ABSTRACT

Title of dissertation: LEARNING BINARY CODE REPRESENTATIONS FOR EFFECTIVE AND EFFICIENT IMAGE RETRIEVAL

Bahadir Ozdemir, Doctor of Philosophy, 2016

Dissertation directed by: Professor Larry S. Davis
Department of Computer Science

The size of online image datasets is constantly increasing. Considering an image dataset with millions of images, image retrieval becomes a seemingly intractable problem for exhaustive similarity search algorithms. Hashing methods, which encodes high-dimensional descriptors into compact binary strings, have become very popular because of their high efficiency in search and storage capacity.

In the first part, we propose a multimodal retrieval method based on latent feature models. The procedure consists of a nonparametric Bayesian framework for learning underlying semantically meaningful abstract features in a multimodal dataset, a probabilistic retrieval model that allows cross-modal queries and an extension model for relevance feedback.

In the second part, we focus on supervised hashing with kernels. We describe a flexible hashing procedure that treats binary codes and pairwise semantic similarity as latent and observed variables, respectively, in a probabilistic model based on Gaussian processes for binary classification. We present a scalable inference algo-

rithm with the sparse pseudo-input Gaussian process (SPGP) model and distributed computing.

In the last part, we define an incremental hashing strategy for dynamic databases where new images are added to the databases frequently. The method is based on a two-stage classification framework using binary and multi-class SVMs. The proposed method also enforces balance in binary codes by an imbalance penalty to obtain higher quality binary codes. We learn hash functions by an efficient algorithm where the NP-hard problem of finding optimal binary codes is solved via cyclic coordinate descent and SVMs are trained in a parallelized incremental manner. For modifications like adding images from an unseen class, we propose an incremental procedure for effective and efficient updates to the previous hash functions. Experiments on three large-scale image datasets demonstrate that the incremental strategy is capable of efficiently updating hash functions to the same retrieval performance as hashing from scratch.

LEARNING BINARY CODE REPRESENTATIONS FOR EFFECTIVE AND EFFICIENT IMAGE RETRIEVAL

by

Bahadir Ozdemir

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2016

Advisory Committee:
Professor Larry S. Davis, Chair/Advisor
Professor Şennur Ulukuş, Dean's Representative
Professor Ramani Duraiswami
Professor Hal Daumé III
Professor Héctor Corrada Bravo

Acknowledgments

I would like to express my sincere thanks to my advisor, *Larry Davis*, for his guidance, suggestions, and support throughout the development of this thesis. He encouraged me to develop and explore my research ideas for the image retrieval problem while helping each step with his great vision and breadth of knowledge. Whenever I got stuck in details, he provided me a different viewpoint. Working with him has been a valuable experience for me.

I would like to extend my thanks to the dissertation committee members – *Şennur Ulukuş*, *Ramani Duraiswami*, *Hal Daumé III* and *Héctor Corrada Bravo* – for agreeing to serve on my dissertation committee and reviewing this thesis despite their busy schedules.

My special thanks must be sent to *Fatoş Tünay Yarman-Vural* who introduced me to computer vision and my previous advisor *Selim Aksoy* who introduced me to the world of research. I would like to thank *Hal Daumé III*, *Jordan Boyd-Graber* and *Naomi Feldman* who introduced me Bayesian nonparametrics, which have been studied in this dissertation.

I would like to thank my mentors in the bioinformatics project as part of the NCI-UMD partnership program, *Wael Abd-Almageed*, *Stephanie Roessler* and *Xin Wei Wang*. Developing myself in the field of machine learning during that project led to the development of advanced probabilistic models in this thesis. I would also like to thank *Luís Sarmento* for mentoring me during my internship at Amazon. During the internship, I was exposed to challenging real world retrieval system

problems. It was a fruitful experience for me to learn optimization algorithms and high-performance computing which I benefit from for the development of inference algorithms in this dissertation.

I am very grateful to my friends outside the lab with whom I was having nice days during my Ph.D. – *Yunus*, *Adil*, *Tuğrul*, *Şimal*, *Ferhan*, *Elif*, *Nilsu*, *Besim* – and my labmates with whom I had several interesting discussions – *Sonya*, *Guangxiao*, *Varun*, *Sravanthi*, *Philip*, *Choi*, *Mohammad* and *Mahyar*.

Lastly but most importantly, I would like to express my deepest gratitude to my family, always standing by me, for their endless support and understanding. This thesis is dedicated to them.

Bahadır Özdemir, May 5, 2016

College Park, Maryland

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

First, writing a thesis should be fun.

"How to Write a Thesis" – Umberto Eco

1.1 Overview

As the size of online image datasets like Flickr is constantly increasing due to rapid advances in digital camera technology, image processing tools, and photo sharing platforms, the problem of image retrieval has attracted more attention from researchers in computer vision, machine learning, and information retrieval. Massive amounts of information lead to the requirement of efficient search algorithms. Considering an image dataset with millions of images, image retrieval becomes a seemingly intractable problem for exhaustive similarity search algorithms due to their linear time complexity. Although k-d trees are useful data structures for low-dimensional nearest neighbor search, they are not as applicable to high-dimensional image descriptors. We can reduce the linear time complexity of exhaustive similarity search to sublinear time complexity for approximate nearest neighbor search with the help of discretization [2]. Therefore, encoding high-dimensional descriptors into compact binary strings has become a very popular representation for images because of

their high efficiency in query processing and storage capacity [2, 3, 4, 5]. Similarity-preserving binary codes have received significant attention for image search and retrieval in large-scale image collections [6, 7].

In this thesis, we state that *utilizing text or label data in an incremental learning strategy makes binary codes effective and efficient representations for image retrieval.*

1.2 Motivation

Despite the increasing amount of multimodal data, especially in multimedia domains *e.g.* images with captions or tags, most existing hashing techniques, unfortunately, focus on unimodal data. Hence, they inevitably suffer from the semantic gap, which is defined in [8] as the lack of coincidence between low-level visual features and high-level semantic interpretation of an image. A holistic approach that strives the integration of information from various input modalities into computational models contributes towards improved search and retrieval capabilities. However, it also poses challenges associated with handling cross-view similarities and interactions.

Other alternatives for bridging the semantic gap are supervised hashing methods that use semantic class labels in learning hash functions. However, overfitting to training data and long training time are two common problems with those methods. Bayesian models with model choice or averaging are promising approaches to prevent overfitting when a scalable inference algorithm is available. On the other

hand, considering online image databases where new images are added steadily every day, hash functions need to be updated as the database becomes larger with new pictures. Besides, those changes may emerge new semantic classes. As a result, existing hashing methods should be trained on the final data from scratch. In this aspect, a hashing approach that learns binary codes incrementally offers an efficient solution to dynamic image datasets.

1.3 Problem Definition

In the first part, we focus on unsupervised multimodal retrieval and processing user feedback [9]. Although many hashing approaches rely on supervised information like semantic class labels, class memberships are not available for many image datasets. Also, many supervised methods cannot be generalized to unseen classes that are not used during training [10]. Besides, every user's need is different and time-varying [11]. Therefore, user judgments indicating the relevance of an image retrieved for a query are utilized to achieve better retrieval performance in the revised ranking of images [12]. Development of an efficient retrieval system that embeds information from multiple domains into short binary codes and takes relevance feedback into account is quite challenging. We consider some hidden common causes explain the dependency among modalities. We propose a multimodal image retrieval procedure by integrating visual features with tags or descriptive texts. The process first identifies underlying semantically meaningful abstract elements in a multimodal dataset using a nonparametric Bayesian framework based on a la-

tent feature model. In retrieval phase, these abstract features are employed in a probabilistic model that allows cross-modal queries. Finally, we use an extension model for relevance feedback to achieve better retrieval performance by utilizing user relevance judgments.

In the second part, we deal with the problem of supervised hashing [13]. Overfitting is a general issue of this type of hashing methods. Despite different formulations, many supervised hashing methods in practice find one binary string pattern for each class and the same binary hash code is assigned to all images in that class. We use a fully probabilistic approach with model averaging for learning binary codes from data. We utilize the Gaussian process classification (GPC) model [14] for supervised hashing in a Bayesian framework. In our model, the variables corresponding to the binary classes in the original GPC model become latent variables and a new level of variables that correspond to the pairwise similarity between data points is integrated into the design. Averaging over nonlinear classification models by Gaussian processes is our motivation behind the GPH model to overcome the overfitting problem of supervised hashing. Unfortunately, Gaussian processes are computationally heavy models with $\mathcal{O}(n^3)$ time complexity. Therefore, we developed a scalable inference algorithm based on a sparse approximation with distributed computing and scalable expectation propagation in a stochastic fashion.

In the last part, we address the dynamic hashing problem [15]. Despite the fact that new images are added to online image datasets every day, to best of our knowledge, no supervised hashing method learns hash functions incrementally for newly added images. Efficiently updating hash functions concerning new images

is a challenging task. In addition, some of the new images will possibly not belong to existing classes *i.e.* forming new semantic classes, that makes this task even more challenging. Existing supervised hashing methods cannot be easily generalized to unseen classes. Such modifications to a database require recomputation of hash functions from scratch whenever a change occurs in a dynamic dataset, which is computationally intractable. We propose an incremental supervised hashing method with kernels based on binary and multi-class SVMs, and we adopt the incremental learning fashion of support vector machines (SVM) in a hashing framework. Consequently, the proposed hashing method can be easily extended to unseen classes incrementally.

1.4 Datasets

In this study, we used five different image datasets in total, namely the Pascal-Sentence, SUN-Attribute, CIFAR-10, MNIST and NUS-WIDE datasets. Each dataset is explained in a separate section as follows.

1.4.1 Pascal-Sentence Dataset

The PASCAL-Sentence 2008 dataset is formed from the PASCAL 2008 images by randomly selecting 50 images belonging to each of the 20 categories [16]. In the experiments, we used the precomputed visual and textual features provided by Rastegari *et al.* [17]. Amazon Mechanical Turk workers annotate five sentences for each of the 1000 images. Each image is labeled by a triplet of $<$*object, action,*

scene> representing the semantics of the picture from these sentences. For each image, the semantic similarity between each word in its triplet and all words in a dictionary constructed from the entire dataset is computed by the Lin similarity measure [18] using the WordNet hierarchy. The textual features of an image are the sum of all similarity vectors for the words in its triplet. Visual features are built from various object detectors, image classifiers and scene classifiers. These features contain the coordinates and confidence values that object detectors fire and the responses of image and scene classifiers trained on low-level image descriptors. Figure 1.1 presents sample images and corresponding image descriptions by Amazon Turk workers.

1.4.2 SUN-Attribute Dataset

The SUN-Attribute dataset [19], a large-scale dataset of attribute-labeled scenes, is built on top of the existing SUN categorical dataset [20]. The dataset contains 102 attribute labels annotated by three Amazon Mechanical Turk workers for each of the 14,340 images from 717 categories. Each class has 20 annotated images. The precomputed visual features [19, 20] include gist, 2×2 histogram of oriented gradient, self-similarity measure, and geometric context color histograms. The attribute features are computed by averaging the binary labels from multiple annotators where each image is annotated with attributes from five types: materials, surface properties, functions or affordances, spatial envelope attributes and object presence. Sample images are visualized on a projected two-dimensional semantic

	• A green and gray plane is taking off from the runway. • A green and white cargo plane taking of from an airport. • A green and white jet taking off. • A Jade Cargo jet in green and white taking off from an airport. • White and green commercial airliner taking off of runway.
	• A man on a mountain bike going down an incline. • A mountain biker riding over a small stone ridge. • A person on a bicycle rides on a rocky path. • Man on mountain bike coming off of small rocky ledge. • This is a man riding a mountain bike over rocks on a clear day.
	• A bird carrying a branch over the water. • A bird flies over water carrying a branch in its talons. • A large bird flying across the water. • An osprey flies over water with a stick. • The hawk is flying with a stick in his talons.
	• A cruiseliner docked at a port • A cruise ship is in front of a docking area. • A docked cruise ship. • The cruise ship docks to possibly load new passengers. • White cruise ship floating on the water.
	• A 4-H booth with posters, television, and three people presenting near a laptop. • A presenting being interviewed • People pose with a microphone in front of a display. • The people are in front of the 4-H display. • Vendors work their booth at the trade show.
	• A closet door stands open next to a marble-top counter. • A kitchen sink and countertop with bowls on shelves. • A kitchen with granite countertops and tiled flooring. • A small kitchen with items stacked on the shelves and on the counter. • A small tiled kitchen area.
	• A large white "Victory Liner" bus with red and yellow trim is in a parking lot. • A side view of a passenger bus. • A Victory Liner bus is white with red and gold stripes. • A white tour bus with red, orange and yellow stripes. • The bus has a red, orange and yellow design on it.
	• A group of cows in a field with yellow tags in their ears. • Group of several cows standing close together. • The tagged black cows gather for a group picture. • Three cows looking at the screen with one cow in the background. • Three cows staring forward and one in the background.

Figure 1.1: Each row shows an image from one class in the Pascal-Sentence dataset and its description sentences by Amazon Turk workers.

