bit by fixing all the other bits. In fact, $\mathcal{L}(\cdot,\cdot)$ is only a function of the l^{th} bit when fixing all the other bits, and we denote it as $\kappa_{nl}(\cdot)$

$$\kappa_{nl}(b_{nl}) = \mathcal{L}(b_{nl}; \mathbf{b}_{n/l}, \mathbf{W}, \mathbf{t}_n), \tag{6.5}$$

where b_{nl} is the l^{th} bit of the n^{th} sample's binary code, and $\mathbf{b}_{n/l}$ denotes the sample's binary code vector by excluding the l^{th} bit. We have used the subscript nl here to identify $\kappa_{nl}(\cdot)$ since each function is defined with respect to a specific sample and binary bit. Then our problem can be simplified as

$$\min_{b_{nl}\in\{1,-1\}} \sum_{n}^{N} \kappa_l(b_{nl}). \tag{6.6}$$

Here we have omitted the constant terms (i.e. independent of $\mathbf{B}$) and multiplicative coefficients that do not affect the solution to the problem.

With the following proposition, we show that the problem in (6.6) can be written as a general linear BIP problem with a simple analytic solution.

Proposition 1. *For any loss function $\kappa(x)$ defined on the binary input $x \in \{1, -1\}$, there exists a linear function $\gamma(x)$ equal to $\kappa(x)$, and it's defined as*

$$\gamma(x) = \frac{\kappa(1) - \kappa(-1)}{2}x + \frac{\kappa(1) + \kappa(-1)}{2}. \tag{6.7}$$

Proof. The above proposition can be proven by evaluating both functions at all possible inputs. Since there are only two possible inputs 1 and -1, they can be easily verified as follows:

$$\begin{aligned} \gamma(1) &= \frac{\kappa(1) - \kappa(-1)}{2} + \frac{\kappa(1) + \kappa(-1)}{2} = \kappa(1) \\ \gamma(-1) &= -\frac{\kappa(1) - \kappa(-1)}{2} + \frac{\kappa(1) + \kappa(-1)}{2} = \kappa(-1) \end{aligned}$$

This concludes that $\kappa(x) = \gamma(x)$. □

Applying the above proposition to problem (6.6) leads to the following form

$$\begin{aligned} &\min_{\mathbf{b}_{,l}} \mathbf{c}_l^T \mathbf{b}_{,l}, \\ &\text{s.t. } \mathbf{b}_{,l} \in \{-1, 1\}^N, \end{aligned} \tag{6.8}$$

where $\mathbf{b}_{,l}$ is the l^{th} row vector of $\mathbf{B}$, and the constant vector $\mathbf{c}_l$ is defined as

$$\mathbf{c}_l = [\frac{\kappa_{1l}(1) - \kappa_{1l}(-1)}{2}, \cdots, \frac{\kappa_{nl}(1) - \kappa_{Nl}(-1)}{2}]^T. \tag{6.9}$$

The problem in (6.8) has a simple closed-form solution

$$\mathbf{b}_{,l}^* = \text{sgn}(-\mathbf{c}_l) \tag{6.10}$$

In sum, the major steps for learning label preserving binary codes are presented in Algorithm 4. Note that our discussion so far is not tied to any specific loss function, and thus the presented algorithm is a general one. Generally speaking, any proper loss functions can be used in our algorithm, but we only discuss a few commonly used loss functions in the following.

Cross-Entropy Loss The cross-entropy loss is a probabilistic loss function frequently used in classification tasks

$$\mathcal{L}^{(ce)}(\mathbf{y}, \mathbf{t}) = -\sum_i y_i \ln t_i. \tag{6.11}$$

Square Loss The squared loss imposes a squared error on the difference between the prediction and the target and can be used for both classification and regression tasks.

$$\mathcal{L}^{(se)}(\mathbf{y}, \mathbf{t}) = ||\mathbf{y} - \mathbf{t}||_2^2. \tag{6.12}$$

Logistic Loss The logistic loss function measures the degree of fit between the prediction and the target and is mostly used in regression tasks. It's defined as

$$\mathcal{L}^{(lg)}(\mathbf{y}, \mathbf{t}) = \sum_i \log(1 + e^{-y_i t_i}). \tag{6.13}$$

Hinge Loss The hinge loss is typically used for max-margin classification, most notably for SVMs. It's defined as

$$\mathcal{L}^{(hg)}(\mathbf{y}, \mathbf{t}) = \sum_i \max(0, 1 - y_i t_i) \tag{6.14}$$

The gradient/subgradient of those loss functions can be computed easily as follows

$$\begin{aligned}
\nabla \mathcal{L}_n^{(ce)}(\mathbf{W}) &= \mathbf{b} \cdot (\mathbf{t} - e^{\mathbf{t}}/||e^{\mathbf{t}}||_1)^T \\
\nabla \mathcal{L}_n^{(se)}(\mathbf{W}) &= \mathbf{b} \cdot (\mathbf{y} - \mathbf{t})^T \\
\nabla \mathcal{L}_n^{(lg)}(\mathbf{W}) &= \mathbf{b} \cdot (\mathbf{t} \circ \mathrm{I}(\mathbf{y} \circ \mathbf{t} < 1))^T \\
\nabla \mathcal{L}_n^{(hg)}(\mathbf{W}) &= \mathbf{b} \cdot (\mathrm{diag}^{-1}(1 + \mathbf{t}) \cdot \mathbf{y})^T,
\end{aligned} \tag{6.15}$$

where '$\circ$' is the element-wise Hadamard product, $\mathrm{I}(condition)$ is the indicator function that outputs 1 when the condition holds and 0 otherwise, and $\mathrm{diag}(\mathbf{v})$ is the diagonal matrix with $\mathbf{v}$ as its diagonal entries.

6.2.3 Bit Balance Constraints

The bit balance constraint forces each bit position to have an equal number of 1s and -1s (or 0s), and this has been found to be beneficial to the quality of binary codes. As a result, it has been widely used in the hashing literature [87, 59, 103, 38]. With the bit balance constraint, problem (6.3) can be rewritten as

$$\begin{aligned} &\min_{\mathbf{B},\mathbf{W}} \frac{1}{N}\sum_{n}^{N} \mathcal{L}(\mathbf{W}^T\mathbf{b}_n, \mathbf{t}_n) + \frac{\mu}{2}||\mathbf{W}||_F^2, \\ &\text{s.t. } \mathbf{B1} = \mathbf{0}, \\ &\quad \mathbf{B} \in \{-1,1\}^{L\times N}. \end{aligned} \tag{6.16}$$

The first constraint is the bit balance constraint, where $\mathbf{1}$ and $\mathbf{0}$ denote the vector of all ones and zeros respectively. Such a constraint often makes the binary optimization problem more complex. Therefore, some recent hashing methods [77, 54, 46] simply ignore it.

The steps for solving the bit balanced binary optimization problem remain the same except that the sub-problem in (6.8) becomes

$$\min_{\mathbf{b}\in\{-1,1\}^N} g(\mathbf{b}) = \mathbf{c}^T\mathbf{b} + \lambda|\mathbf{1}^T\mathbf{b}|. \tag{6.17}$$

Algorithm 4: Label Preserving Mutimedia Hashing

```
   Input: Data $\mathbf{X}$ and their semantic labels $T$.
   Output: Binary codes $\mathbf{B}$ and weight matrix $\mathbf{W}$.
1  Initialization: Randomly initialize $\mathbf{B}$ and $\mathbf{W}$
2  repeat
3      repeat
4          Estimate the gradient/subgradient of $\mathcal{L}(\cdot,\cdot)$ w.r.t. $\mathbf{W}$ over a sample or a
             mini-batch
5          Update $\mathbf{W}$ in the gradient/subgradient descent direction (e.g. using equation
             (6.4))
6      until convergence
7      repeat
8          for $l = 1, 2, \cdots, L$ do
9              Compute $\mathbf{c}_l$ as defined in (6.9)
10             Solve the $l^{th}$ row of $\mathbf{B}$ according to (6.10)
11         end
12     until convergence
13 until convergence or maximum iteration reached
```

Here $\lambda > 0$ controls the weight of the bit balance constraint, and the resultant hash codes could be perfectly balanced with sufficiently large λ. Note that we have omitted the subscripts to make the following discussion clearer.

Clearly, problem (6.17) no longer enjoys any closed-form solutions, and the complexity of the brute-force search would be $O(2^N)$. In the following proposition, we propose an effective solution to this problem and prove its optimality.

Proposition 2. *There exists an algorithm that finds the optimal solution to problem (6.17) in at most polynomial time.*

Proof. We derive the optimal solution by construction. Specifically, we search for the optimal solution of (6.17) by starting from the solution in (6.10). First consider the case when $\mathbf{b}^*$ is balanced (i.e. $|\mathbf{1}^T\mathbf{b}^*| \leq 1$), then $\mathbf{b}^*$ is also the solution to (6.17) because it minimizes both terms of $g(\mathbf{b})$.