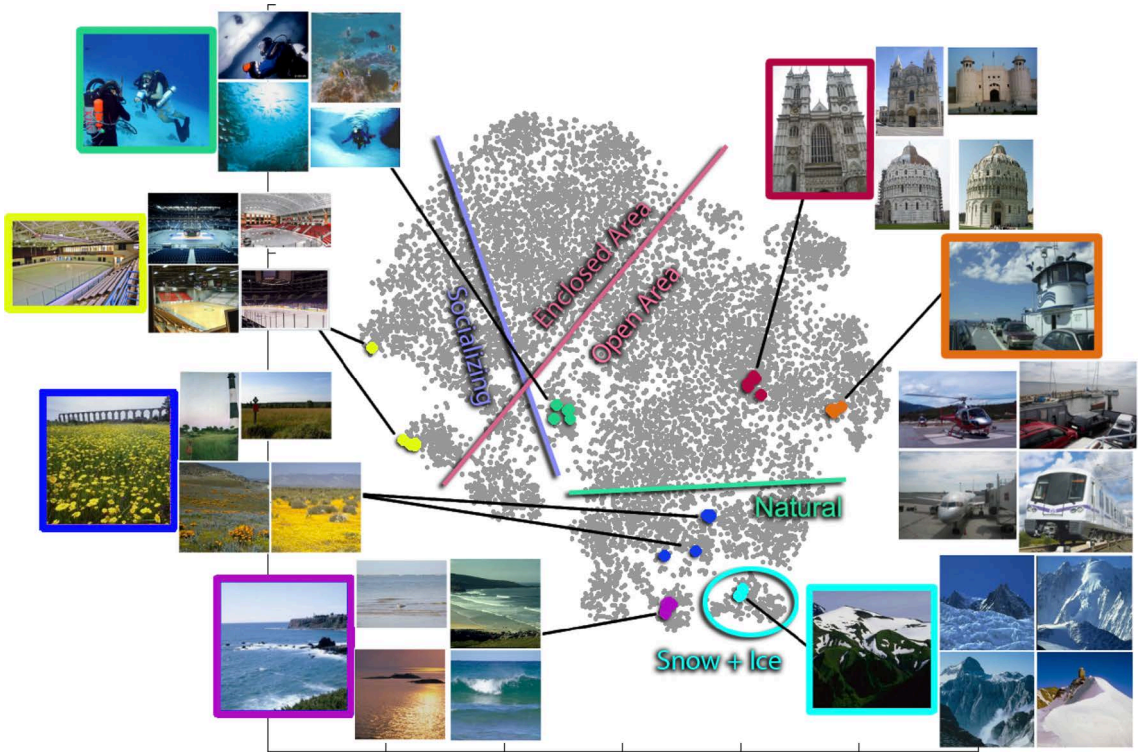


Figure 1.2: Sample images from the SUN-Attribute dataset are visualized on a two-dimensional space which is constructed by the projection of its 102-dimensional attribute feature vector onto two dimensions (This figure is taken from [1]).

space in Figure 1.2.

1.4.3 CIFAR-10 Dataset

The CIFAR-10 dataset [21], which is a labeled subset of the 80M tiny images dataset, has 60,0000 images from 10 classes of vehicles and animals such as airplane, bird, frog, and truck. There are 50,000 training and 10,000 test images in the dataset. We used a GIST descriptor [22] of 512 dimensions to represent each image. Random images from each class in the CIFAR-10 dataset are shown in Figure 1.3.

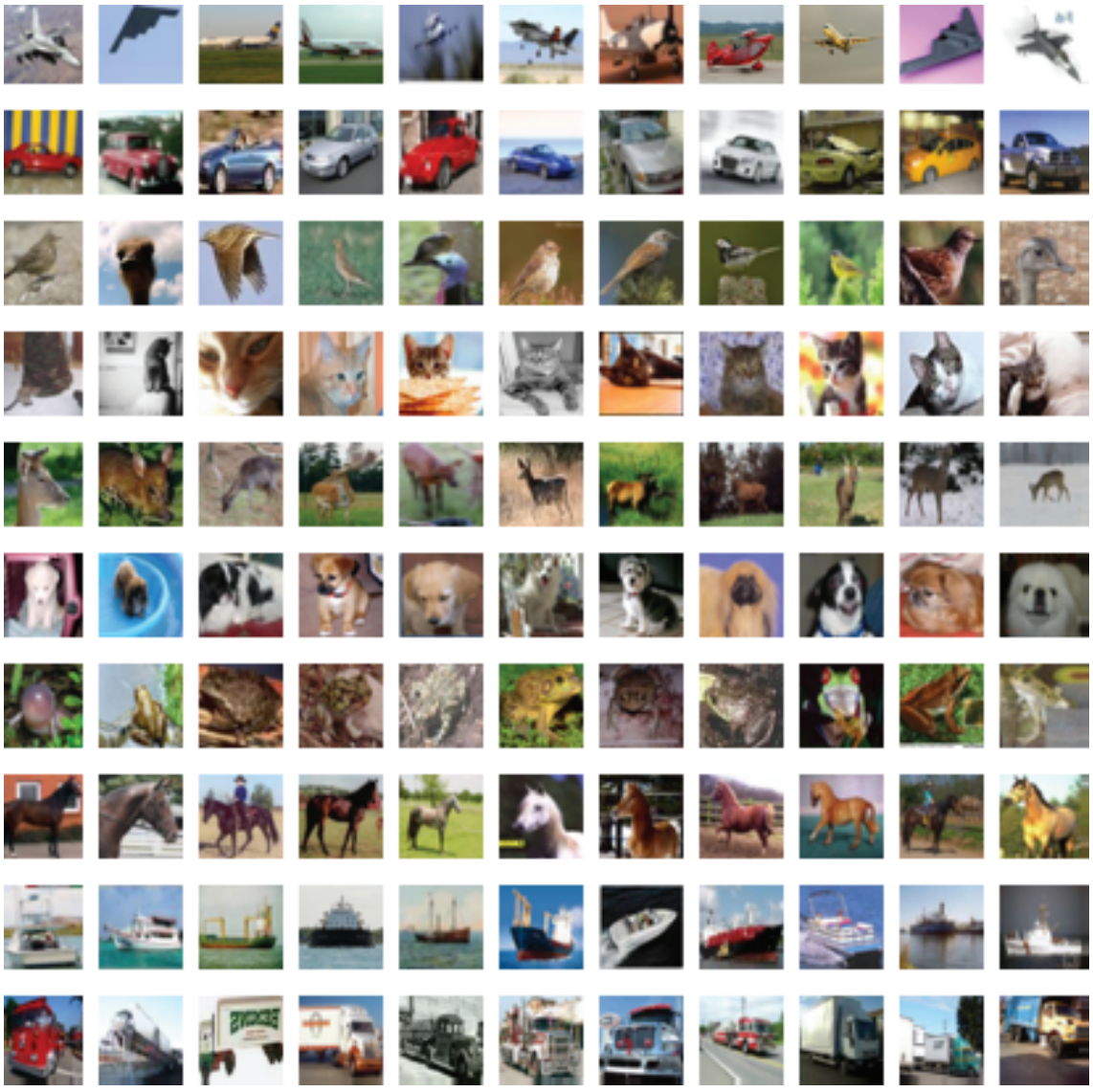


Figure 1.3: Each row shows sample images from one class in the CIFAR-10 dataset.

1.4.4 MNIST Dataset

The MNIST dataset [23] consists of 28×28 pixel images of handwritten digits from '0' to '9'. The dataset is split into a training set of 60,000 examples and a test set of 10,000 examples. We used the pixel data in our experiments. Random images from each class in the CIFAR-10 dataset can be seen in Figure 1.4.

1.4.5 NUS-WIDE Dataset

The NUS-WIDE dataset [24] includes 269,648 images and associated semantic labels of 81 concepts from Flickr such as dog, flower, street, and dancing. The dataset is split into 161,789 training images and 107,859 test images. Each image is associated with zero or more labels. The distribution of concepts among images has a long tail. The numbers of images related to the most frequent 21 concepts are demonstrated by Figure 1.5. A 500-dimensional bag-of-words vector is used to represent each image in our experiments where the codebook is generated from SIFT descriptors.

1.5 Summary of Contributions

Our contributions to image retrieval can be grouped into three categories: Unsupervised multimodal retrieval, supervised hashing, and dynamic hashing. Our contributions are summarized as follows:

1. *Unsupervised multimodal retrieval*:

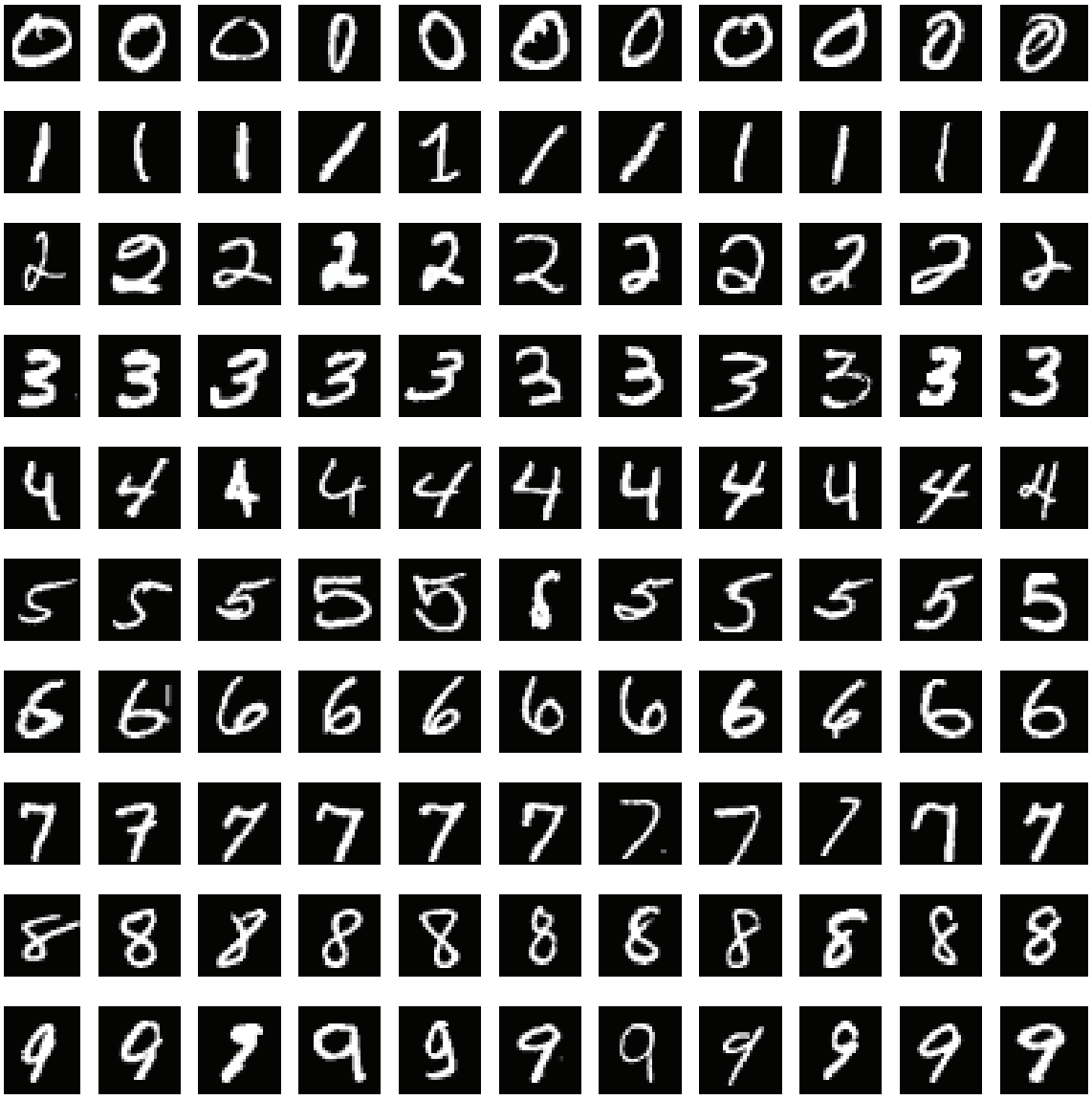


Figure 1.4: Each row shows sample images from one class in the MNIST dataset.

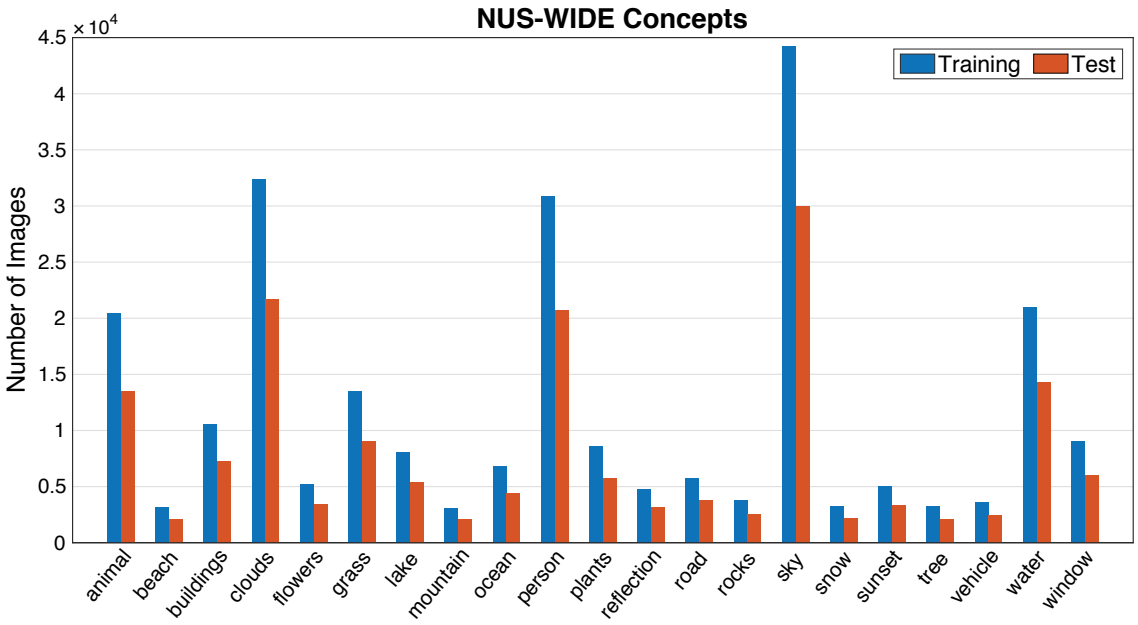


Figure 1.5: The number images associated with the most frequent 21 concepts in the NUS-WIDE dataset.

- We propose a Bayesian nonparametric framework based on the Indian Buffet Process (IBP) [25] for integrating multimodal data in a latent space. Since the IBP is a nonparametric prior in an infinite latent feature model, the proposed method offers a flexible way to determine the number of underlying abstract features in a dataset.
- We develop a retrieval system that can respond to cross-modal queries by introducing new random variables indicating relevance to a query. We present a Markov chain Monte Carlo (MCMC) algorithm for inference of the relevance of data.
- We formulate relevance feedback as pseudo-images to alter the distribution of images in the latent space so that user preferences influence the

ranking of images for a query.

2. *Supervised hashing*:

   - We propose a flexible Bayesian nonparametric model based on binary Gaussian process classification (GPC) for supervised learning of binary codes with better generalization.
   - We utilize the GPC predictive distribution to define a hash function through hyperplanes with kernels.
   - We developed a scalable parallel inference algorithm for the proposed model using a sparse approximation to the GPC model via a hybrid approach combining MCMC and message passing.