If $\mathbf{b}^*$ is not balanced (i.e. $|\mathbf{1}^T\mathbf{b}^*| > 1$), we need to flip some bits to decrease the second term of $g(\mathbf{b})$. Depending on the sign of $\mathbf{1}^T\mathbf{b}^*$, one may flip positive or negative bits to decrease the value of the second term. For instance, when $\mathbf{1}^T\mathbf{b}^* > 1$, flipping a bit from 1 to -1 would decrease the second term by 2λ. Note that, however, the decrease of the second term doesn't necessarily decrease the overall value of the objective function $g(\mathbf{b})$, because flipping any bit $\hat{b_i}$ of $\hat{\mathbf{b}}$ would increase the value of the first term by $2|c_i|$. Therefore, the net decrease of $g(\mathbf{b})$ caused by flipping the i^{th} bit is

$$\delta_i = 2\lambda - 2|c_i|.$$

In order to minimize $g(\mathbf{b})$, one can simply pick the bit with the maximum δ_i at each step until there doesn't exist any bit that satisfies $\delta_i > 0$, or when the code has become balanced. The optimality of the obtained solution can be verified easily by noticing that $g(\mathbf{b})$ no longer decreases with any bit flips in both stop conditions.

The constructive process above consists of up to $O(N)$ bit flips, and each bit flip involves a max operation (i.e. selecting the maximum δ_i) with $O(N)$ complexity. Therefore, the above algorithm has a $O(N^2)$ time complexity, which concludes the proof. □

While the above proposition alludes to a simple iterative algorithm that flips one bit at a time, the actual implementation can take advantage of the independence among multiple flips to arrive at the optimal solution in one step. The pseudo-algorithm for our implementation is summarized in Algorithm 5. In fact, the complexity of Algorithm 5 is only $O(N)$ because each line can be computed in no more than $O(N)$ time[1].

To enable or disable the bit balance constraints, one can simply switch between equation (6.10) and Algorithm 5 while solving for one row of the code matrix (i.e. line 10 of Algorithm 4). Note that the time complexity of the entire algorithm remains unchanged with the bit balance constraints, as a result of the proposed $O(N)$ solution in Algorithm 5.

Algorithm 5: Solving Balanced Binary Codes

```
   Input: Constant vector $\mathbf{c}$, weight $\lambda$.
   Output: Balanced binary codes $\mathbf{b}$.
1  Initialize $\mathbf{b} \leftarrow \text{sgn}(-\mathbf{c})$
2  Compute the sign of unbalance $s \leftarrow \text{sgn}(\mathbf{1}^T\mathbf{b})$
3  Compute the level of unbalance $m \leftarrow \lceil |\mathbf{1}^T\mathbf{b}|/2 \rceil$
4  Find the set of candidate bits $\mathcal{A} \leftarrow \{b_i | b_i = s \text{ and } |\mathrm{c_i}| < \lambda\}$
5  if $|\mathcal{A}| > m$ then
6      Find $m$ bits with the smallest $c_i$s in $\mathcal{A}$
7      Flip the signs of those $m$ bits
8  else
9      Flip the signs of all the bits in $\mathcal{A}$
10 end
```

6.2.4 Algorithm Complexity

It's not hard to verify that the complexity of the entire discrete optimization algorithm is linear with respect to the training size N. Here we analyze its computational complexity in detail. Solving for $\mathbf{W}$ typically involves going over the training set a number of times, and its complexity

[1]The $O(N)$ algorithm for line 6 is QuickSelect.

can be denoted as $O(I_wN)$, where I_w is the number of sweeps. The complexity for solving the binary code matrix once is $O(LN)$, which leads to the complexity of $O(I_bLN)$ when iterating for I_b times. Therefore, the complexity for one outer loop in Algorithm 4 is $O(I_wN + I_bLN)$. Let I_o be the number of outer iterations, then the overall complexity of Algorithm 4 adds up to $O((I_w+I_bL)I_oN))$, where the typical values of I_w, I_b, and I_o are 20, 2, and 2 respectively. In sum, the linear complexity of the proposed hashing scheme makes the training process highly efficient and scalable, as will be evidenced in the experiments.

6.2.5 Hash Function Learning

We have obtained the hash codes that preserve the semantic label information for the training set $\mathcal{S}$. We also need to learn a set of hash functions, one for each media type, so that new data from different media types can be encoded into a common Hamming space to support cross-media search. In fact, as pointed out by [54], once the hash codes have been obtained, the hash function learning can be modeled as a set of classification problems, with each hash bit corresponding to one binary classifier. Specifically, each bit of the obtained hash codes can be used as the binary label to train a binary classifier, which is open to a wide range of solutions such as SVM, logistic regression, decision trees etc. Similar to [52, 38], here we choose to use boosted decision trees for its testing efficiency and good nonlinear mapping capabilities. Formally, each binary bit is determined by the following hash function

$$h_l(\mathbf{x}) = \mathrm{sgn}\Big(\sum_{t=1}^{T} \alpha_t D_t(\mathbf{x})\Big), \tag{6.18}$$

where T is size of the decision trees ensemble, $D_t : \mathbb{R}^d \mapsto \{-1, 1\}$ is the t^{th} decision tree, and α_t is a function of the error rate of D_t, defined as $\alpha_t = \ln(1/\epsilon_t - 1)$. Note that we have omitted the media-specific superscript for ease of presentation, and the solution is applicable to different media types (i.e. $\mathbf{X}^{(m)}, m = 1, \cdots, M$) to obtain the media specific hash function.

Briefly, a sequence of binary decision trees are learned by adapting the weights of the training samples based on whether they're correctly classified by the previous one. The learning of each decision tree is essentially finding a set of decision stumps, and the testing could be much more efficient than other non-linear mapping methods since it only involves value comparisons. More details of this classic ensemble learning approach can be found in [2].

6.3 Experiments

To evaluate the proposed algorithm, we have conducted extensive experiments on a range of widely used datasets, including both single-media datasets (i.e. CIFAR-10, SVHN and ImageNet-200) and multimedia datasets (i.e. Labelme, MIRFlickr, and NUSWIDE). The details of those datasets are briefly explained as follows.

CIFAR-10[2] . The CIFAR-10 [40] image dataset is a labeled subset of the 80 million tiny images. It consists of 60,000 32×32 color images, each labeled as one of ten object classes. Every image in this dataset is represented by a 512-D GIST feature vector.

Street View House Numbers (SVHN)[3] . The SVHN [66] dataset consists of 630,420 labeled color images of house numbers from Google Street View. The images in this dataset are cropped to 32×32 with digits roughly in the center, and represented by 512-D GIST vectors.

ImagNet-200[4] . The ImageNet-200 is a subset of the ImageNet [25] that contains 200 classes. The number of training, validation and test images in each class are 500, 50 and 50 respectively. We use the fully labeled 110,000 images in our experiments. The 2048-D CNN [31] features are used to represent images in this dataset.

Labelme[5] . The LabelMe dataset contains 2688 images, and each image is described by a few tags. The image-tag pairs are labeled as one of eight unique outdoor scenes, such as 'coast', ‘forest’ and ‘highway’. Images in this dataset are represented by a 512-D GIST vector and the corresponding textual tags are represented by the tag occurrence features (TOF).

MIRFlickr[6] . The MIRFLICKR dataset [32] contains 25,000 image-tag pairs, and each pair is associated with one or more of 24 semantic labels. We remove tags appearing less than 20 times and then discard instances without tags, which leaves 18159 instances remaining. Images and texts are represented by 150-D Edge Histograms (EH) and 1075-D tag occurrence features respectively.

NUS-WIDE[7] . The NUS-WIDE dataset [20] is a real-world multimedia dataset with $269,648$ instances of image-text pairs. Each instance belongs to one or more of 81 concepts. We select the $209,347$ image-tag pairs belonging to the 10 largest concepts. The images and texts are represented by 500-D bag-of-visual-words (BOVW) and 1000-D tag occurrence features respectively.

Table 6.1: Dataset Statistics

	Type	**Features**		**Concepts**	**Size**
Name		Image	Text		
CIFAR-10	Single-media	512D GIST	N/A	10	60000
SVHN	Single-media	512D GIST	N/A	10	99289
ImageNet-200	Single-media	2048D CNN	N/A	200	120000
Labelme	Multimedia	512D GIST	245D TOF	8	2688
MIRFlickr	Multimedia	150D EH	1075D TOF	24	25000
NUS-WIDE	Multimedia	500D BOVW	1000D TOF	81	209347

Since the proposed method handles multiple media types in a homogeneous way, it’s equally optimized for single-media and cross-media retrieval tasks. Therefore, we evaluate its performance on both. Specifically, three types of retrieval tasks are considered in our experiments, that is, “image query image”, “text query image” and “image query text”. As most of the existing hashing methods are either designed for single-media queries or cross-media queries while not optimized for both, we compare with the state-of-the-arts in each setting separately. In detail, we compare with seven single-media hashing methods, including Iterative Quantization (ITQ) [30], Minimal Loss Hashing (MLH) [67], Supervised Hashing with Kernels (KSH) [58], Latent Factor Hashing (LFH) [102], FastHash [52], Supervised Discrete Hashing (SDH) [77] and Column Sampling Discrete Hashing [38]; and seven cross-media hashing methods, including Cross-View Hashing (CVH) [44], Inter-media Hashing (IMH) [80], Latent Semantic Sparse Hashing (LSSH) [110], Collective Matrix Factorization Hashing (CMFH) [26], Semantic Correlation Maximization (SCM) [99], Quantization Correlation Hashing (QCH) [89] and Semantics-Preserving Hashing [54]. Most of the compared algorithms have been briefly introduced in Section 2.1, and the source code for all of the algorithms are publicly available; therefore, we're able to run all of them on the same system settings. In addition, we have set the parameters (if applicable) based on the suggestions of the original papers to obtain the best performance.