3. *Dynamic hashing*:

   - We propose a supervised hashing approach based on a two-stage classification framework that provides better generalization with regularizations and maximizes the entropy by balancing binary codes. We formulate our hashing objectives in a single optimization task.
   - We describe an algorithm that solves the optimization problem efficiently by an incremental strategy for training SVMs and an approximation to the solution of an NP-hard problem.
   - We define an incremental algorithm for the proposed hashing method that takes the earlier hashing functions and the final state of the database

as its input and efficiently computes new hashing functions which perform similarly to those computed from scratch on the final state of the database.

1.6 Organization of the Thesis

The rest of the thesis is organized as follows. Chapter 2 presents an overview of related works in the literature. In Chapter 3, we explain our multimodal image retrieval framework that allows cross-view queries and utilizes user relevance feedback. Chapter 4 introduces our supervised hashing method namely Gaussian process hashing and its inference algorithm. Chapter 5 describes our strategy for hashing on dynamic datasets. Experimental results are delivered separately in Chapters 3-5. Chapter 6 provides conclusions and future work.

Chapter 2: Literature Review

Our knowledge can be only finite,

while our ignorance must necessarily be infinite.

"Conjectures and Refutations" – Karl Popper

2.1 Overview

In this chapter, we give a brief review of the previous studies on hashing. Several hashing approaches have been proposed to discretize the feature space to for searching. Methods for learning binary codes can be roughly grouped into two learning schemes: Unsupervised hashing where binary codes are designed to preserve similarity in the original feature space; and supervised hashing where binary codes are constructed to capture semantic similarity between images. Representative methods for each category are given in the following sections.

2.1.1 Unsupervised Hashing

Unsupervised hashing techniques can be categorized as data independent and dependent schemes. Locality Sensitive Hashing (LSH) [2], is the most well-known data-independent hashing technique that uses random projections. The main draw-

back of LSH and its variants [26, 3, 27] is that their hash functions are independent of the data. This independence results in lower performance for shorter binary codes (256 bits or fewer). On the other hand, usage of longer binary codes to achieve good performance comes with an additional computational cost. Several methods have been proposed to learn hash functions from a training set representing the data at hand. Spectral Hashing [7] and graph-based methods [28, 29, 30, 31] try to construct compact binary codes by preserving the notion of similarity in the underlying manifold of the data. Quantization-based approaches [5, 32, 33, 34] focus on reducing the quantization error. Random Maximum Margin Hashing [35] replaces hash functions with support vector machines and train them on random splits of the data to obtain better generalization. Binary reconstructive embedding (BRE) [36] and bi-linear hyperplane hashing [37] learn hash functions by minimizing the difference between pairwise distances in the original space and the Hamming space. Unfortunately, unsupervised hashing methods unavoidably suffer from the semantic gap when they are employed for retrieval purposes where the relevance to a query is usually defined regarding semantic similarity rather than visual similarity.

Most recent unsupervised methods focus on multimodal data to bridge this gap by utilizing both textual and visual features. Bronstein *et al.* proposed cross-modality similarity learning via a boosting procedure [38]. Kumar and Udupa presented a cross-view similarity search [39] by generalizing spectral hashing [7] for multi-view data objects. Zhen and Yeung described two recent methods: Co-regularized hashing [40] based on a boosted co-regularization framework and a probabilistic generative approach called multimodal latent binary embedding [41] based

on binary latent factors. Nitish and Salakhutdinov proposed a deep Boltzmann machine for multimodal data [42]. Recently, Rastegari *et al.* proposed a predictable dual-view hashing [17] that aims to minimize the Hamming distance between binary codes obtained from two different views by utilizing multiple SVMs. Most of the multimodal hashing techniques are computationally expensive, especially when dealing with large-scale data. High computational and storage complexity restricts their scalability.

2.1.2 Supervised Hashing

Supervised hashing methods usually rely on pairwise labels, where $+1$ and -1 indicate that two points are similar and dissimilar, respectively. Furthermore, some methods use 0 for points which are neighbors in the metric space without being semantically related. Several supervised methods have been proposed as extensions to unsupervised techniques by making them aware of the label information. Iterative Quantization (ITQ) is used for both unsupervised and supervised hashing in conjunction with principal component analysis (PCA) and canonical correlation analysis (CCA), respectively [5]. Some methods focus on not only semantic similarity but also the similarity in the feature space. Semi-supervised hashing (SSH) [43, 44] and LDAHash [45] extends the spectral hashing framework to the supervised case. A supervised version of BRE was proposed by Kulis and Darrell [36]. Supervised hashing with kernels (KSH) [46] tries to minimize the Hamming distances between similar pairs and simultaneously maximize them for dissimilar pairs using inner products

of binary codes. Discriminative binary coding methods [47, 48], like Random Maximum Margin Hashing approach, try to improve generalization by using SVMs to preserve the semantic structure in the Hamming space. Lin *et al.* propose FastHash [49] that employs decision trees as hash functions and a GraphCut based method for the same type of optimization problem. Minimal loss hashing (MLH) [50] uses a structured SVM framework to generate binary codes. Supervised hashing with latent factor models (LFH) [51] uses latent factors for learning binary codes in a probabilistic framework. Recently proposed supervised discrete hashing (SDH) [52] employs a linear support vector machine (SVM) to capture the similarity in the semantic space. In most of the supervised hashing methods, each class is, in practice, associated with a binary string pattern. The methods aim to learn a hash function that maps all data points belonging to one class to the corresponding binary string. Consequently, overfitting to the training data becomes the most common problem for supervised hashing methods.

Chapter 3: Unsupervised Multimodal Retrieval

Photography, as a powerful medium of expression and communications, offers an infinite variety of perception, interpretation, and execution.

Ansel Adams

3.1 Overview

In this chapter, we propose a multimodal retrieval method based on latent features. A probabilistic approach is employed for learning binary codes, and also for modeling relevance and user preferences in image retrieval. Our model is built on the assumption that each image can be explained by a set of semantically meaningful *abstract features* which have both visual and textual components. For example, if a picture in the dataset contains a side view of a car, the words "car", "automobile" or "vehicle" will probably appear in the description; also, an object detector trained for vehicles will detect the vehicle in the image. Therefore, each image can be represented as a binary vector, with entries indicating the presence or absence of each abstract feature. Figure 3.1 shows the schematic overview of our retrieval algorithm.

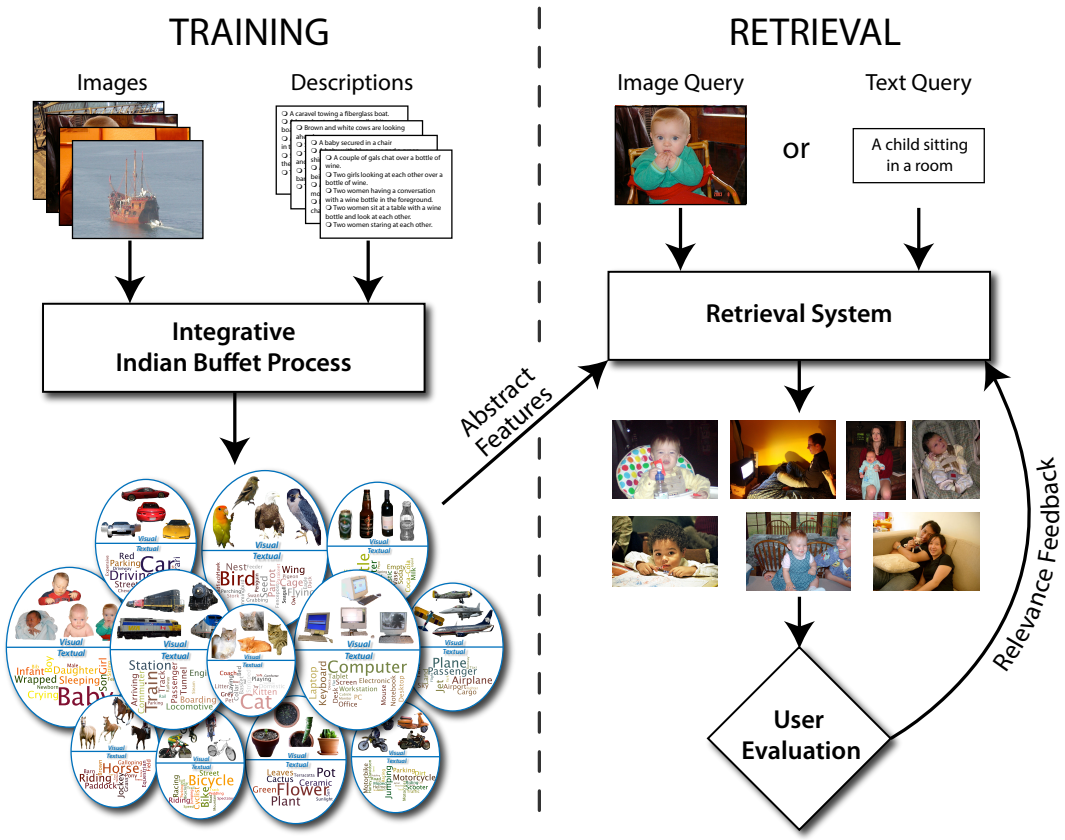


Figure 3.1: Schematic overview of our retrieval algorithm. The flow chart illustrates discovery of abstract features from multimodal data, the retrieval system for cross-view queries and user relevance feedback.

3.2 Latent Feature Model for Multimodal Retrieval

In our data model, each image has both textual and visual components. To facilitate the discussion, we assume that the dataset is composed of two full matrices; our approach can easily handle images with only one component, and it can be generalized to more than two modalities as well. We denote the data in the textual and visual space by $\mathbf{X}^{\tau}$ and $\mathbf{X}^{v}$, respectively. $\mathbf{X}^{*}$ is an $N \times D^{*}$ matrix whose rows corresponds to images in either space where $*$ is a placeholder used for either v or τ. The values in each column of $\mathbf{X}^{*}$ are centered by subtracting the sample mean of that column. The dimensionality of the textual space D^{τ} and the dimensionality of the visual space D^{v} can be different. We use $\mathcal{X}$ to represent the set $\{\mathbf{X}^{\tau}, \mathbf{X}^{v}\}$.

3.2.1 Integrative Latent Feature Model

We focus on how textual and visual values of an image are generated by a linear-Gaussian model and its extension for retrieval systems. Given a multimodal image dataset, the textual and visual data matrices, $\mathbf{X}^{\tau}$ and $\mathbf{X}^{v}$, can be approximated by $\mathbf{Z}\mathbf{A}^{\tau}$ and $\mathbf{Z}\mathbf{A}^{v}$, respectively. $\mathbf{Z}$ is an $N \times K$ binary matrix where Z_{nk} equals to one if abstract feature k is present in image n and zero otherwise. $\mathbf{A}^{*}$ is a $K \times D^{*}$ matrix where the textual and visual values for abstract feature k are stored in row k of $\mathbf{A}^{\tau}$ and $\mathbf{A}^{v}$, respectively (See Figure 3.2 for an illustration). The set $\{\mathbf{A}^{\tau}, \mathbf{A}^{v}\}$ is denoted by $\mathcal{A}$.

Our initial goal is to learn abstract features present in the dataset. Given $\mathcal{X}$,

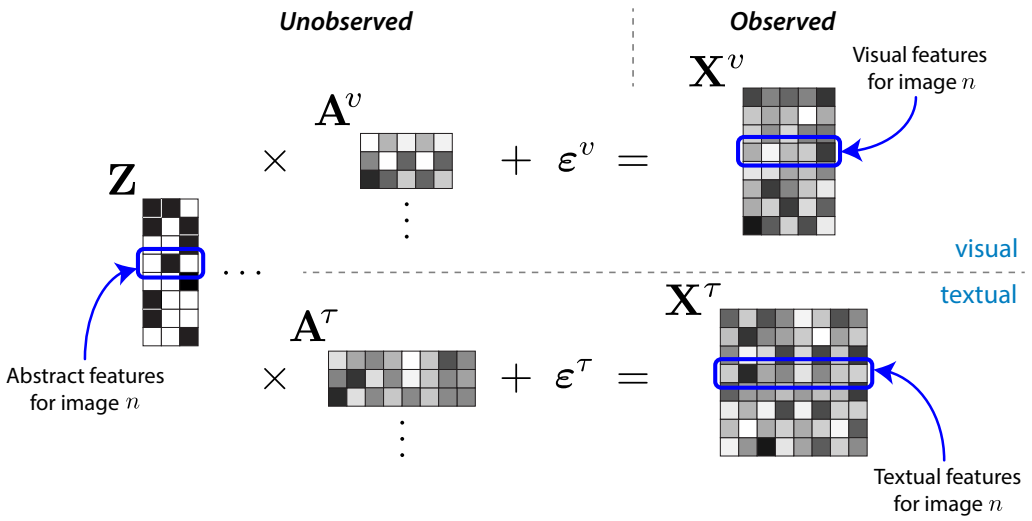


Figure 3.2: The latent abstract feature model proposes that visual data $\mathbf{X}^v$ is a product of $\mathbf{Z}$ and $\mathbf{A}^v$ with some noise; and similarly the textual data $\mathbf{X}^\tau$ is a product of $\mathbf{Z}$ and $\mathbf{A}^\tau$ with some noise.

we wish to compute the posterior distribution of $\mathbf{Z}$ and $\mathcal{A}$ using Bayes' rule

$$p(\mathbf{Z}, \mathcal{A}|\mathcal{X}) \propto p(\mathbf{X}^\tau|\mathbf{Z}, \mathbf{A}^\tau)p(\mathbf{A}^\tau)p(\mathbf{X}^v|\mathbf{Z}, \mathbf{A}^v)p(\mathbf{A}^v)p(\mathbf{Z}) \tag{3.1}$$

where $\mathbf{Z}$, $\mathbf{A}^\tau$ and $\mathbf{A}^v$ are assumed to be a priori independent. In our model, the vectors for textual and visual properties of an image are generated from Gaussian distributions with covariance matrix $(\sigma_x^*)^2\mathbf{I}$ and expectation $\mathbb{E}[\mathbf{X}^*]$ equal to $\mathbf{ZA}^*$. Similarly, a prior on $\mathbf{A}^*$ is defined to be Gaussian with zero mean vector and covariance matrix $(\sigma_a^*)^2\mathbf{I}$. Since we do not know the exact number of abstract features present in the dataset, we employ the Indian Buffet Process (IBP) to generate $\mathbf{Z}$, which provides a flexible prior that allows K to be determined at inference time (See [25] for details). The graphical model of our integrative approach is shown in Figure 3.3. The same integrative model first proposed by Yildirim and Jacobs in the cognitive science area [53].

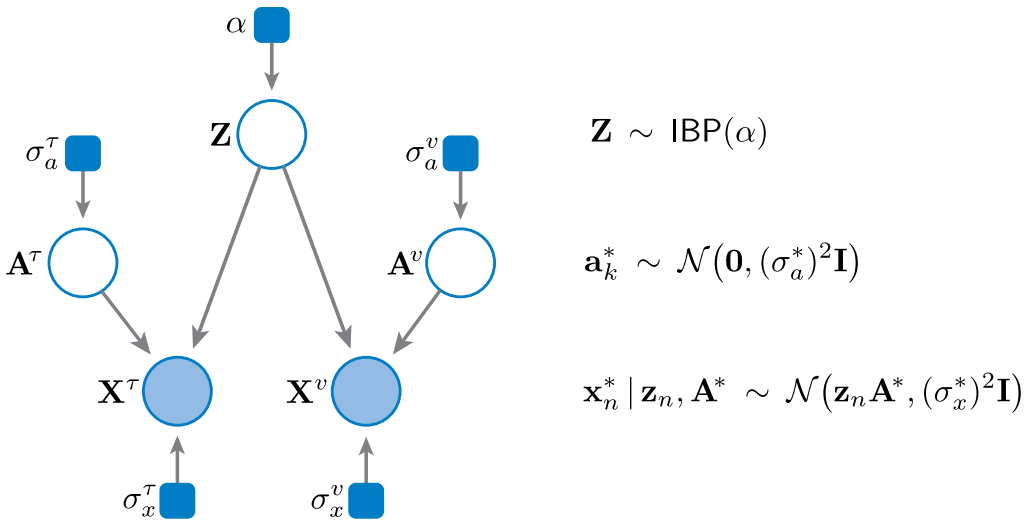


Figure 3.3: Graphical model for the integrative IBP approach where circles indicate random variables, shaded circles denote observed values, and the blue square boxes are hyperparameters.

The exchangeability property of the IBP leads directly to a Gibbs sampler which takes image n as the last customer to have entered the buffet. Then, we can sample Z_{nk} for all initialized features k via

$$p(Z_{nk} = 1|\mathbf{Z}_{-nk}, \mathcal{X}) \propto p(Z_{nk} = 1|\mathbf{Z}_{-nk})p(\mathcal{X}|\mathbf{Z}). \tag{3.2}$$

where $\mathbf{Z}_{-nk}$ denotes entries of $\mathbf{Z}$ other than Z_{nk}. In the finite latent feature model (where K is fixed), the conditional distribution for any Z_{nk} is given by

$$p(Z_{nk} = 1|\mathbf{Z}_{-nk}) = \frac{m_{-n,k} + \frac{\alpha}{K}}{N + \frac{\alpha}{K}} \tag{3.3}$$

where $m_{-n,k}$ is the number of images possessing abstract feature k apart from image n. In the infinite case like the IBP, we obtain $p(Z_{nk} = 1|\mathbf{Z}_{-nk}) = \frac{m_{-n,k}}{N}$ for any k such that $m_{-n,k} > 0$. We also need to draw new features associated with image n from $\mathsf{Poisson}\big(\frac{\alpha}{N}\big)$, and the likelihood term is now conditioned on $\mathbf{Z}$ with new additional

columns set to one for image n.

For the linear-Gaussian model, the collapsed likelihood function $p(\mathcal{X}|\mathbf{Z}) = p(\mathbf{X}^{\tau}|\mathbf{Z})p(\mathbf{X}^{v}|\mathbf{Z})$ can be computed using

$$p(\mathbf{X}^*|\mathbf{Z}) = \int p(\mathbf{X}^*|\mathbf{Z}, \mathbf{A}^*)p(\mathbf{A}^*)\, d\mathbf{A}^* = \frac{\exp\{-\frac{1}{2(\sigma_x^*)^2}\mathrm{tr}(\mathbf{X}^{*T}(\mathbf{I} - \mathbf{Z}\mathbf{M}\mathbf{Z}^T)\mathbf{X}^*)\}}{(2\pi)^{\frac{ND^*}{2}}(\sigma_x^*)^{(N-K)D^*}(\sigma_a^*)^{KD^*}|\mathbf{M}|^{\frac{-D^*}{2}}} \quad (3.4)$$

where $\mathbf{M} = \left(\mathbf{Z}^T\mathbf{Z} + \frac{(\sigma_x^*)^2}{(\sigma_a^*)^2}\mathbf{I}\right)^{-1}$ and $\mathrm{tr}(\cdot)$ is the trace of a matrix [25]. To reduce the computational complexity, Doshi-Velez and Ghahramani proposed an accelerated sampling in [54] by maintaining the posterior distribution of $\mathbf{A}^*$ conditioned on partial $\mathbf{X}^*$ and $\mathbf{Z}$. We use this approach to learn binary codes, *i.e.* the feature-assignment matrix $\mathbf{Z}$, for multimodal data. Unlike the hashing methods that learn optimal hyperplanes from training data [5, 34, 17], we only sample $\mathbf{Z}$ without specifying the length of binary codes in this process. Therefore, the binary codes can be updated efficiently if new images are added in a long run of the retrieval system.

3.2.2 Retrieval Model

We extend the integrative IBP model for image retrieval. Given a query, we need to sort the images in the dataset concerning their relevance to the query. A query can be comprised of textual and visual data, or either component can be absent. Let $\mathbf{q}^{\tau}$ be a D^{τ}-dimensional vector for the textual values and $\mathbf{q}^{v}$ be a D^{v}-dimensional vector for the visual values of the query. We can write $\mathcal{Q} = \{\mathbf{q}^{\tau}, \mathbf{q}^{v}\}$. As for the images in $\mathcal{X}$, we consider a query to be generated by the same model described in the previous section except the prior on abstract features. In the retrieval part, we consider $\mathbf{Z}$ as a known quantity, and we fix the number abstract

features to K. Therefore, the feature assignments for the dataset are not affected by queries. Besides, queries are explained by known abstract features only.

We extend the Indian restaurant metaphor to construct the retrieval model. A query corresponds to the $(N+1)$th customer to enter the buffet. The previous customers are divided into two classes as friends and non-friends based on their relevance to the new customer. The new customer now samples from at most K dishes in proportion to their popularity among friends and also their unpopularity among non-friends. Consequently, the dishes sampled by the new customer are expected to be similar to those of friends and dissimilar to those of non-friends. Let $\mathbf{r}$ be an N-dimensional vector where r_n equals to one if customer n is a friend of the new customer and zero otherwise. For this finitely long buffet, the sampling probability of dish k by the new customer can be written as $\frac{m'_k+\alpha/K}{N+1+\alpha/K}$ where $m'_k = \sum_{n=1}^{N}(Z_{nk})^{r_n}(1-Z_{nk})^{1-r_n}$, that is the total number of friends who tried dish k and non-friends who did not sample dish k. Let $\mathbf{z}'$ be a K-dimensional vector where z'_k records if the new customer (query) sampled dish k. We place a prior over r_n as $\mathsf{Bernoulli}(\theta)$. Then, we can sample z'_k from

$$p(z'_k = 1|\mathbf{z}'_{-k}, \mathcal{Q}, \mathbf{Z}, \mathcal{X}) \propto p(z'_k = 1|\mathbf{Z})p(\mathcal{Q}|\mathbf{z}', \mathbf{Z}, \mathcal{X}). \tag{3.5}$$

The probability $p(z'_k = 1|\mathbf{Z})$ can be computed efficiently for $k = 1, \ldots, K$ by marginalizing over $\mathbf{r}$ as below:

$$p(z'_k = 1|\mathbf{Z}) = \sum_{\mathbf{r}\in\{0,1\}^N} p(z'_k = 1|\mathbf{r}, \mathbf{Z})p(\mathbf{r}) = \frac{\theta m_k + (1-\theta)(N-m_k) + \frac{\alpha}{K}}{N+1+\frac{\alpha}{K}}. \tag{3.6}$$

The collapsed likelihood of the query, $p(\mathcal{Q}|\mathbf{z}', \mathbf{Z}, \mathcal{X})$, is given by the product of textual and visual likelihood values, $p(\mathbf{q}^\tau|\mathbf{z}', \mathbf{Z}, \mathbf{X}^\tau)p(\mathbf{q}^v|\mathbf{z}', \mathbf{Z}, \mathbf{X}^v)$. If either textual

or visual component is missing, we can simply integrate out the missing one by omitting the corresponding term from the equation. The likelihood of each part can be calculated as follows:

$$p(\mathbf{q}^*|\mathbf{z}', \mathbf{Z}, \mathbf{X}^*) = \int p(\mathbf{q}^*|\mathbf{z}', \mathbf{A}^*)p(\mathbf{A}^*|\mathbf{Z}, \mathbf{X}^*)\, d\mathbf{A}^* = \mathcal{N}(\mathbf{q}^*; \boldsymbol{\mu}_q^*, \boldsymbol{\Sigma}_q^*). \tag{3.7}$$

where the mean and covariance matrix of the normal distribution are given by $\boldsymbol{\mu}_q^* = \mathbf{z}'\mathbf{M}\mathbf{Z}^T\mathbf{X}^*$ and $\boldsymbol{\Sigma}_q^* = (\sigma_x^*)^2(\mathbf{z}'\mathbf{M}\mathbf{z}'^T + \mathbf{I})$, akin to the update equation in [54] (Refer to (3.4) for $\mathbf{M}$).

Finally, we use the conditional expectation of $\mathbf{r}$ to rank images in the dataset with respect to their relevance to the given query. Calculating the expectation $\mathbb{E}[\mathbf{r}|\mathcal{Q}, \mathbf{Z}, \mathcal{X}]$ is computationally expensive; however, it can be empirically estimated using the Monte Carlo method as follows:

$$\hat{\mathbb{E}}[r_n|\mathcal{Q}, \mathbf{Z}, \mathcal{X}] = \frac{1}{I}\sum_{i=1}^{I} p(r_n = 1|\mathbf{z}'^{(i)}, \mathbf{Z}) = \frac{\theta}{I}\sum_{i=1}^{I}\prod_{k=1}^{K}\frac{p\big(z_k'^{(i)}|r_n = 1, \mathbf{Z}\big)}{p\big(z_k'^{(i)}|\mathbf{Z}\big)} \tag{3.8}$$

where $\mathbf{z}'^{(i)}$ represents i.i.d. samples from (3.5) for $i = 1, \ldots, I$. The last equation required for computing (3.8) is

$$p(z_k' = 1|r_n = 1, \mathbf{Z}) = \frac{Z_{nk} + \theta m_{-n,k} + (1-\theta)(N - 1 - m_{-n,k}) + \frac{\alpha}{K}}{N + 1 + \frac{\alpha}{K}}. \tag{3.9}$$

The retrieval system returns a set of top ranked images to the user. Note that we compute the expectation of relevance vector instead of sampling directly since binary values indicating the relevance are less stable and they hinder the ranking of images.

3.2.3 Relevance Feedback Model

In our data model, user preferences can be described over abstract features. For instance, if abstract feature k is present in the most of the positive samples *i.e.* images judged as relevant by the user and it is absent in the irrelevant ones, then we can say that the user is more interested in the semantic subspace represented by abstract feature k. In the revised query, the images having abstract feature k are expected to be ranked in higher positions in comparison to the initial query. We can achieve this desirable property from query-specific alterations to the sampling probability in (3.5) for the corresponding abstract features. Our approach is to add pseudo-images to the feature assignment matrix $\mathbf{Z}$ before the computations of the revised query. For the Indian restaurant analogy, pseudo-images correspond to some additional friends of the new customer (query), who do not exist in the restaurant. The distribution of dishes sampled by those imaginary customers reflects user relevance feedback. Thus, the updated expectation of the relevance vector has a bias towards user preferences.

Let $\mathbf{Z}_u$ be an $N_u \times K$ feature-assignment matrix for pseudo-images only; then the number of pseudo-images, N_u, determines the influence of relevance feedback. Therefore, we set an upper limit on N_u as the number of real images, N, by placing a prior distribution as $N_u \sim \mathsf{Binomial}(\gamma, N)$ where γ is a parameter that controls the weight of feedback. Let $m_{u,k}$ be the number of pseudo-images containing abstract feature k; then this number has an upper bound N_u by definition. For abstract feature k, a prior distribution conditioned on N_u can be defined as $m_{u,k}|N_u \sim$

$\mathsf{Binomial}(\phi_k, N_u)$ where ϕ_k is a parameter that can be tuned by relevance judgments.

Let $\mathbf{z}''$ be a K-dimensional feature-assignment vector for the revised query; then we can sample each z''_k via

$$p(z''_k = 1|\mathbf{z}''_{-k}, \mathcal{Q}, \mathbf{Z}, \mathcal{X}) \propto p(z''_k = 1|\mathbf{Z})p(\mathcal{Q}|\mathbf{z}'', \mathbf{Z}, \mathcal{X}) \tag{3.10}$$

where the computation of the collapsed likelihood is already shown in (3.7). Note that we do not actually generate all entries of $\mathbf{Z}_u$ but only the sum of its columns $\mathbf{m}_u$ and number of rows N_u for computing the sampling probability. We can write the first term as:

$$\begin{aligned} p(z''_k = 1|\mathbf{Z}) &= \sum_{N_u=0}^{N} p(N_u) \sum_{m_{u,k}=0}^{N_u} p(m_{u,k}|N_u) \sum_{\mathbf{r}\in\{0,1\}^N} p(z''_k = 1|\mathbf{r}, \mathbf{Z}_u, \mathbf{Z})p(\mathbf{r}) \\ &= \sum_{j=0}^{N} \binom{N}{j} \gamma^j (1-\gamma)^{N-j} \frac{\theta m_k + (1-\theta)(N - m_k) + \frac{\alpha}{K} + \phi_k j}{N + 1 + \frac{\alpha}{K} + j} \end{aligned} \tag{3.11}$$

Unfortunately, this expression has no compact analytic form; however, it can be efficiently computed numerically by contemporary scientific computing software even for large values of N. In this equation, one can alternatively fix r_n to 1 if the user marks observation n as relevant or 0 if it is indicated to be irrelevant. Finally, the expectation of $\mathbf{r}$ is updated using (3.8) with new i.i.d. samples $\mathbf{z}''^{(i)}$ from (3.10) and the system constructs the revised set of images. Figure 3.4 demonstrates the graphical model of the relevance feedback model. Note that this model becomes equivalent to the query model when $\gamma = 0$.