We measure the performances of different methods with three widely used retrieval metrics; that is, precision@top-k, precision-recall curves, and mean Average Precision (mAP) [52, 77, 54, 46]. In detail, the top-k precision is the percentage of true neighbors among the k nearest neighbors in terms of Hamming distance; the precision-recall curve is obtained by computing the precision at different recall values; and the mAP is computed as

$$\mathrm{mAP} = \frac{1}{Q}\sum_{q=1}^{Q}\frac{\sum_{r=1}^{R} P_q(r)\delta_q(r)}{\sum_{r=1}^{R}\delta_q(r)}, \tag{6.19}$$

where $P_q(r)$ is the top-r precision of the q^{th} query, $\delta_q(r)$ indicates whether the r^{th} neighbor is a true neighbor of the q^{th} query, and Q is the size of the query set.

The proposed LPMH takes two hyper-parameters, that is, the regularization coefficient μ and the weight of bit balance term λ, and they are fixed to 1 throughout the experiments. The experiments are conducted on a system with THE Intel Xeon E5-2680 CPU and 64 GB of memory.

Table 6.2: Test results of all the methods in terms of mAP and precision@top-100 on the three image datasets. The length of the binary code is varied from 16 bits to 64 bits. The best result for each metric is shown in bold. The proposed LPMH consistently outperforms the baselines in different metrics.

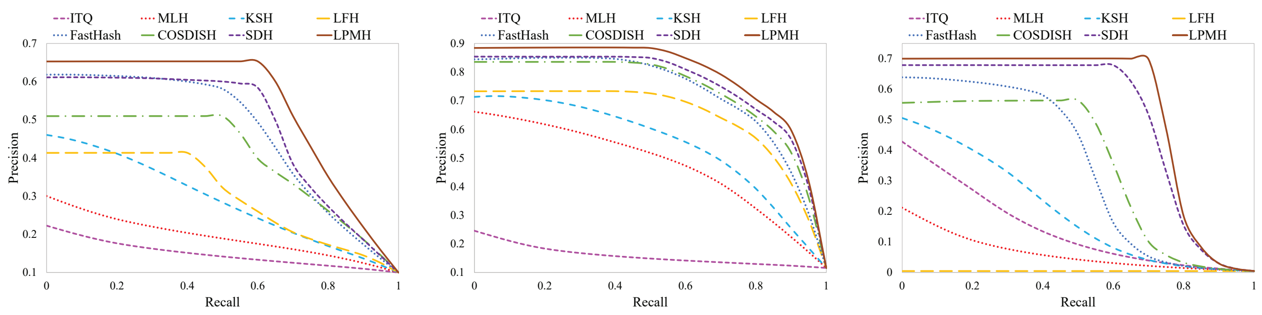
	CIFAR-10			**SVHN**			**ImageNet-200**		
Method	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
				mAP					
ITQ	0.1495	0.1619	0.1780	0.15604	0.1712	0.1927	0.0684	0.1366	0.2049
MLH	0.2036	0.2220	0.2406	0.4600	0.5429	0.5697	0.0396	0.0637	0.0951
KSH	0.3189	0.3420	0.3242	0.5488	0.6224	0.6524	0.1266	0.2063	0.2827
LFH	0.4165	0.5160	0.6137	0.5449	0.7290	0.8231	-	-	-
FastHash	0.5285	0.5957	0.6389	0.7841	0.8344	0.8608	0.2176	0.3652	0.4915
COSDISH	0.5662	0.6020	0.6120	0.8241	0.8572	0.8750	0.1903	0.4140	0.6053
SDH	0.5847	0.6476	0.6859	0.8716	0.8785	0.8828	0.5097	0.6462	0.7093
LPMH	**0.6754**	**0.7217**	**0.7359**	**0.8803**	**0.9017**	**0.9119**	**0.5331**	**0.6594**	**0.7247**
				Precision@Top-100					
ITQ	0.2174	0.2600	0.3115	0.2543	0.3345	0.4278	0.1498	0.2692	0.3676
MLH	0.2804	0.3280	0.3679	0.5836	0.6701	0.7089	0.0699	0.1261	0.1936
KSH	0.3962	0.4558	0.4758	0.6234	0.7172	0.7749	0.2078	0.3444	0.4593
LFH	0.3292	0.4272	0.5288	0.4824	0.6816	0.7848	-	-	-
FastHash	0.5779	0.6225	0.6492	0.8078	0.8427	0.8632	0.3536	0.5194	0.6334
COSDISH	0.8241	0.8572	0.8750	0.7961	0.8293	0.8470	0.1623	0.3846	0.5766
SDH	0.5409	0.6119	0.6224	0.4880	0.6261	0.6791	0.8011	0.6838	0.5895
LPMH	**0.6186**	**0.6611**	**0.6718**	**0.8653**	**0.8839**	**0.8931**	**0.5138**	**0.6405**	**0.7004**

All of the experiment results are averaged over five independent runs unless otherwise specified.

6.3.1 Image Retrieval

We evaluate the image retrieval performance of the proposed LPMH on three large-scale image datasets: CIFAR-10, SVHN and ImageNet-200. For each of the three datasets, we randomly sample the same number of images from each class to form a query set of 2000 images, and the rest are used as the training set and database. Due to the prohibitive pairwise-based training costs, we can only train MLH, KSH, and FastHash on 50k samples. We define the true neighbors of a query to be those sharing the same class label.

The results of the major evaluation metrics are shown in Table 6.2. Note that the reported mAP is computed over all retrieved samples, and thus it's a good indicator of the overall performance of a hashing method. In contrast, the precision of top-100 neighbors is more relevant in scenarios where only the quality of the top returns are concerned. Here we have removed the test results of LFH from ImageNet-200 because the algorithm fails to converge on this dataset, obtaining performance numbers similar to random guesses.

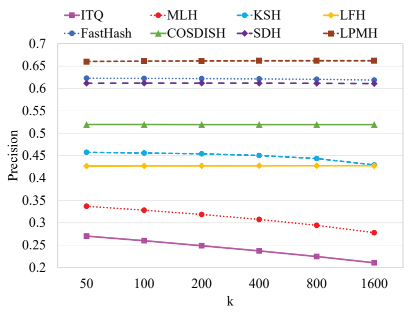


(a) CIFAR-10 (b) SVHN (c) ImageNet-200

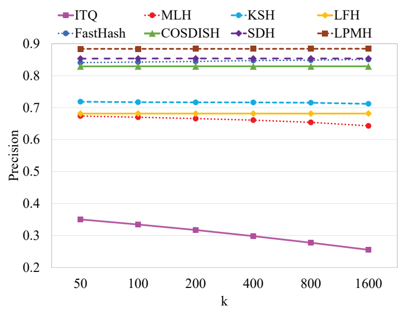
Figure 6.1: Precision-recall curves with 32-bit hash code on three large-scale image datasets. Larger area under the curve indicates better performance. The proposed LPMH achieves the best overall performance.

We can observe from those results that the proposed LPMH is competitive against or superior to the baselines at different code lengths. Additionally, we have plotted the precision-recall curve and the precision with varying number of returned neighbors in Figure 6.1 and Figure 6.2 respectively. Similarly, the proposed LPMH also generates leading performances in those tests. We find that the discrete hashing methods such as FastHash and SDH are generally much better than non-discrete methods such as MLH and KSH, which can be explained by the fact that the continous relaxation used in the non-discrete methods can cause accumulated quantization errors that degrade the quality of resultant hash codes. Such findings are consistent with that of the previous work [52, 77, 59]. Another interesting observation is that classification-based SDH further improves upon the pairwise-based discrete hashing methods (e.g. COSDISH and FastHash), which verifies the effectiveness of classification-based hashing methods in capturing the semantic similarities among training samples [77]. Although both SDH and the proposed LPMH are classification-based hashing methods, the proposed LPMH performs much better in most of the benchmarks, thus demonstrating the superiority of the proposed hash learning framework.

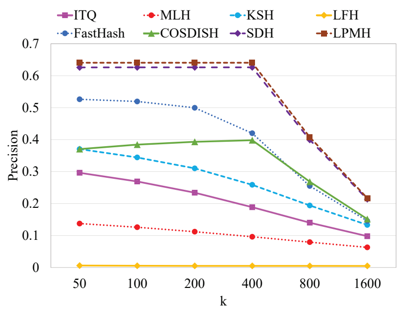
Although the discrete hashing methods typically generate better hash codes, the optimization process can be computationally expensive. In order to test the scalability of the four most competitive discrete hashing methods (i.e. FastHash, COSDISH, SDH and LPMH), we profile the running time of the discrete optimization solvers used in those methods with different code lengths and training sizes. The results of this experiment are shown in Table 6.3. Among the four discrete hashing methods, FastHash takes the most time to run. Specifically, we couldn't finish running FastHash with 100,000 samples at 64 bits within 2 hours. Based on our analysis, the prohibitive complexity of FastHash is mainly caused by the fact that the its graph-cut based formulation on pairwise losses (i.e. the complexity is at least $O(n^2)$) is very computationally expensive.



(a) CIFAR-10



(b) SVHN



(c) ImageNet-200

Figure 6.2: The precision of different methods with varying number of returned neighbors. The results are obtained with 32-bit hash code. The proposed LPMH outperforms all the baselines.