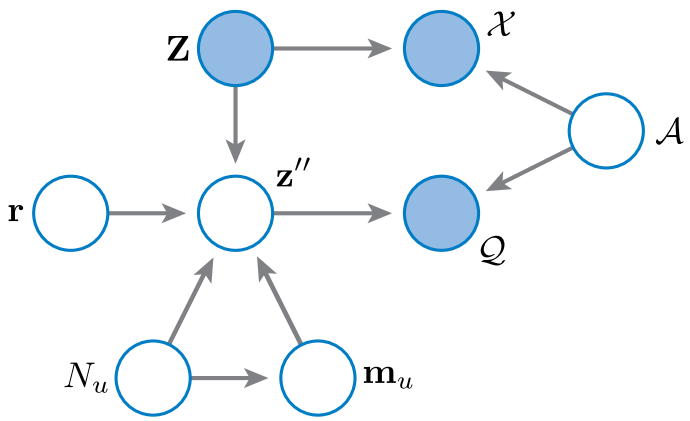


Figure 3.4: Graphical model for the feedback query model. Circles indicate random variables, shaded circles denote observed values. Hyperparameters are omitted for clarity. Note that $\mathbf{Z}$ is considered as an observed variable in the retrieval part.

3.3 Experiments

The experiments were performed in two phases. We first compared the performance of our method in category retrieval with several state-of-the-art hashing techniques. Next, we evaluated the improvement in the performance of our method with relevance feedback. We used the same multimodal datasets as [17], namely PASCAL-Sentence 2008 dataset [55] and the SUN-Attribute dataset [19]. In the quantitative analysis, we used the mean of the interpolated precision at standard recall levels for comparing the retrieval performance. In the qualitative analysis, we present the images retrieved by our proposed method for a set of text-to-image and image-to-image queries. All experiments were performed in the Matlab environment[1].

[1]Our code is available at http://www.cs.umd.edu/∼ozdemir/iibp

3.3.1 Experimental Setup

Firstly, all features were centered to zero and normalized to unit variance; also, duplicate features were removed from the data. We reduced the dimensionality of visual features in the SUN dataset from 19,080 to 1,000 by random feature selection, which is preferable to PCA for preserving the variance among visual features. The Gibbs sampler was initialized with a randomly sampled feature assignment matrix $\mathbf{Z}$ from an IBP prior. We set $\alpha = 1$ in all experiments to keep binary codes short. The other hyperparameters σ_a^* and σ_x^* were determined by adding Metropolis steps to the MCMC algorithm to prevent one modality from dominating the inference process.

In the retrieval part, the relevance probability θ was set to 0.5 so that all abstract features have equal prior probability from (3.6). Feature assignments of a query were initialized with all zero bits. For relevance feedback analysis, we set $\gamma = 1$ (equal significance for the data and feedback) and we decide each ϕ_k as follows:

Let $\bar{z}'_k = \frac{1}{I}\sum_{i=1}^{I} z_k'^{(i)}$ where each $\mathbf{z}'^{(i)}$ is drawn from (3.5) for a given query; and $\hat{z}'_k = \frac{1}{T}\sum_{t=1}^{T}(Z_{tk})^{r_t}(1 - Z_{tk})^{1-r_t}$ where t represents the index of each image judged by the user and T is the size of relevance feedback. The difference between these two quantities, $\delta_k = \bar{z}'_k - \hat{z}'_k$, controls ϕ_k which is defined by a logistic function as

$$\phi_k = \frac{1}{1 + e^{-(c\delta_k + \beta_{0,k})}} \tag{3.12}$$

where c is a constant and $\beta_{0,k} = \ln \frac{p(z'_k=1|\mathbf{Z})}{p(z'_k=0|\mathbf{Z})}$ (refer to (3.6) for $p(z'_k|\mathbf{Z})$). We set $c = 5$ in our experiments. Note that $\phi_k = p(z'_k = 1|\mathbf{Z})$ when $\bar{z}'_k$ is equal to $\hat{z}'_k$.

3.3.2 Experimental Results

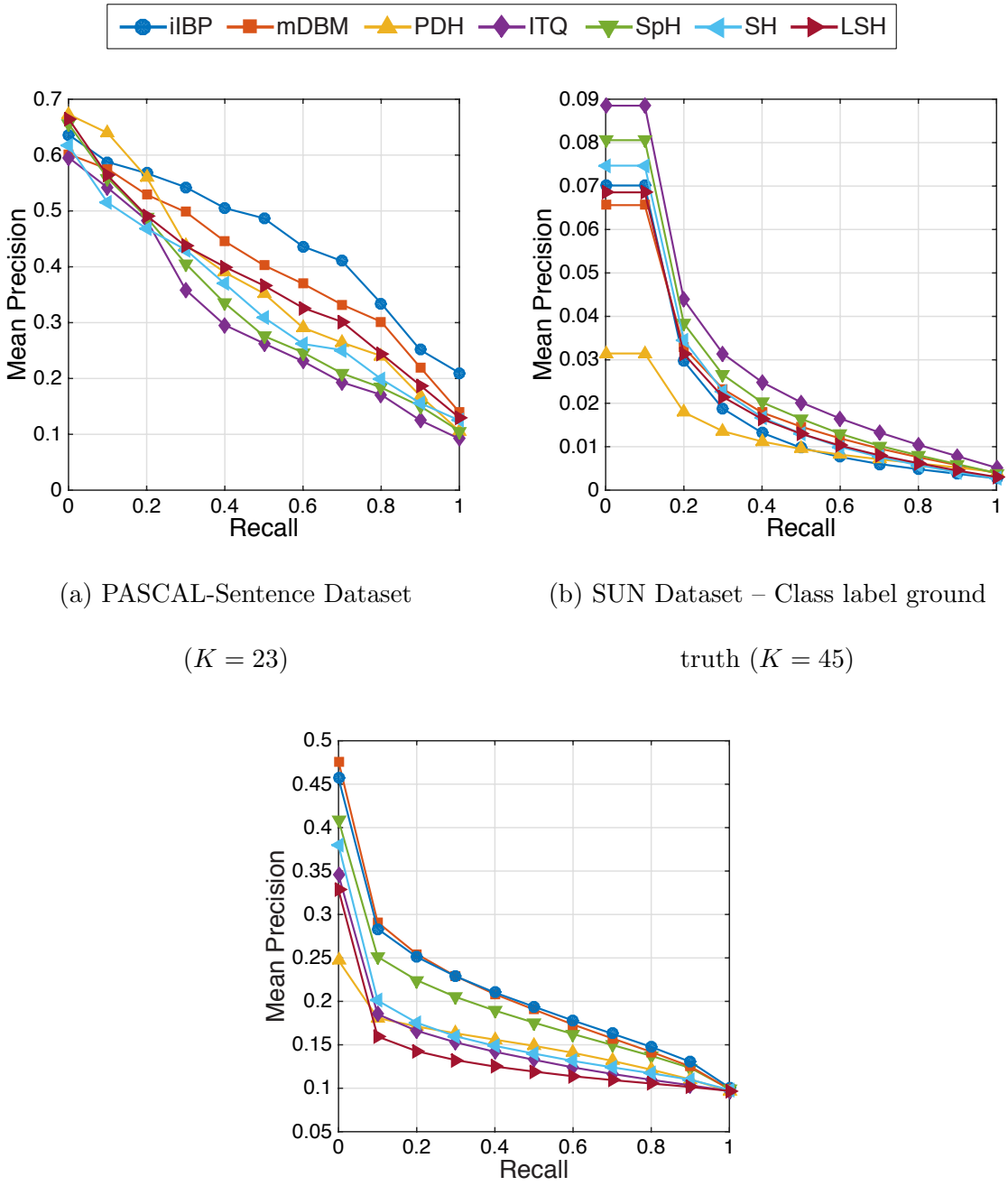
We compared our method, called integrative IBP (iIBP), with several hashing methods including locality sensitive hashing (LSH) [2], spectral hashing (SH) [7], spherical hashing (SpH) [34], iterative quantization (ITQ) [5], multimodal deep Boltzmann machine (mDBM) [42] and predictable dual-view hashing (PDH) [17]. We divided each dataset into two equal sized train and test segments. The train segment was first used for learning the feature assignment matrix $\mathbf{Z}$ by iIBP. Then, the other binary code methods were trained with the same code length K. We used supervised ITQ coupled with CCA [56] and took the dual-view approach [17] to construct basis vectors in a common subspace. However, LSH, SH, and SpH were applied on single-view data since they do not support cross-view queries.

All images in the test segment were used as both image and text queries. Given a query, images in the train set were ranked by iIBP with (3.8). For all other methods, we use Hamming distance between binary codes in the nearest-neighbor search. Mean precision curves are presented in Figure 3.5 for both datasets. Unlike the experiments in [17] performed in a supervised manner, the performance on the SUN-Attribute dataset is very low due to the small number of positive samples compared to the number of categories (Figure 3.5b). There are only ten relevant images among 7,170 training images. Therefore, we also used Euclidean neighbor ground truth labels computed from visual data as in [5] (Figure 3.5c). As seen in the figure, our method (iIBP) outperforms all other methods. Although unimodal hashing methods perform well on text queries, they suffer badly on image queries

because the semantic similarity to the query does not necessarily require visual similarity. By the joint analysis of visual and textual spaces, our approach improves the performance of image queries by bridging the semantic gap [8]. Mean precision curves are presented in Figures 3.6 and 3.7 for text and image queries, respectively. The curves given in Figure 3.5c are the averages of these two types of queries. Also, we analyzed the stability of our Gibbs sampler in retrieval from the PASCAL-Sentence dataset by running each query for 50 times. Figure 3.8 shows the range of mean precisions in trials by error bars for text and image queries. The error bars are tiny at all recall levels for both query types. Hence, the image set retrieved by our method for a given query is very stable. Figure 3.9 shows the effect of sample size in the Monte Carlo estimation, I of (3.8), on the retrieval performance for the PASCAL-Sentence dataset. The precision curves start to overlap at $I = 5$. Therefore, one can use a small sample set to estimate the expectation of relevance vector $\mathbf{r}$ for faster processing of the query at a similar precision level.

For qualitative analysis, Figure 3.10a shows the top-5 retrieved images from the PASCAL-Sentence 2008 dataset for image queries. Thanks to the integrative approach, the retrieved images share remarkable semantic similarity with the query images. Similarly, most of the retrieved images for the text-to-image queries in Figure 3.10b comprise the semantic structure in the query sentences.

In the second phase of analyzes, we utilized the rankings in the first step to deciding relevance feedback parameters independently for each query. We picked the top two relevant images as positive samples and top two irrelevant images as negative samples. We set each ϕ_k by (3.12) and reordered the images using the rel-

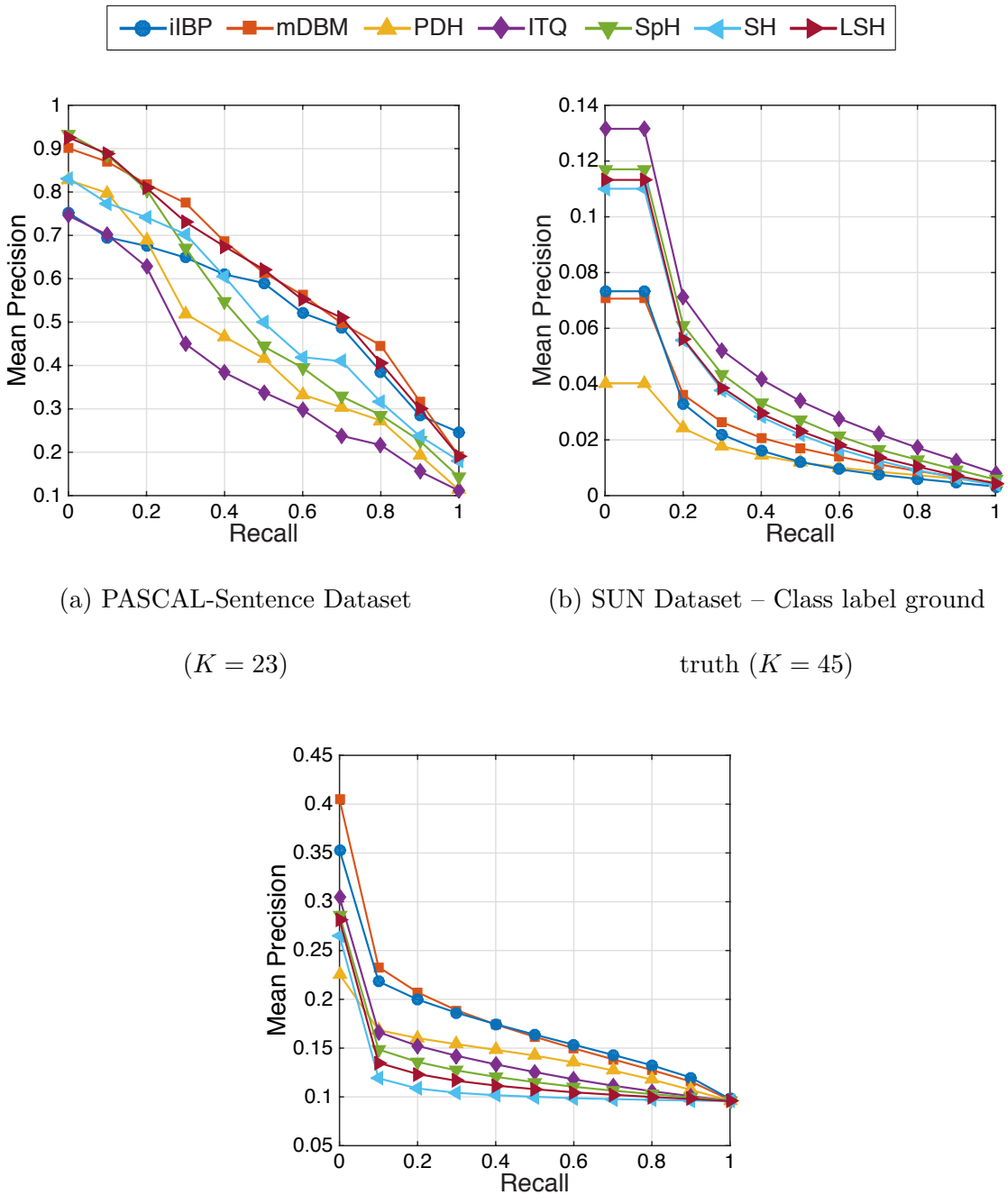


(a) PASCAL-Sentence Dataset ($K = 23$)

(b) SUN Dataset – Class label ground truth ($K = 45$)

(c) SUN Dataset – Euclidean ground truth ($K = 45$)

Figure 3.5: The result of category retrieval for all query types (image-to-image and text-to-image queries). Our method (iIBP) is compared with the-state-of-the-art methods.