Table 6.3: Running time of the binary optimization solvers used in four discrete hashing methods under different settings. The results are in seconds. Column 2 to 4 show the running time of different code lengths with 50,000 training samples. Column 5 to 7 show the running time to generate 64 bits binary codes with different training sizes. The results of training 64-bit FastHash with 100000 samples are not shown because it couldn't finish running in two hours and we therefore stopped it. The proposed LPMH is much faster than the best baselines, and it scales very well with long codes and large datasets.

	Running Time@50000			**Running Time@64 bits**		
Method	16 bits	32 bits	64 bits	10000	50000	100000
FastHash	846.0	1710.1	3365.4	90.0	3365.4	-
COSDISH	2.6	10.6	44.5	11.2	44.5	90.2
SDH	6.5	6.9	23.8	4.7	23.8	50.9
LPMH	**0.7**	**1.3**	**2.8**	**0.5**	**2.8**	**5.8**

Although COSDISH is also based on a pairwise objective, it adopts a smart column sampling algorithm to avoid dealing with the entire pairwise similarity matrix, thus achieving significant speed up. In comparison, the classification-based hashing methods are inherently easier to train, as demonstrated by the lower running time of SDH and LPMH, especially with larger training size. In particular, we find that the proposed LPMH solver scales very well with the increase of code length and training size. Actually, the running time of LPMH is much faster than the second fastest baseline (i.e. SDH), with up to 8x speed up across the tests, which demonstrates the superior efficiency and scalability of LPMH.

6.3.2 Cross-media Retrieval

The cross-media retrieval experiments are performed on the three widely used [110, 89, 54] multimedia datasets: Labelme, MIRFlickr and NUS-WIDE. We use 80% of the data in Labelme as the training set and database, and the remaining 20% are used as the query set. For MIRFlickr and NUS-WIDE, we randomly sample 2000 image-text pairs as the query set and the remaining data are used as the text and image databases. Additionally, we follow previous works [110, 46] to select 5000 image-text pairs from the database of MIRFlickr and NUS-WIDE as the training set to learn the multimedia hash functions. The learned hash functions are applied to both the database set and the query set to generate the database hash codes and query hash codes. Such practice has been widely adopted [10, 110, 89, 54] to test the out-of-sample extension capabilities of different cross-media hashing algorithms, and it also simulates realworld scenarios where labeled multimedia data is limited. Additionally, we have followed [13, 108, 62] to set R = 50 while computing the mAPs.

Table 6.4: The cross-modal retrieval mAP of the proposed LPMH and compared baselines on three multimodal datasets. The best result of each benchmark is shown in bold. The proposed LPMH outperforms the baselines in almost all the tests.

	Labelme			**MIRFlickr**			**NUS-WIDE**		
Method	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
Text query image									
CVH	0.5482	0.5459	0.5480	0.6389	0.6328	0.6302	0.4466	0.4395	0.4351
IMH	0.5295	0.4737	0.4251	0.6392	0.6384	0.6337	0.4950	0.4975	0.4885
CMFH	0.6881	0.7052	0.7217	0.6534	0.6508	0.6485	0.4960	0.4831	0.4824
LSSH	0.7486	0.7789	0.7884	0.6497	0.6749	0.6897	0.5013	0.5066	0.5299
SCM	0.6834	0.7900	0.8748	0.6692	0.6919	0.6970	0.5281	0.5553	0.5651
QCH	0.8222	0.8264	0.8279	0.6610	0.6994	0.7067	0.5562	0.5584	0.5565
SePH	0.8956	0.9031	0.9147	0.6965	0.7161	0.7407	0.5675	0.5963	0.6253
LPMH	**0.9256**	**0.9288**	**0.9298**	**0.7656**	**0.8128**	**0.8543**	**0.5955**	**0.6519**	**0.6637**
Image query text									
CVH	0.4498	0.4341	0.4117	0.6398	0.6379	0.6334	0.4529	0.4356	0.4250
IMH	0.4925	0.4322	0.3839	0.6542	0.6408	0.6463	0.4657	0.4815	0.4903
CMFH	0.5866	0.6035	0.6123	0.6609	0.6655	0.6618	0.4648	0.4249	0.4087
LSSH	0.7140	0.7399	0.7499	0.6436	0.6589	0.6638	0.5201	0.5318	0.5372
SCM	0.5512	0.6666	0.7484	0.6556	0.6618	0.6682	0.4293	0.4172	0.4619
QCH	0.6951	0.7058	0.7015	0.6677	0.6920	0.6852	0.5295	0.5463	0.5531
SePH	0.7683	0.8108	0.8139	0.6964	0.7206	0.7372	0.5343	0.5497	0.5729
LPMH	**0.8458**	**0.8692**	**0.8663**	**0.7045**	**0.7376**	**0.7642**	**0.5509**	**0.5684**	**0.5837**

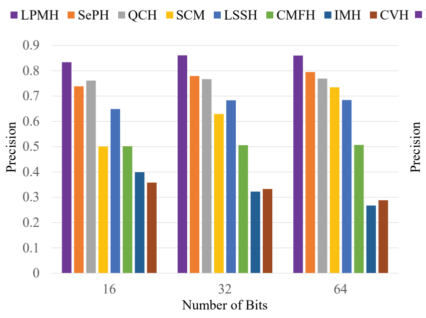
Table 6.5: The test results of all the cross-media hashing algorithms in terms of top-100 precision on the three multimedia datasets. 'T→I' and 'I→T' refer to the results of "text query image" and "image query text" respectively. The 'Mean' column is the average results of 'T→I' and 'I→T'. The hash code is set to 64 bits in this experiment. The proposed LPMH consistently outperforms the baselines in different datasets.

	Labelme			**MIRFlickr**			**NUS-WIDE**		
Method	$T \to I$	$I \to T$	Mean	$T \to I$	$I \to T$	Mean	$T \to I$	$I \to T$	Mean
CVH	0.3529	0.2885	0.3207	0.5917	0.5947	0.5932	0.3789	0.3659	0.3679
IMH	0.2959	0.2674	0.2816	0.6046	0.5695	0.5871	0.4453	0.4379	0.4416
CMFH	0.5823	0.5070	0.5447	0.6119	0.6249	0.6184	0.4334	0.3525	0.3930
LSSH	0.7142	0.6844	0.6993	0.6456	0.6093	0.6275	0.4334	0.3525	0.3930
SCM	0.8317	0.7349	0.7833	0.6705	0.6403	0.6554	0.5312	0.4142	0.4727
QCH	0.7693	0.6803	0.7248	0.6816	0.6590	0.6703	0.5182	0.5144	0.5163
SePH	0.8994	0.7950	0.8472	0.7227	0.6384	0.6806	0.5927	**0.5431**	0.5670
LPMH	**0.9286**	**0.8602**	**0.8944**	**0.8340**	**0.7301**	**0.7821**	**0.6187**	0.5398	**0.5793**

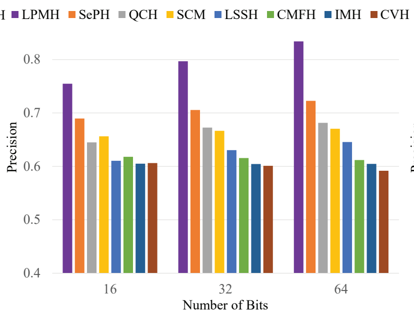
In Labelme, the groudtruth neighbors of each query is defined as instances with the same class label. Since MIRFlickr and NUS-WIDE are multi-label datasets, two instances are defined to be similar if they share at least one common label.

We first compute the mAP of different methods by varying the hash codes from 16 bits to 64 bits, and the results are shown in Table 6.4. It can be noted from the table that the proposed LPMH performs very well on both cross-media retrieval tasks, beating the compared methods in all three datasets. We observe that SePH has also shown strong performances in different tests, which is consistent with the results in previous work [54]. However, the performance of SePH is still secondary to the proposed LPMH, and the improvement of LPMH over SePH can be up to 15%; for instance, in the "text query image" task in MIRFlickr with 64-bit hash code. Note that SePH adopts a similar two-step hashing framework as the proposed LPMH, and therefore the better performance of LPMH can be attributed to the effectiveness of the proposed classification-based discrete optimization strategy. Another interesting observation is that the performances of different methods in “text query image” are typically slightly better than in “image query text”. This can be explained by the fact that there usually exists a gap between feature representations and the semantic concept; and the semantic gap between the low-level image features and the concept are usually much larger than that between the texts and the concept. Similar findings have also been reported in [110, 89, 85].

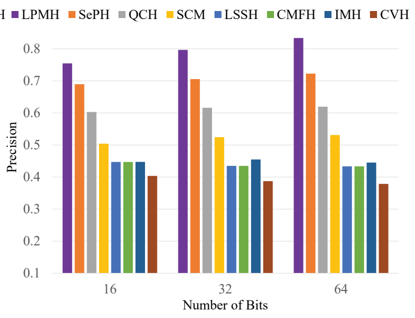
The top-100 precision of different methods are reported in Table 6.5 and Figure 6.3. Specifically, Table 6.5 shows the top-100 precision with 64-bit hash code, while Figure 6.3 shows the precision with a varying number of hash bits. Apparently, the relative performances of different methods are consistent with the mAP tests, with the proposed LPMH leading in most metrics.



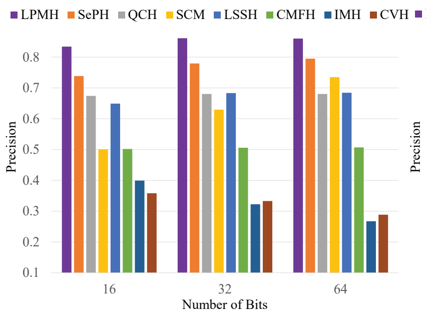
(a) Labelme



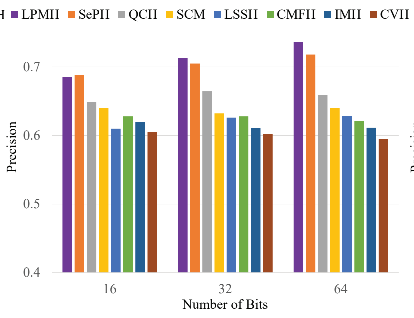
(b) MIRFlickr



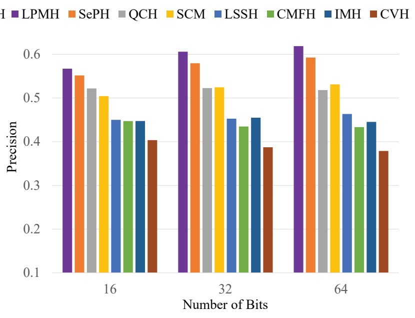
(c) NUS-WIDE



(d) Labelme

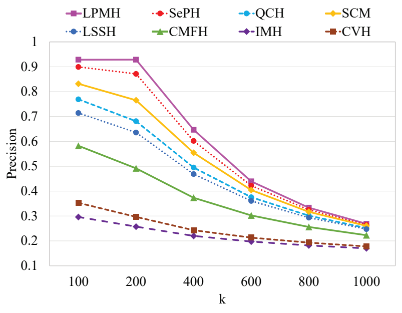


(e) MIRFlickr

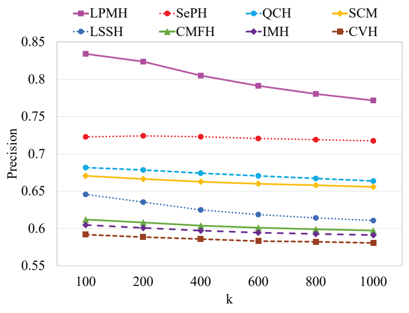


(f) NUS-WIDE

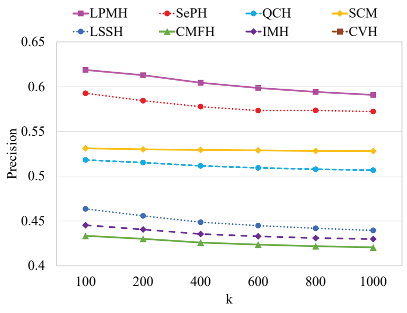
Figure 6.3: The top-100 precision of different cross-media hashing methods on three multimodal datasets. The length of the hash code is varied from 16 bits to 64 bits. The first row shows the “text query image” results and the second row shows the “image query text” results.



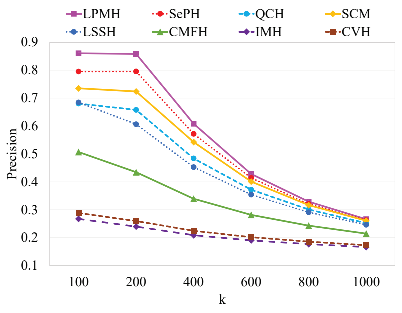
(a) Labelme



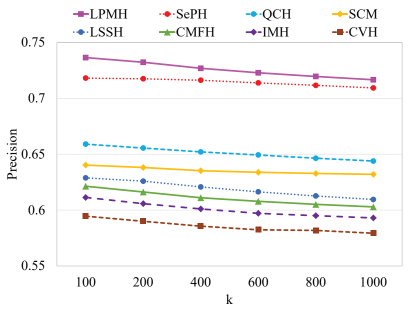
(b) MIRFlickr



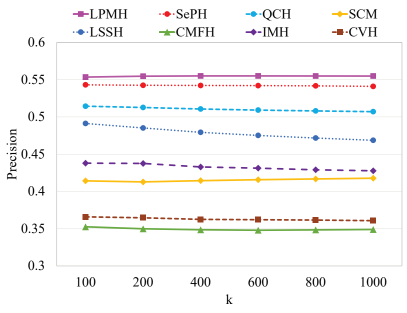
(c) NUS-WIDE



(d) Labelme



(e) MIRFlickr

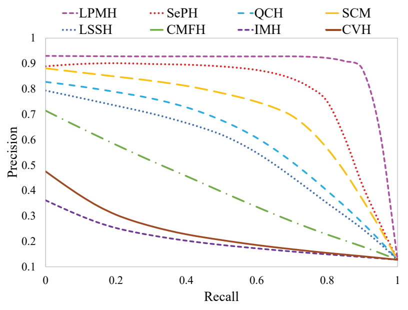


(f) NUS-WIDE

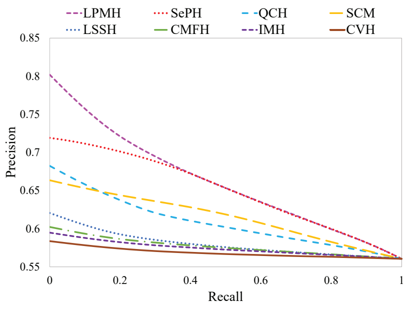
Figure 6.4: The precision of different methods with varying number of returned neighbors. The hash code is set to 64 bits. The top row shows the results of "text query image" and the bottom row shows that of "image query text".

Additionally, we have shown another view of the precision under varying number of returned neighbors in Figure 6.4, as well as the precision-recall curve in Figure 6.5. Similarly, the proposed LPMH demonstrates consistent performance advantages over the compared methods in most of the tests.

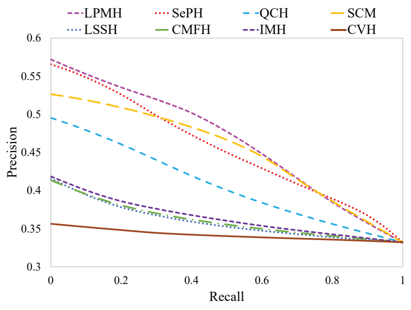
We have also profiled the running time of different methods in Table 6.6. This experiment was performed on the NUS-WIDE dataset with 32-bit hash code. As can be seen from the table, the proposed method can be trained very fast, more than 10 times faster than the other competitive baselines such as SePH and QCH. In terms of test time, most of the compared methods are very fast, with SePH and LSSH being the two exceptions. The slow encoding of SePH and LSSH are caused by the high computational costs of non-linear transformation or matrix inverse operations.



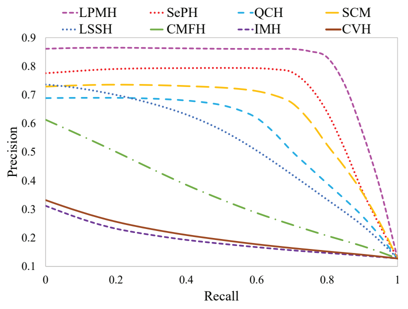
(a) Labelme



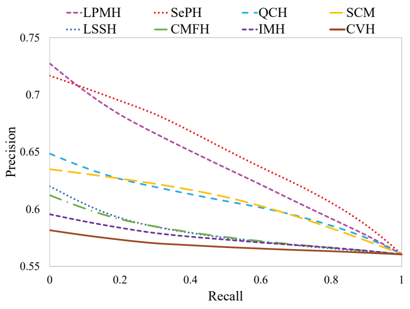
(b) MIRFlickr



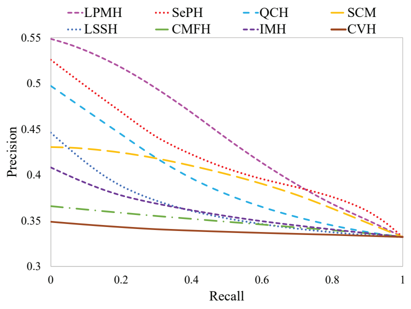
(c) NUS-WIDE



(d) Labelme



(e) MIRFlickr



(f) NUS-WIDE

Figure 6.5: The precision-recall curves of different methods using 64-bit hash codes. The first and second row show the results of "text query image" and "image query text" respectively. Better performance is indicated by larger area under the precision-recall curve. The proposed LPMH performs the best in this metric.

Table 6.6: The training and test time of different cross-media hashing methods on NUS-WIDE. We use 32-bit code in this experiment and the results are shown in seconds. The proposed LPMH can be trained and tested very efficiently.

	Method							
Time (s)	CVH	IMH	CMFH	LSSH	SCM	QCH	SePH	LPMH
Training	1.1	16.8	13.0	181.6	15.0	246.53	250.53	20.1
Test	0.2	0.4	0.3	396.5	0.2	0.2	106.0	0.6

Although the boosted decision tree used in LPMH is also a nonlinear mapping, it only involves value comparisons and therefore could be much more computationally efficient. In order to factor both performance and costs into the comparison, we plot the mAP results against the training times in Figure 6.6. It can be noted that LPMH strikes the best balance between performance and training costs.