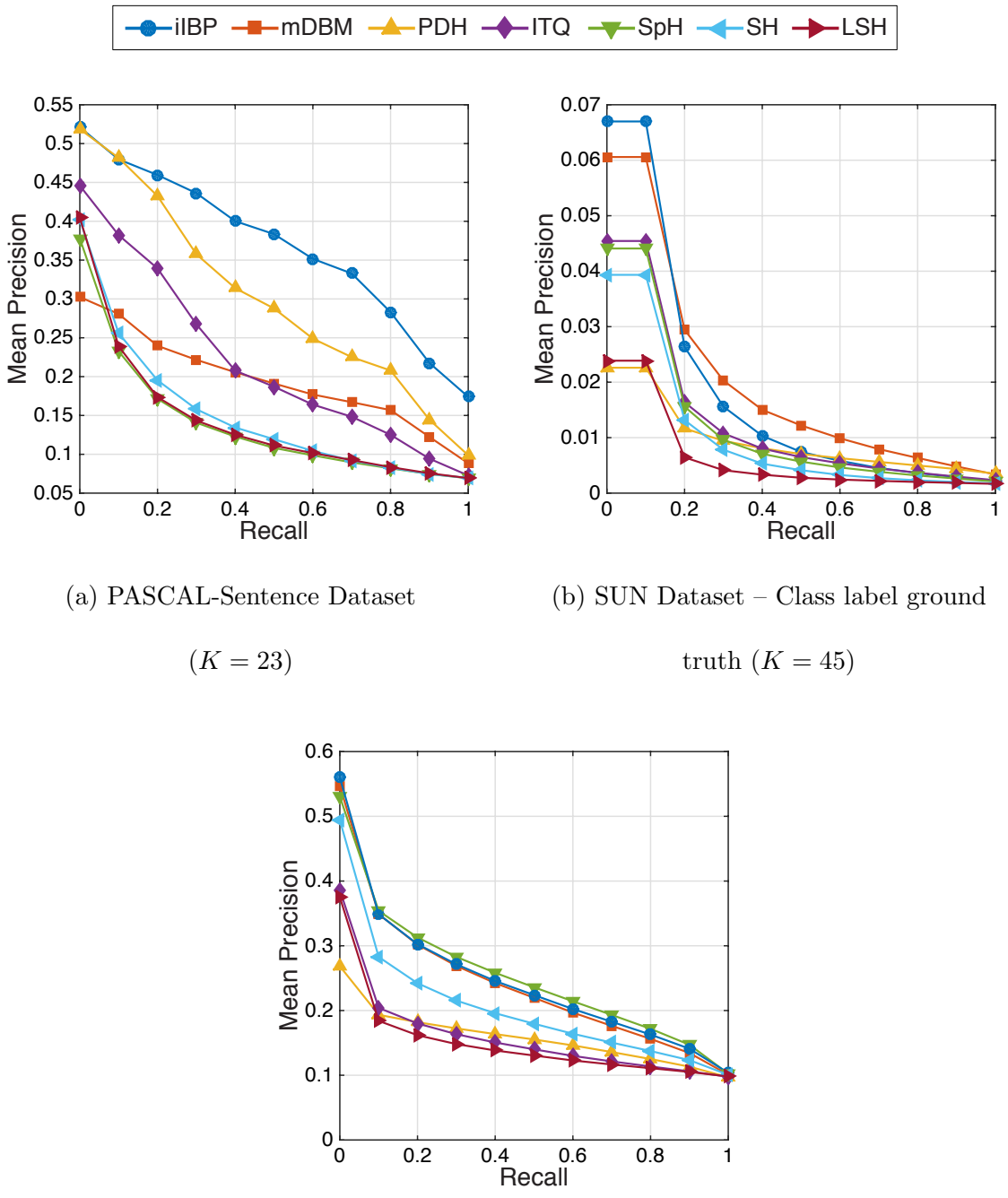


(a) PASCAL-Sentence Dataset ($K = 23$)

(b) SUN Dataset – Class label ground truth ($K = 45$)

(c) SUN Dataset – Euclidean ground truth ($K = 45$)

Figure 3.6: The result of category retrieval for text-to-image queries. Our method (iIBP) is compared with the-state-of-the-art methods.



(a) PASCAL-Sentence Dataset ($K = 23$)

(b) SUN Dataset – Class label ground truth ($K = 45$)

(c) SUN Dataset – Euclidean ground truth ($K = 45$)

Figure 3.7: The result of category retrieval for image-to-image queries. Our method (iIBP) is compared with the-state-of-the-art methods.

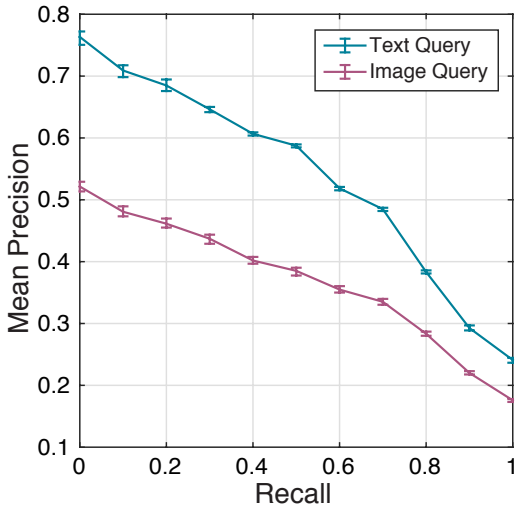


Figure 3.8: The result of stability analysis for text and image queries. Our method (iIBP) is applied on the PASCAL-Sentence dataset for 50 times. The curves represent the average level of mean precisions. Error bars indicate the range of mean precisions observed at each standard recall level.

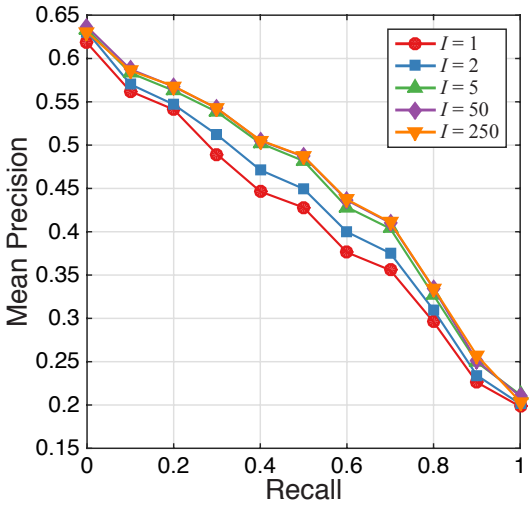
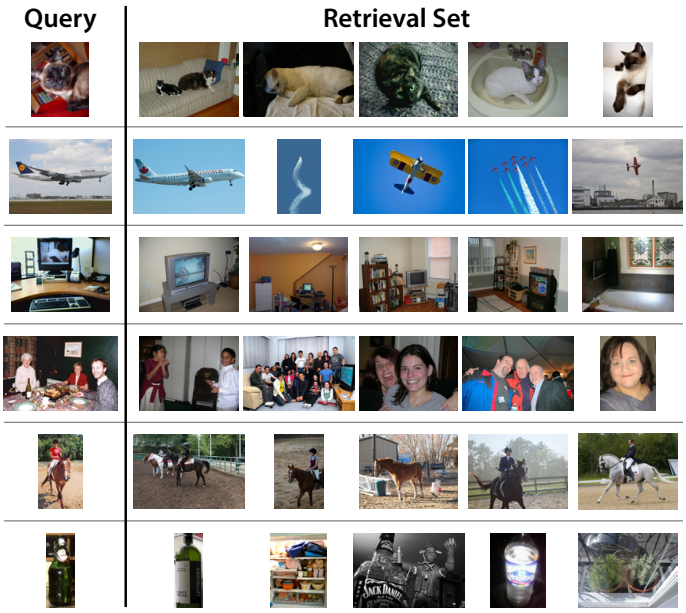


Figure 3.9: The result of parameter effect analysis. Our method (iIBP) is applied on the PASCAL-Sentence dataset for different values of I, the sample size in the Monte Carlo estimation.



(a) Image-to-image queries

(b) Text-to-image queries

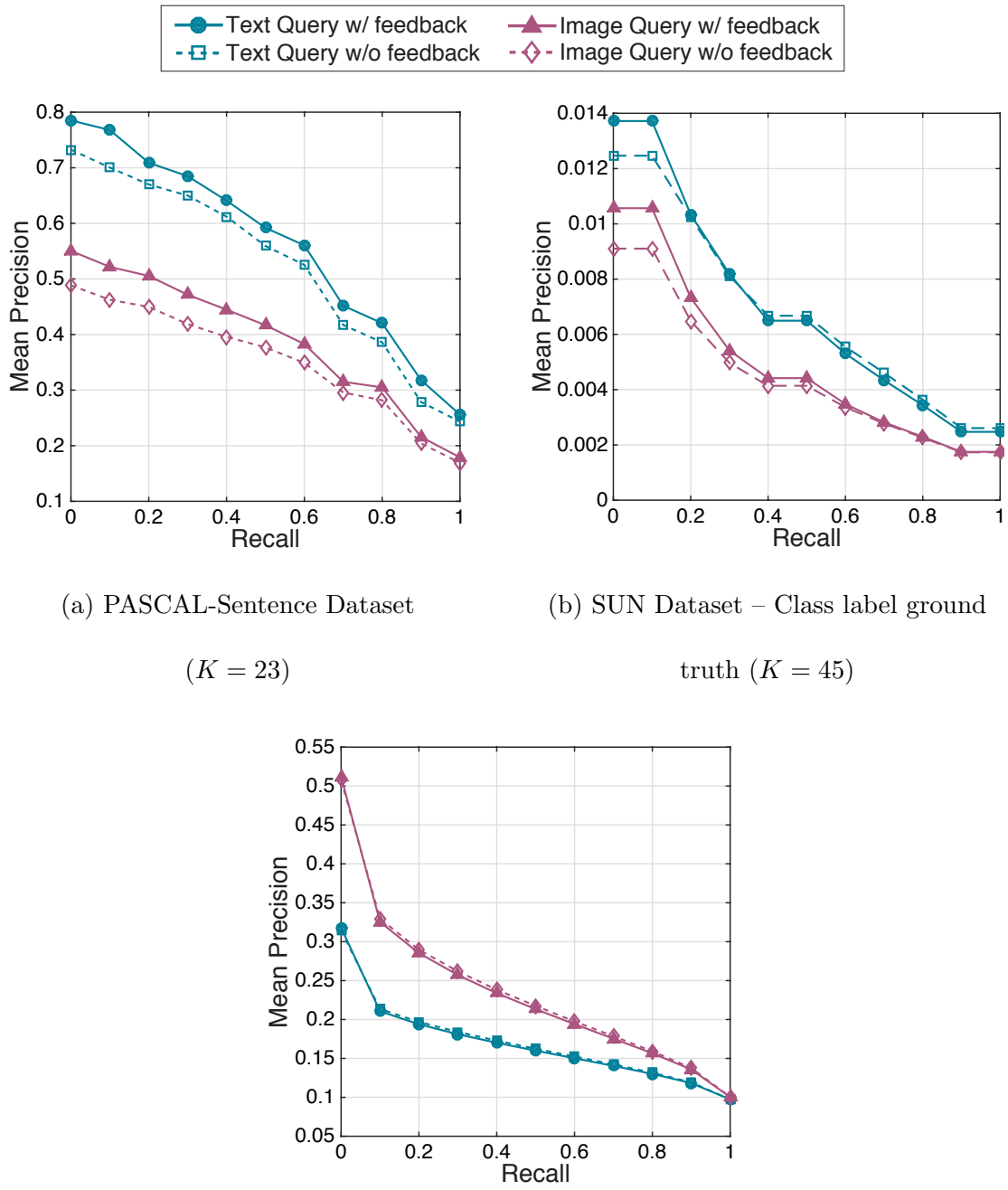
Figure 3.10: Sample images retrieved from the PASCAL-Sentence dataset by our method (iIBP).

evance feedback model excluding the ones used as user relevance judgments. Those images were omitted from precision-recall calculations as well. Figure 3.11 illustrates that relevance feedback slightly boosts the retrieval performance, especially for the PASCAL-Sentence dataset.

The computational complexity of an iteration is $O(K^2 + KD^*)$ for a query and $O(N(K^2 + KD^\tau + KD^v))$ for training [54]. The feature assignment vector $\mathbf{z}'$ of a query usually converges in a few iterations. A typical query took less than 1 second in our experiments for $I = 50$ with our optimized Matlab code.

3.4 Conclusion

We proposed a novel retrieval scheme based on latent binary features for multimodal data. We also describe how to utilize relevance feedback for better retrieval performance. The experimental results on real-world data demonstrate that our method outperforms state-of-the-art hashing techniques.



(a) PASCAL-Sentence Dataset ($K = 23$)

(b) SUN Dataset – Class label ground truth ($K = 45$)

(c) SUN Dataset – Euclidean ground truth ($K = 45$)

Figure 3.11: The result of category retrieval by our approach (iIBP) with relevance feedback for text and image queries. Revised retrieval with relevance feedback is compared with initial retrieval.

Chapter 4: Supervised Hashing

All stable processes we shall predict.

All unstable processes we shall control.

John von Neumann

4.1 Overview

The Gaussian process is an elegant model based on Bayesian nonparametrics for nonlinear regression and classification. Similar to SVM, which is employed in [17, 50], binary Gaussian process classification (GPC) model [14] can be utilized for hashing in a Bayesian framework. In contrast to SVM, the GPC model provides an error bar for its prediction via an estimate of prediction variance. In this chapter, we propose a supervised hashing method with kernels based on the GPC model, so we call the proposed approach *Gaussian Process Hashing* (GPH). We use a fully probabilistic approach for learning binary codes from data. In the GPH model, the variables corresponding to the binary classes in the original GPC model become latent variables and a new level of variables that correspond to a pairwise similarity between data points is integrated into the model. Averaging over nonlinear classification models by Gaussian processes is our motivation behind the GPH model to

overcome the overfitting problem of supervised hashing.