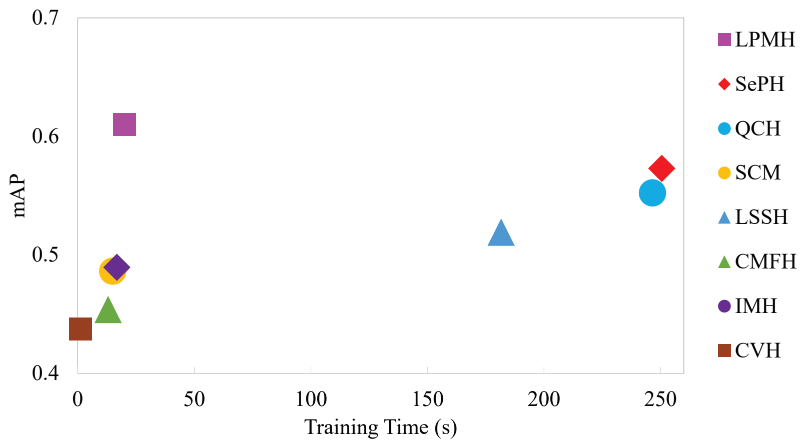


Figure 6.6: The mAP performances of different cross-media hashing against the training time. The results are based on the experiment with 32-bit hash code on the NUS-WIDE dataset. The proposed method strikes the best balance between performance and training costs.

To sum up, the overall good performances of LPMH in different datasets and retrieval tasks as well as the efficiency of training and testing further consolidates the superiority of the proposed multimedia hashing framework.

6.3.3 *Evaluation of Bit Balance Constraint*

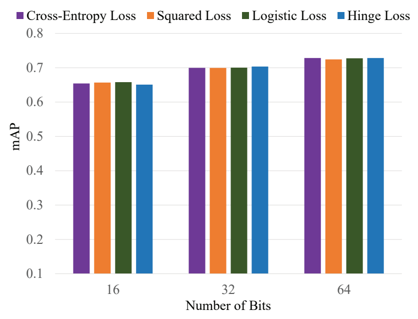
We have also studied the effect of the widely used bit balance constraints on the proposed method, and this experiment was carried out on the CIFAR-10 and NUS-WIDE dataset. The performance results of the proposed method with and without the bit balance constraint are shown in Table 6.7. It can be observed that the bit balance constraint has a positive impact on the quality of the learned binary codes, and better results can be obtained when the constraint is incorporated. We remark that the ability to incorporate bit balance constraints into the binary code learning is one of the advantages over existing discrete hashing methods [57, 52, 77].

Table 6.7: Performances of the proposed method with and without the bit balance constraints on the CIFAR-10 and NUS-WIDE datasets, where 'I$\rightarrow$I', 'I$\rightarrow$T' and 'T$\rightarrow$I' represent "image query image", "text query image" and "image query text" respectively. Better results can be obtained for both single-media retrieval and cross-media retrieval tasks when the bit balance constraint is enabled.

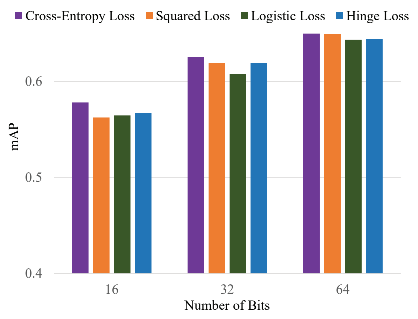
Balance	**CIFAR-10 (I$\rightarrow$I)**			**NUS-WIDE (T$\rightarrow$I)**			**NUS-WIDE (I$\rightarrow$T)**		
Constraints	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
No	0.6457	0.6992	0.7251	0.5846	0.6449	0.6441	0.5458	0.5593	0.5802
Yes	**0.6616**	**0.7053**	**0.7281**	**0.6193**	**0.6474**	**0.6687**	**0.5710**	**0.5746**	**0.5833**

6.3.4 Study of different loss functions

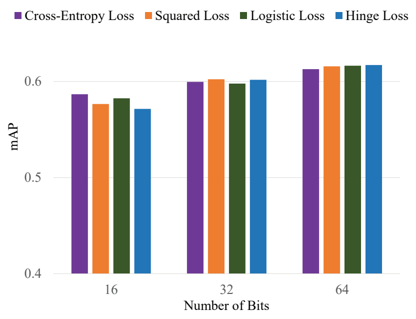
The proposed hashing method is able to accommodate different types of loss functions in a unified discrete optimization framework. Here we evaluate the performance of the proposed method using four different types of loss functions, namely, the cross-entropy loss, squared loss, logistic loss and hinge loss. Those loss functions have been briefly introduced in Section 6.2.2. We perform this experiment on CIFAR-10 and NUS-WIDE, and the results are shown in Figure 6.7. Interestingly, we find that while different loss functions could be advantageous in different scenarios, the performance results of different loss functions are very close in most tests, which demonstrates the robustness and consistency of the proposed optimization method with a range of different loss types. In general, the ability to incorporate different loss functions in a unified optimization framework greatly extends the flexibility of LPMH.



(a) CIFAR-10 (I→I)



(b) NUS-WIDE (T→I)



(c) NUS-WIDE (I→T)

Figure 6.7: The performances of the proposed LPMH with different types of loss functions. The mAP results with different code lengths are reported on CIFAR-10 and NUS-WIDE. 'I→I', 'I→T' and 'T→I' represent "image query image", "text query image" and "image query text" respectively.

6.4 Summary

In this chapter, we present a novel multimedia hashing method, referred to as Linear Label Preserving Multimedia Hashing (LPMH), for large-scale multimedia similarity search. Specifically, we exploit a two-stage discrete hashing framework and propose a general approach for solving binary codes through classification-based optimization objectives. The proposed discrete optimization method is both flexible and efficient: it can accommodate different types of loss functions with minimal changes to the learning steps; it can also be easily combined with the bit balance constraint to obtain highly compact and discriminative binary codes in a much faster speed than existing methods. The experimental results demonstrate the efficiency and effectiveness of LPMH in generating highly discriminative compact hash codes for multimedia retrieval tasks.

CHAPTER 7: CONCLUSION AND FUTURE WORK

7.1 Conclusion

Driven by the pervasiveness of mobile devices and every-increasing popularity of social network websites, the last decade has witnessed the unprecedented growth of multimedia content on the Internet. The ubiquitous multimedia big data presents a number of challenges and opportunities for research and development of efficient storage, indexing, and retrieval techniques. Due to the binary nature of the hash codes, hashing has been widely recognized as a promising solution for these big multimedia data problems. Although the hashing has received great research attentions in the past few years, there are still many research challenges in both theory and applications. This dissertation presents a series of novel research on hashing from both the theoretical and application perspective: in terms of theories, this dissertation has involved studies on random hashing, ranking-based hashing, supervised hashing, and discrete hashing; in terms of applications, the presented algorithms have been applied audio-visual correlation analysis, image retrieval, approximate nearest neighbor search and cross-media similarity search. Specifically, the major contributions of this dissertation are summarized as follows:

- A novel hashing-based audio-visual correlation analysis method has been proposed to solve the problem of audio-visual source localization and segmentation. We present novel audio and visual feature modeling techniques that transform the original problem into audio-visual temporal similarity modeling. Then we apply temporal hashing to the audio and visual representations and solve the problem with hashing-based nearest neighbor search.
- A novel ranking-based supervised hashing algorithm has been proposed for single-media retrieval. Specifically, we explore a new type of hash function based on the ranking of feature subspaces. An effective optimization algorithm was presented to solve the ranking-based hash function learning problem.

- A ranking-based cross-media hashing framework has been proposed to transform multimedia data into a common Hamming space to support cross-media retrieval. We present two alternative optimization algorithms to learn two groups of linear subspaces jointly, one for each modality, such that the ranking ordering in one subspace is maximally aligned with that of the other.

- A novel discrete multimedia hashing method has been proposed for large-scale multimedia similarity search. We exploit a two-stage discrete hashing framework and propose a flexible and efficient discrete optimization approach for learning joint binary codes, that can easily accommodate different types of loss functions and be combined with the bit balance constraint to obtain high-quality binary codes and hash functions.

We have extensively evaluated the proposed methods in public benchmark datasets and compared with the state-of-the-art methods in each specific target domain. Our experimental results have demonstrated the effectiveness of the proposed methods in both qualitative and quantitative evaluation.

7.2 Future Work

Although this dissertation has made significant progress on hashing research, there remain many open research problems of practical use. In this section, we shed light on some of the interesting topics that are worth pursuing for our future research.

The first promising direction is to study deep hashing solutions that learn hash functions in an end-to-end fashion, without the need to use hand-crafted features. In fact, the algorithms introduced in this dissertation have assumed the availability of pre-extracted features, and all the learning procedures are carried out given the fixed feature representations. Inspired by the recent success of deep Convolutional Neural Networks (CNN), it would be promising to extend the methods proposed in this dissertation to CNN-based end-to-end learning hashing methods so that the feature representations and hash functions can be optimized simultaneously in a synergic manner.

Another promising direction is semi-supervised and unsupervised hashing, preferably learned in an end-to-end fashion. The primary focus of this dissertation is on supervised hashing, which is only applicable when data labels are available. However, the labor cost of the manual labeling process is prohibitively expensive, and the labeled data typically constitutes only a very small portion of the available data corpus. To take advantage of the large amount of unlabeled data, it would be preferable to design unsupervised or semi-supervised hashing algorithms that do not require a large number of data labels. Most of the existing unsupervised or semi-supervised hashing techniques, however, are limited in the sense that they require manually crafted features and do not deal with raw data (e.g. image pixels). Therefore, it is an important topic to design end-to-end unsupervised and semi-supervised hashing algorithms.

LIST OF REFERENCES

[1] J.B. Allen. Short term spectral analysis, synthesis, and modification by discrete fourier transform. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 25(3):235–238, Jun 1977.