4.2 Gaussian Process Hashing

4.2.1 Problem Definition

Given a set of data points $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, each $\mathbf{x}_i \in \mathbb{R}^d$, the goal of hashing is to find a function H that maps data points from $\mathbb{R}^d$ to m-dimensional Hamming space with respect to some optimization criteria. Let $\mathcal{S} = \{s_{il}\}$ denote the set of similarity labels for pairs of data points where $s_{il} = +1$ if $\mathbf{x}_i$ and $\mathbf{x}_l$ are similar and $s_{il} = -1$ otherwise, then a desirable hash function yields a smaller Hamming distance between $H(\mathbf{x}_i)$ and $H(\mathbf{x}_l)$ when $s_{il} = +1$ and a larger one when $s_{il} = -1$. Many techniques [2, 5, 46, 3] construct such a hash function $H : \mathbb{R}^d \to \{-1, +1\}^m$ by combining m independent binary encoding functions $h_1(\mathbf{x}), \ldots, h_m(\mathbf{x})$ such that $h_j(\mathbf{x}) = \text{sgn}(\mathbf{w}_j^\top \mathbf{x} + b_j)$ where $\mathbf{w}_j \in \mathbb{R}^d$ is a hyperplane, $b_j \in \mathbb{R}$ is an intercept for $j = 1, \ldots, m$ and

$$\text{sgn}(v) = \begin{cases} +1 & \text{if } v \geq 0, \\ -1 & \text{otherwise.} \end{cases}$$

Some hashing methods [3, 46] employ a kernel function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ such that each binary encoding hash function has the form $h_j(\mathbf{x}) = \text{sgn}(\mathbf{w}_j^\top \mathbf{k} + b_j)$ where $\mathbf{k} = [k(\mathbf{x}, \bar{\mathbf{x}}_1), \ldots, k(\mathbf{x}, \bar{\mathbf{x}}_r)]^\top$ and $\{\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_r\} \subseteq \mathcal{X}$.

4.2.2 A Probabilistic Approach

The hash function H can be modeled as a latent function that builds a bridge between $\mathcal{X}$ and $\mathcal{S}$, *i.e.* the observed data $\mathcal{D} = \{\mathcal{X}, \mathcal{S}\}$. Let $\mathbf{Y}$ be a random binary matrix of size $n \times m$ where $Y_{ij} = h_j(\mathbf{x}_i)$ and $\mathbf{y}_j$ is the jth column of $\mathbf{Y}$, *i.e.* shorthand for the values of the hash function. Then using Bayes' rule we can write

$$p(\mathbf{Y}|\mathcal{D}) = \frac{p(\mathcal{S}|\mathbf{Y})p(\mathbf{Y}|\mathcal{X})}{p(\mathcal{S}|\mathcal{X})}. \tag{4.1}$$

Given the binary codes $\mathbf{Y}$, the similarity labels $\mathcal{S}$ are assumed to be independent Bernoulli variables. Let $\mathbf{V} = \mathbf{Y}\mathbf{Y}^\top$, *i.e.* V_{il} is the inner product of binary codes for sample i and sample l; then the joint likelihood factorizes as

$$p(\mathcal{S}|\mathbf{Y}) = \prod_{(i,l)} p(s_{il}|V_{il}) = \prod_{i=2}^{n-1} \prod_{l=i+1}^{n} p(s_{il}|V_{il}). \tag{4.2}$$

We use the probit model to define $p(s_{il} = +1|V_{il}) = \Phi(\sigma_y V_{il})$ where Φ denotes the cumulative distribution function of the standard normal distribution and $\sigma_y > 0$ is a scaling parameter. The individual likelihood terms can be written as $p(s_{il}|V_{il}) = \Phi(\sigma_y s_{il} V_{il})$ due to the symmetry of Φ around zero. Note that $p(\mathcal{S}|\mathbf{Y})$ is maximized when the columns of $\mathbf{Y}$ are orthogonal and images of the same semantic class have the same binary embeddings. The prior term of (4.1) can be designed as a Gaussian process model for binary classification as described in the following section.

4.2.3 The Gaussian Process Model for Binary Classification

In this section we briefly describe Gaussian Process Classification (GPC); for more details see [14, 57, 58]. The GPC model is a discriminative Bayesian classifier

that models $p(y|\mathbf{x})$ as a Bernoulli distribution for a given data point $\mathbf{x}$. The class membership probability is characterized by an underlying latent function $f(\mathbf{x})$. The value of the latent function is mapped into the unit interval by a sigmoid function $\sigma : \mathbb{R} \to [0, 1]$ such that the probability $p(y = +1 \,|\, \mathbf{x})$ becomes $\sigma(f(\mathbf{x}))$. We again prefer the probit model $p(y = +1 \,|\, \mathbf{x}) = \Phi(f(\mathbf{x}))$ due to analytical convenience of the inference algorithm.

In GPH, there exist m latent functions, one for each bit j in H, which are assumed to be a priori independent. Let $\mathbf{F}$ be a random real-valued matrix of size $n \times m$ where $F_{ij} = f_j(\mathbf{x}_i)$ and $\mathbf{f}_j$ be the jth column of $\mathbf{F}$ *i.e.* shorthand for the values of the latent functions; then the binary codes $\mathbf{Y}$ are independent Bernoulli variables conditioned on $\mathbf{F}$, so the joint likelihood factorizes as

$$p(\mathbf{Y}|\mathbf{F}) = \prod_{j=1}^{m} p(\mathbf{y}_j|\mathbf{f}_j) = \prod_{j=1}^{m} \prod_{i=1}^{n} \Phi(Y_{ij} F_{ij}) \tag{4.3}$$

We place a zero mean Gaussian process prior on each latent function f_j to obtain $p(y_j = +1 \,|\, \mathbf{x}) = \frac{1}{2}$ for $j = 1, \ldots, m$ [14]. Note that the probability for each similarity label $p(s_{il} = +1|\mathbf{x}_i, \mathbf{x}_l)$ eventually becomes $\frac{1}{2}$ as well.

Through this stochastic process, each data point $\mathbf{x}_i$ is associated with m random variables $\{f_j(\mathbf{x}_i)\}_{j=1}^{m}$. Considering the entire dataset $\mathcal{X}$, we therefore have m independent multivariate Gaussian distributions. The joint distribution of latent function values for the jth bit is $p(\mathbf{f}_j|\mathcal{X}) = \mathcal{N}(\mathbf{f}_j|\mathbf{0}, \mathbf{K})$ where the covariance matrix is constructed from a kernel function $K_{il} = k(\mathbf{x}_i, \mathbf{x}_l)$ that depends on a set of kernel hyperparameters $\boldsymbol{\theta}$. In our experiments, we use the squared exponential covariance

function with the isotropic distance measure of the form:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2\ell^2}\|\mathbf{x} - \mathbf{x}'\|^2\right)$$

such that $\boldsymbol{\theta} = \{\sigma_f, \ell\}$ where σ_f^2 and ℓ are referred as the signal variance and the characteristic length-scale, respectively.

Given $\Theta = \{\sigma_y, \boldsymbol{\theta}\}$, we can compute the posterior distribution over the latent function values using the likelihood (4.3) and Bayes' rule as

$$p(\mathbf{F}|\mathcal{D}) = \frac{p(\mathcal{S}|\mathbf{F})p(\mathbf{F}|\mathcal{X})}{p(\mathcal{D})} = \frac{\sum_{\mathbf{Y}} p(\mathcal{S}|\mathbf{Y})p(\mathbf{Y}|\mathbf{F})p(\mathbf{F}|\mathcal{X})}{p(\mathcal{D})}. \tag{4.4}$$

Unfortunately, neither the posterior $p(\mathbf{F}|\mathcal{D})$ nor the marginal likelihood $p(\mathcal{D})$ can be computed analytically. Therefore, we approximate the posterior $p(\mathbf{F}|\mathcal{D})$ by joint Gaussian distributions $q(\mathbf{F}) = \prod_{j=1}^{m} \mathcal{N}(\mathbf{f}_j|\mathbf{m}_j, \mathbf{A}_j)$. Note that the approximate posterior distribution for each bit is independent of other bits. The details of the Gaussian approximation are presented in Section 4.3.

4.2.4 Predictions for Queries

The main purpose of hashing models is to predict binary codes for queries. Prediction at a query input $\mathbf{x}_*$ is made by marginalizing out over $\mathbf{F}$ in the joint distribution $p(\mathbf{F}, \mathbf{F}_*|\mathcal{D}, \mathbf{x}_*)$. This can be done separately for each bit j similar to [14] because of the independence in the approximate posterior distribution:

$$q(f_{*j}|\mathcal{D}, \mathbf{x}_*) = \int p(f_{*j}|\mathbf{f}_j, \mathcal{X}, \mathbf{x}_*) q(\mathbf{f}_j|\mathcal{D})\, d\mathbf{f}_j = \mathcal{N}(f_{*j}|\mu_{*j}, \sigma_{*j}^2) \tag{4.5}$$

with mean and variance:

$$\mu_{*j} = \mathbf{k}_*^\top \mathbf{K}^{-1} \mathbf{m}_j \tag{4.6a}$$

$$\sigma_{*j}^2 = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^\top (\mathbf{K}^{-1} - \mathbf{K}^{-1} \mathbf{A}_j \mathbf{K}^{-1})\, \mathbf{k}_* \tag{4.6b}$$

where $\mathbf{k}_* = [k(\mathbf{x}_1, \mathbf{x}_*), \ldots, k(\mathbf{x}_n, \mathbf{x}_*)]^\top$ is a vector of prior covariances between the query input $\mathbf{x}_*$ and training inputs $\mathcal{X}$. Finally, the approximate predictive distribution for each binary encoding is given as follows [57]:

$$\begin{aligned} q(y_{*j} | \mathcal{D}, \mathbf{x}_*) &= \int p(y_{*j} | f_{*j})\, q(f_{*j} | \mathcal{D}, \mathbf{x}_*)\, df_{*j} \\ &= \int \Phi(y_{*j}\, f_{*j}) \mathcal{N}(f_{*j} | \mu_{*j}, \sigma_{*j}^2)\, df_{*j} \\ &= \Phi\left(\frac{y_{*j} \mu_{*j}}{\sqrt{1 + \sigma_{*j}^2}} \right). \end{aligned} \tag{4.7}$$

This probability is maximized for each binary encoding by

$$\hat{y}_{*j} = \underset{y_{*j} \in \{\pm 1\}}{\arg\max}\; q(y_{*j} | \mathcal{D}, \mathbf{x}_*) = \operatorname{sgn}(\mu_{*j}) \tag{4.8}$$

As a result, we introduce our hash function in the form of binary encodings using (4.6a):

$$h_j(\mathbf{x}_*) = \operatorname{sgn}(\mathbf{w}_j^\top \mathbf{k}_*) \tag{4.9}$$

where $\mathbf{w}_j = \mathbf{K}^{-1} \mathbf{m}_j$, for $j = 1, \ldots, m$. Our has function consists of only multiplication between weight matrix $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_m]$ and query kernel vector $\mathbf{k}_*$, and then sign operation. Consequently, our hash function do not contain the costly cumulative distribution function Φ.

4.2.5 Sparse Approximation

To accelerate training and query times, we employ a sparse approximation to the full Gaussian process known as the fully independent training conditional (FITC) approximation [59, 60]. Let $\bar{\mathcal{X}} = \{\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_r\}$, each $\bar{\mathbf{x}}_i \in \mathbb{R}^d$, which might be a subset of $\mathcal{X}$, and associated latent function values $\mathbf{U}$, analogous to $\mathbf{F}$. Then we obtain the FITC approximation for each bit j as below:

$$\begin{aligned} p(\mathbf{f}_j, f_{*j} | \mathcal{X}, \bar{\mathcal{X}}, \mathbf{x}_*) &= \int p(\mathbf{f}_j, f_{*j} | \mathbf{u}_j, \mathcal{X}, \mathbf{x}_*) p(\mathbf{u}_j | \bar{\mathcal{X}}) \, d\mathbf{u}_j \\ &\approx \int q(\mathbf{f}_j | \mathbf{u}_j, \mathcal{X}) q(f_{*j} | \mathbf{u}_j, \mathbf{x}_*) p(\mathbf{u}_j | \bar{\mathcal{X}}) \, d\mathbf{u}_j \end{aligned} \tag{4.10}$$

where $p(\mathbf{u}_j | \bar{\mathcal{X}}) = \mathcal{N}(\mathbf{u}_j | \mathbf{0}, \mathbf{K_{uu}})$. Marginalizing over the exact prior on $\mathbf{u}_j$ in the final expression yields

$$q(\mathbf{f}_j | \mathcal{X}) = \mathcal{N}\big(\mathbf{f}_j | \mathbf{0}, \mathbf{Q_{ff}} + \mathsf{diag}(\mathbf{K_{ff}} - \mathbf{Q_{ff}})\big) \tag{4.11}$$

where $\mathbf{Q_{ab}} = \mathbf{K_{au}} \mathbf{K_{uu}^{-1}} \mathbf{K_{ub}}$ for $j = 1, \ldots, m$. The computations of the FITC approximation for binary classification are explained thoroughly in [61]. We can define binary encodings similar to those of (4.9) for the FITC model as well. However, the hyperplanes $\mathbf{w}_j \in \mathbb{R}^r$ and $\mathbf{k}_*$ will be defined between the query input $\mathbf{x}_*$ and inducing inputs $\bar{\mathcal{X}}$ since training data and queries are conditionally independent given $\mathbf{U}$ in the FITC approximation. Therefore, the communication between them is established through the bottleneck of inducing inputs [61]. Similarly, we employ a sparse set in place of all pairwise similarities, $\mathcal{S}$, in (4.2). We randomly sample t images, called *representatives*, from the dataset and use the pairwise similarities only between those representatives and the entire dataset. The sparse similarity

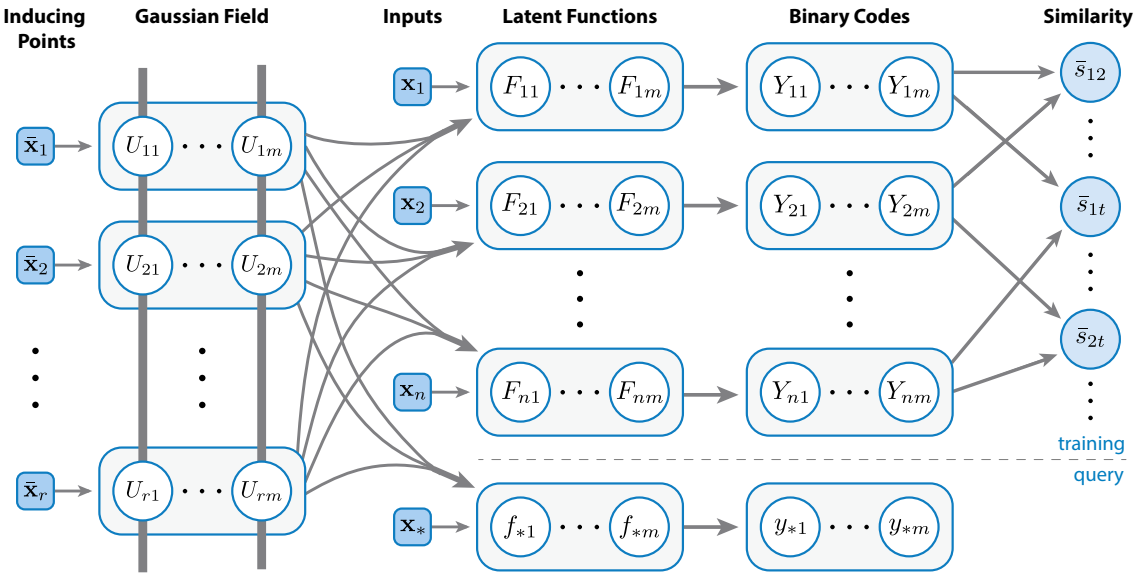


Figure 4.1: Graphical model for the Gaussian Process Hashing where circles indicate random variables, shaded circles denote observed values. The thick vertical bars represent a set of fully connected nodes.

set is denoted by $\bar{\mathcal{S}}$. By these sparsity changes, the time complexity of Gaussian process hashing is reduced from $\mathcal{O}(n^3m)$ to $\mathcal{O}\big(nm(r^2+t)\big)$. The graphical model of our hashing approach with sparse approximation is shown in Figure 4.1.