[2] Ron Appel, Thomas J Fuchs, Piotr Dollár, and Pietro Perona. Quickly boosting decision trees-pruning underachieving features early. In *ICML (3)*, pages 594–602, 2013.

[3] Artem Babenko and Victor Lempitsky. Tree quantization for large-scale similarity search and classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4240–4248, 2015.

[4] Shumeet Baluja and Michele Covell. Learning to hash: forgiving hash functions and applications. *Data Mining and Knowledge Discovery*, 17(3):402–430, 2008.

[5] Z. Barzelay and Y.Y. Schechner. Harmony in motion. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, June 2007.

[6] Matthew J. Beal, Nebojsa Jojic, and H. Attias. A graphical model for audiovisual object tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(7):828–836, July 2003.

[7] Samy Bengio. Multimodal authentication using asynchronous hmms. In *Proceedings of the 4th International Conference on Audio- and Video-based Biometric Person Authentication*, AVBPA'03, pages 770–777, Berlin, Heidelberg, 2003. Springer-Verlag.

[8] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[9] Andrei Z Broder, Moses Charikar, Alan M Frieze, and Michael Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60(3):630 – 659, 2000.

[10] Michael M Bronstein, Alexander M Bronstein, Fabrice Michel, and Nikos Paragios. Data fusion through cross-modality metric learning using similarity-sensitive hashing. In *CVPR, 2010*, pages 3594–3601, 2010.

[11] Yue Cao, Mingsheng Long, Wang Jianmin, Qiang Yang, and Philip Yu. Deep visual-semantic hashing for cross-modal retrieval. In *SIGKDD, 2016*.

[12] Yue Cao, Mingsheng Long, Jianmin Wang, and Han Zhu. Correlation autoencoder hashing for supervised cross-modal search. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, pages 197–204. ACM, 2016.

[13] Yue Cao, Mingsheng Long, Jianmin Wang, and Han Zhu. Correlation autoencoder hashing for supervised cross-modal search. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, pages 197–204. ACM, 2016.

[14] Yue Cao, Mingsheng Long, Jianmin Wang, Han Zhu, and Qingfu Wen. Deep quantization network for efficient image retrieval. In *AAAI*, pages 3457–3463, 2016.

[15] AL. Casanovas, G. Monaci, P. Vandergheynst, and R. Gribonval. Blind audiovisual source separation based on sparse redundant representations. *Multimedia, IEEE Transactions on*, 12(5):358–371, Aug 2010.

[16] AL. Casanovas and P. Vandergheynst. Audio-based nonlinear video diffusion. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 2486–2489, March 2010.

[17] AL. Casanovas and P. Vandergheynst. Unsupervised extraction of audio-visual objects. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 2284–2287, May 2011.

[18] Neal Checka and Kevin Wilson. Person tracking using audio-video sensor fusion. *MIT Artificial Intelligence Laboratory*, 2002, 2001.

[19] Wenlin Chen, James T Wilson, Stephen Tyree, Kilian Q Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *ICML*, pages 2285–2294, 2015.

[20] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. Nus-wide: A real-world web image database from national university of singapore. In *Proceedings of the ACM International Conference on Image and Video Retrieval*, CIVR '09, pages 48:1–48:9, New York, NY, USA, 2009. ACM.

[21] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619, May 2002.

[22] M. Cristani, M. Bicego, and V. Murino. Audio-visual event recognition in surveillance video sequences. *Multimedia, IEEE Transactions on*, 9(2):257–267, Feb 2007.

[23] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262. ACM, 2004.

[24] Thomas Dean, Mark A Ruzon, Mark Segal, Jonathon Shlens, Sudheendra Vijayanarasimhan, and Jay Yagnik. Fast, accurate detection of 100,000 object classes on a single machine. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1814–1821. IEEE, 2013.

[25] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[26] Guiguang Ding, Yuchen Guo, and Jile Zhou. Collective matrix factorization hashing for multimodal data. In *CVPR, 2014*, pages 2083–2090.

[27] J. Driver. Enhancement of selective listening by illusory mislocation of speech sounds due to lip-reading. *Nature*, 381:66–68, 1996.

[28] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. Deep hashing for compact binary codes learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2475–2483, 2015.

[29] Lixin Fan. Supervised binary hash code learning with jensen shannon divergence. In *ICCV, 2013*, pages 2616–2623. IEEE, 2013.

[30] Yunchao Gong and Svetlana Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 817–824. IEEE, 2011.

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[32] Mark J. Huiskes and Michael S. Lew. The mir flickr retrieval evaluation. In *MIR '08: Proceedings of the 2008 ACM International Conference on Multimedia Information Retrieval*, New York, NY, USA, 2008. ACM.

[33] Go Irie, Hiroyuki Arai, and Yukinobu Taniguchi. Alternating co-quantization for cross-modal hashing. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1886–1894, 2015.

[34] H. Izadinia, I Saleemi, and M. Shah. Multimodal analysis for identification and segmentation of moving-sounding objects. *Multimedia, IEEE Transactions on*, 15(2):378–390, Februray 2013.

[35] Qing-Yuan Jiang and Wu-Jun Li. Deep cross-modal hashing. *arXiv preprint arXiv:1602.02255*, 2016.

[36] Wei Jiang, Courtenay Cotton, Shih-Fu Chang, Dan Ellis, and Alexander Loui. Short-term audio-visual atoms for generic video concept classification. In *Proceedings of the 17th ACM International Conference on Multimedia*, MM '09, pages 5–14, New York, NY, USA, 2009. ACM.

[37] Wei Jiang and Alexander C. Loui. Audio-visual grouplet: Temporal audio-visual interactions for general video concept classification. In *Proceedings of the 19th ACM International Conference on Multimedia*, MM '11, pages 123–132, New York, NY, USA, 2011. ACM.

[38] Wang-Cheng Kang, Wu-Jun Li, and Zhi-Hua Zhou. Column sampling based discrete supervised hashing. In *AAAI*, pages 1230–1236, 2016.

[39] E. Kidron, Y.Y. Schechner, and M. Elad. Cross-modal localization via sparsity. *Trans. Sig. Proc.*, 55(4):1390–1404, April 2007.

[40] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[41] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[42] Brian Kulis and Trevor Darrell. Learning to hash with binary reconstructive embeddings. In *Advances in neural information processing systems*, pages 1042–1050, 2009.

[43] Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing for scalable image search. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2130–2137. IEEE, 2009.

[44] Shaishav Kumar and Raghavendra Udupa. Learning hash functions for cross-view similarity search. In *IJCAI, 2011*, volume 22, page 1360.

[45] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. Simultaneous feature learning and hash coding with deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3270–3278, 2015.

[46] Kai Li, Guo-Jun Qi, Jun Ye, and Kien A Hua. Linear subspace ranking hashing for cross-modal retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP(99):1–1, 2016.

[47] Kai Li, Guo-Jun Qi, Jun Ye, Tuoerhongjiang Yusuph, and Kien A Hua. Supervised ranking hash for semantic similarity search. In *Multimedia (ISM), 2016 IEEE International Symposium on*, pages 551–558. IEEE, 2016.

[48] Kai Li, Guo-Jun Qi, Jun Ye, Tuoerhongjiang Yusuph, and Kien A Hua. Semantic image retrieval with feature space rankings. *International Journal of Semantic Computing*, 11(02):171–192, 2017.

[49] Kai Li, Guojun Qi, Jun Ye, and Kien A Hua. Cross-modal hashing through ranking subspace learning. In *Multimedia and Expo (ICME), 2016 IEEE International Conference on*, pages 1–6. IEEE, 2016.

[50] Kai Li, Jun Ye, and Kien A Hua. What's making that sound? In *Proceedings of the ACM International Conference on Multimedia*, pages 147–156. ACM, 2014.

[51] Xi Li, Guosheng Lin, Chunhua Shen, Anton van den Hengel, and Anthony Dick. Learning hash functions using column generation. *arXiv preprint arXiv:1303.0339*, 2013.

[52] Guosheng Lin, Chunhua Shen, Qinfeng Shi, Anton van den Hengel, and David Suter. Fast supervised hashing with decision trees for high-dimensional data. In *CVPR, 2014*, pages 1971–1978. IEEE, 2014.

[53] Guosheng Lin, Chunhua Shen, David Suter, and Anton van den Hengel. A general two-step approach to learning-based hashing. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2552–2559. IEEE, 2013.

[54] Zijia Lin, Guiguang Ding, Mingqing Hu, and Jianmin Wang. Semantics-preserving hashing for cross-view retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3864–3872, 2015.

[55] C. Liu. *Beyond Pixels: Exploring New Representations and Applications for Motion Analysis*. PhD thesis, Massachusetts Institute of Technology, May 2009.

[56] Hong Liu Liu, Ji Rongrong, Wu Yongjian, and Hua Gang. Supervised matrix factorization for cross-modality hashing. In *IJCAI, 2016.*

[57] Wei Liu, Cun Mu, Sanjiv Kumar, and Shih-Fu Chang. Discrete graph hashing. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *NIPS, 2014*, pages 3419–3427. Curran Associates, Inc., 2014.

[58] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. Supervised hashing with kernels. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2074–2081. IEEE, 2012.

[59] Wei Liu, Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Hashing with graphs. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1–8, 2011.

[60] Yuyu Liu and Yoichi Sato. Finding Speaker Face Region by Audiovisual Correlation. In *Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications - M2SFA2 2008*, Marseille, France, 2008. Andrea Cavallaro and Hamid Aghajan.

[61] Yuyu Liu and Yoichi Sato. Visual localization of non-stationary sound sources. In *Proceedings of the 17th ACM International Conference on Multimedia*, MM '09, pages 513–516, New York, NY, USA, 2009. ACM.

[62] Mingsheng Long, Yue Cao, Jianmin Wang, and Philip S Yu. Composite correlation quantization for efficient multimodal retrieval. In *Proceedings of the ACM SIGIR conference on Research and Development in Information Retrieval*, pages 579–588. ACM, 2016.

[63] Jonathan Masci, Michael M Bronstein, Alexander M Bronstein, and Jürgen Schmidhuber. Multimodal similarity-preserving hashing. *TPAMI, 2014.*

[64] Massimo Melucci. On rank correlation in information retrieval evaluation. In *ACM SIGIR Forum*, volume 41, pages 18–33. ACM, 2007.

[65] Sean Moran and Victor Lavrenko. Regularised cross-modal hashing. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 907–910. ACM, 2015.

[66] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5, 2011.

[67] Mohammad Norouzi and David J. Fleet. Minimal loss hashing for compact binary codes. In *ICML, 2011)*, pages 353–360, 2011.

[68] Mohammad Norouzi, David J Fleet, and Ruslan Salakhutdinov. Hamming distance metric learning. In *Advances in Neural Information Processing Systems*, pages 1061–1069, 2012.

[69] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3):145–175, 2001.

[70] Mingdong Ou, Peng Cui, Fei Wang, Jun Wang, Wenwu Zhu, and Shiqiang Yang. Comparing apples to oranges: a scalable solution with heterogeneous hashing. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 230–238. ACM, 2013.

[71] Yanwei Pang, Zhong Ji, Peiguang Jing, and Xuelong Li. Ranking graph embedding for learning to rerank. *Neural Networks and Learning Systems, IEEE Transactions on*, 24(8):1292–1303, Aug 2013.

[72] Novi Quadrianto and Christoph H Lampert. Learning multi-view neighborhood preserving projections. In *ICML, 2011*, pages 425–432.

[73] N. Rasiwasia, J. Costa Pereira, E. Coviello, G. Doyle, G.R.G. Lanckriet, R. Levy, and N. Vasconcelos. A New Approach to Cross-Modal Multimedia Retrieval. In *ACM MM, 2010*, pages 251–260, 2010.

[74] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic hashing. *RBM*, 500(3):500, 2007.

[75] John Savard. http://members.shaw.ca/quadibloc/other/colint.htm.

[76] Gregory Shakhnarovich, Paul Viola, and Trevor Darrell. Fast pose estimation with parameter-sensitive hashing. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 750–757. IEEE, 2003.

[77] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. Supervised discrete hashing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 37–45, 2015.

[78] Qinfeng Shi, Hanxi Li, and Chunhua Shen. Rapid face recognition using hashing. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2753–2760. IEEE, 2010.

[79] C. Sigg, B. Fischer, B. Ommer, V. Roth, and J. Buhmann. Nonnegative cca for audiovisual source separation. In *Machine Learning for Signal Processing, 2007 IEEE Workshop on*, pages 253–258, Aug 2007.

[80] Jingkuan Song, Yang Yang, Yi Yang, Zi Huang, and Heng Tao Shen. Inter-media hashing for large-scale retrieval from heterogeneous data sources. In *ACM SIGMOD, 2013*, pages 785–796.

[81] Mohammad Mehdi Homayounpour Vahid Asadpour and Farzad Towhidkhah. Audio-visual speaker identification using dynamic facial movements and utterance phonetic content. *Applied Soft Computing*, 11(2):2083–2093, 2011.

[82] Andrea Vedaldi and Andrew Zisserman. Sparse kernel approximations for efficient classification and detection. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2320–2327. IEEE, 2012.

[83] Luis Von Ahn, Ruoran Liu, and Manuel Blum. Peekaboom: a game for locating objects in images. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 55–64. ACM, 2006.

[84] Daixin Wang, Peng Cui, Mingdong Ou, and Wenwu Zhu. Deep multimodal hashing with orthogonal regularization. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 2291–2297. AAAI Press, 2015.

[85] Di Wang, Xinbo Gao, Xiumei Wang, and Lihuo He. Semantic topic multimodal hashing for cross-media retrieval. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 3890–3896, 2015.

[86] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Semi-supervised hashing for large-scale search. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(12):2393–2406, 2012.

[87] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *Advances in neural information processing systems*, pages 1753–1760, 2009.

[88] Kevin Wilson, Neal Checka, David Demirdjian, and Trevor Darrell. Audio-video array source separation for perceptual user interfaces. In *Proceedings of the 2001 Workshop on Perceptive User Interfaces*, PUI '01, pages 1–7, New York, NY, USA, 2001. ACM.

[89] Botong Wu, Qiang Yang, Wei-Shi Zheng, Yizhou Wang, and Jingdong Wang. Quantized correlation hashing for fast cross-modal search. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[90] Fei Wu, Zhou Yu, Yi Yang, Siliang Tang, Yin Zhang, and Yueting Zhuang. Sparse multimodal hashing. *IEEE Transactions on Multimedia*, 16(2):427–439, 2014.

[91] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan. Supervised hashing for image retrieval via image representation learning. In *AAAI*, volume 1, page 2, 2014.

[92] Yan Xia, Kaiming He, Pushmeet Kohli, and Jian Sun. Sparse projections for high-dimensional binary codes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3332–3339, 2015.

[93] Jay Yagnik, Dennis Strelow, David A. Ross, and Ruei-sung Lin. The power of comparative reasoning. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 2431–2438, Washington, DC, USA, 2011. IEEE Computer Society.

[94] Jay Yagnik, Dennis Strelow, David A Ross, and Ruei-sung Lin. The power of comparative reasoning. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2431–2438. IEEE, 2011.

[95] J. Ye, K. Li, and K. A. Hua. Wta hash-based multimodal feature fusion for 3d human action recognition. In *2015 IEEE International Symposium on Multimedia (ISM)*, pages 184–190, Dec 2015.

[96] Xiang Yu, Shaoting Zhang, Bo Liu, Lin Zhong, and Dimitris N. Metaxas. Large scale medical image search via unsupervised pca hashing. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2013.

[97] Deming Zhai, Hong Chang, Yi Zhen, Xianming Liu, Xilin Chen, and Wen Gao. Parametric local multimodal hashing for cross-view similarity search. In *IJCAI, 2013*, pages 2754–2760.

[98] Dan Zhang, Fei Wang, and Luo Si. Composite hashing with multiple information sources. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 225–234. ACM, 2011.

[99] Dongqing Zhang and Wu-Jun Li. Large-scale supervised multimodal hashing with semantic correlation maximization. In *AAAI, 2014*.

[100] Hanwang Zhang, Fumin Shen, Wei Liu, Xiangnan He, Huanbo Luan, and Tat-Seng Chua. Discrete collaborative filtering. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 325–334. ACM, 2016.

[101] Hanwang Zhang, Meng Wang, Richang Hong, and Tat-Seng Chua. Play and rewind: Optimizing binary representations of videos by self-supervised temporal hashing. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 781–790. ACM, 2016.

[102] Peichao Zhang, Wei Zhang, Wu-Jun Li, and Minyi Guo. Supervised hashing with latent factor models. In *Proceedings of the ACM SIGIR conference on Research & development in information retrieval*, pages 173–182. ACM, 2014.

[103] S. Zhang, J. Li, J. Guo, and B. Zhang. Scalable discrete supervised hash learning with asymmetric matrix factorization. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 1347–1352, Dec 2016.

[104] Ting Zhang, Guo-Jun Qi, Jinhui Tang, and Jingdong Wang. Sparse composite quantization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4548–4556, 2015.

[105] Ting Zhang and Jingdong Wang. Collaborative quantization for cross-modal similarity search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2036–2045, 2016.

[106] Ziming Zhang, Yuting Chen, and Venkatesh Saligrama. Efficient training of very deep neural networks for supervised hashing. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[107] Fang Zhao, Yongzhen Huang, Liang Wang, and Tieniu Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR2015, Boston, MA, USA, June 7-12, 2015*, pages 1556–1564, 2015.

[108] Yi Zhen and Dit-Yan Yeung. Co-regularized hashing for multimodal data. In *NIPS, 2012*, pages 1376–1384.

[109] Yi Zhen and Dit-Yan Yeung. A probabilistic model for multimodal hash function learning. In *SIGKDD, 2012*.

[110] Jile Zhou, Guiguang Ding, and Yuchen Guo. Latent semantic sparse hashing for cross-modal similarity search. In *ACM SIGIR, 2014*, pages 415–424.

[111] Xiaofeng Zhu, Zi Huang, Heng Tao Shen, and Xin Zhao. Linear cross-modal hashing for efficient multimedia search. In *ACM MM, 2013*, pages 143–152.

[112] Yueting Zhuang, Zhou Yu, Wei Wang, Fei Wu, Siliang Tang, and Jian Shao. Cross-media hashing with neural networks. In *ACM MM, 2014*, pages 901–904.

[113] Dmitry N. Zotkin, Ramani Duraiswami, and Larry S. Davis. Joint audio-visual tracking using particle filters. *EURASIP J. Appl. Signal Process.*, 2002(1):1154–1164, January 2002.