4.3 Inference

We follow the inference approach from [62], which is a hybrid of message passing and Gibbs sampling. This method has some advantages over a compound inference regarding convergence and efficiency. Our inference algorithm alternates between two phases:

- Updating site parameters of the approximate distribution by Expectation Propagation (EP) [63] for each bit j in parallel, and

- Sampling each entry of $\mathbf{Y}$ by Gibbs sampling.

The main idea behind the EP algorithm is to minimize the Kullback-Leibler (KL) divergence, which is achieved by matching of moments, at each step by adjusting site parameters. The details of EP are presented broadly by Gelman *et al.* [64]. In the first phase of our inference algorithm, the problem is reduced to Gaussian process classification since we know the values of the binary matrix $\mathbf{Y}$ thanks to the Gibbs sampler. Therefore, we adopt the scalable inference scheme of [65], which is defined in a distributed and stochastic fashion, for the FITC model. We apply that scalable EP algorithm in a serial and stochastic setting for a full update of site parameters for each bit j in parallel.

In the second phase, we sample each Y_{ij} bit-by-bit using (4.2) via

$$p(Y_{ij}|\mathbf{Y}_{-ij}, \mathcal{D}) \propto p(Y_{ij}|\mathbf{Y}_{-ij}) p(\bar{\mathcal{S}}|\mathbf{Y}) \approx q^{\backslash ij}(Y_{ij}) p(\bar{\mathcal{S}}|\mathbf{Y}) \tag{4.12}$$

where $\mathbf{Y}_{-ij}$ denotes entries of $\mathbf{Y}$ other than Y_{ij} and $q^{\backslash ij}(Y_{ij})$ denotes a tilted distribution for approximating the posterior from the FITC model. The probability $q^{\backslash ij}(Y_{ij})$ has the form $\Phi(\gamma_{ij} Y_{ij})$ where γ_{ij} is a site parameter computed by the scalable EP algorithm (See [65] for details).

Our hashing method is illustrated in Figure 4.2 for $m = 2$ on a two-dimensional sample dataset of 200 points with four equal-sized classes and 30 inducing points randomly selected from the dataset. To make our inference algorithm clearer, we provide the details for the full GPC model as [14], which is equivalent to $\bar{\mathcal{X}} = \mathcal{X}$, in the following section.

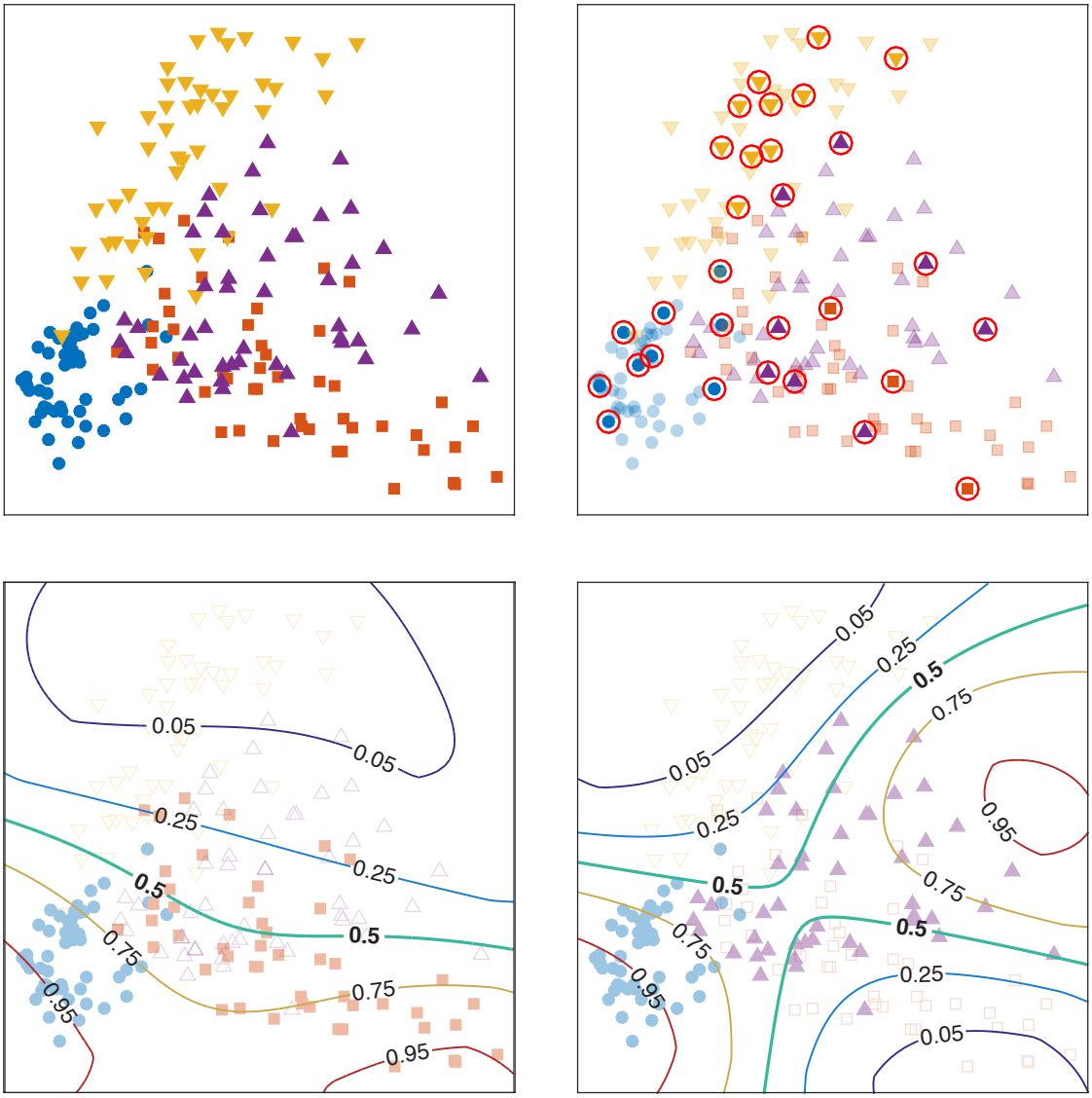


Figure 4.2: From left to right and top to bottom; a sample dataset of 4 classes indicated by color and shape $\mathcal{X}$, inducing points marked with red circles $\bar{\mathcal{X}}$, the predictive probability distribution for the first binary encoding $p(y_{*1}|\mathcal{D}, \mathbf{x}_*)$ and for the second encoding $p(y_{*2}|\mathcal{D}, \mathbf{x}_*)$. The binary codes $\mathbf{Y}$ by the Gibbs sampler are indicated by filled and emptied shapes for $+1$ and -1, respectively.

4.3.1 Inference for the Full GPC Model

We start deriving Gaussian approximations for (4.4) by replacing each $p(Y_{ij}|F_{ij}) = \Phi(Y_{ij}F_{ij})$ by a site function $t_{ij}(Y_{ij}, F_{ij}) = Z_{ij}\,\Phi(\gamma_{ij}Y_{ij})\mathcal{N}(F_{ij}\,|\,\mu_{ij}, \sigma_{ij}^2)$ where Z_{ij} is a normalizing constant and $\{\mu_{ij}, \sigma_{ij}^2, \gamma_{ij}\}$ are site parameters. Given $\mathcal{D}$, the latent function values $\mathbf{F}$ and binary codes $\mathbf{Y}$ become independent by means of site functions. The messages between them must pass through the site parameters. Let $q(\mathbf{Y}) = \prod_{j=1}^{m}\prod_{i=1}^{n}\Phi(\gamma_{ij}Y_{ij})$, then we can find an approximation to the joint posterior distribution of latent variables as follows:

$$\begin{aligned} p(\mathbf{Y}, \mathbf{F}\,|\,\mathcal{D}) &\approx \frac{p(\mathcal{S}|\mathbf{Y})\left(\prod_{j=1}^{m}\prod_{i=1}^{n} t_{ij}(Y_{ij}, F_{ij})\right)p(\mathbf{F}|\mathcal{X})}{q(\mathcal{D})} \\ &= \frac{p(\mathcal{S}|\mathbf{Y})q(\mathbf{Y})}{q(\mathcal{D})}\prod_{j=1}^{m}\mathcal{N}(\mathbf{f}_j|\mathbf{m}_j, \mathbf{A}_j) \\ &= q(\mathbf{Y}|\mathcal{D})q(\mathbf{F}) \end{aligned} \tag{4.13}$$

where $q(\mathcal{D}) = \sum_{\mathbf{Y}} p(\mathcal{S}|\mathbf{Y})\,q(\mathbf{Y})$ can be computed using the Poisson binomial distribution. The Gaussian approximations $q(\mathbf{f}_j) = \mathcal{N}(\mathbf{f}_j\,|\,\mathbf{m}_j, \mathbf{A}_j)$ have mean and covariance:

$$\mathbf{m}_j = \mathbf{A}_j\boldsymbol{\Sigma}_j^{-1}\boldsymbol{\mu}_j \quad \text{and} \quad \mathbf{A}_j = (\mathbf{K}^{-1} + \boldsymbol{\Sigma}_j)^{-1} \tag{4.14}$$

where $\boldsymbol{\mu}_j = (\mu_{1j}, \ldots, \mu_{nj})^\top$ and $\boldsymbol{\Sigma}_j = \mathsf{diag}(\sigma_{1j}^2, \ldots, \sigma_{nj}^2)$ collect site parameters.

In the first phase, EP finds Gaussian approximations $q(\mathbf{f}_j) = \mathcal{N}(\mathbf{f}_j|\mathbf{m}_j, \mathbf{A}_j)$ to the conditional distribution $p(\mathbf{f}_j|\mathbf{Y})$ by moment matching of approximate marginal distributions [14], for $j = 1, \ldots, m$ in parallel, thanks to their conditional indepen-

dence. In the second phase, we sample Y_{ij} using (4.2) via

$$q(Y_{ij}|\mathbf{Y}_{-ij}, \mathcal{D}) \propto \Phi(\gamma_{ij} Y_{ij}) p(\mathcal{S}|\mathbf{Y}). \tag{4.15}$$

We update the site parameters iteratively on each processor j so that the first and second moments of the following expressions on both sides are matched given $\mathbf{Y}$:

$$q_{-ij}(F_{ij}) p(Y_{ij}|F_{ij}) \simeq q_{-ij}(F_{ij}) t_{ij}(Y_{ij}, F_{ij}) \tag{4.16}$$

where $q_{-ij}(F_{ij})$, which is referred to as an approximate cavity distribution, is defined as

$$q_{-ij}(F_{ij}) = \int p(\mathbf{F}|\mathcal{X}) \prod_{j' \neq j} \prod_{i' \neq i} t_{i'j'}(Y_{i'j'}, F_{i'j'}) \, d\mathbf{F}_{-ij} \propto \mathcal{N}(m_{j,i}, A_{j,ii}) \tag{4.17}$$

with cavity parameters are

$$\sigma^2_{-ij} = \left((A_{j,ii})^{-1} - \sigma_{ij}^{-2}\right)^{-1}, \tag{4.18a}$$

$$\mu_{-ij} = \sigma^2_{-ij} \left(\frac{m_{j,i}}{A_{j,ii}} - \frac{\mu_{ij}}{\sigma^2_{ij}} \right). \tag{4.18b}$$

Let $z_{ij} = Y_{ij}\mu_{-ij}/\sqrt{1+\sigma^2_{-ij}}$, then the first and the second moments of the left hand side of (4.16) are given by

$$m_1 = \mu_{-ij} + \frac{Y_{ij}\sigma^2_{-ij}\mathcal{N}(z_{ij}|0,1)}{\Phi(z_{ij})\sqrt{1+\sigma^2_{-ij}}} \tag{4.19a}$$

$$m_2 = 2\mu_{-ij}m_1 - \mu^2_{-ij} + \sigma^2_{-ij} - \frac{z_{ij}\sigma^4_{-ij}\mathcal{N}(z_{ij}|0,1)}{\Phi(z_{ij})\,(1+\sigma^2_{-ij})}. \tag{4.19b}$$

To obtain the same moments for the right hand side of (4.16), we update the site

parameters as below:

$$\gamma_{ij} = \frac{\mu_{-ij}}{\sqrt{1+\sigma^2_{-ij}}}, \tag{4.20a}$$

$$\sigma^2_{ij} = \big((m_2 - m_1^2)^{-1} - \sigma^{-2}_{-ij}\big)^{-1}, \tag{4.20b}$$

$$\mu_{ij} = m_1 + \sigma^2_{ij}\sigma^{-2}_{-ij}(m_1 - \mu_{-ij}), \tag{4.20c}$$

$$Z_{ij} = \sqrt{2\pi(\sigma^2_{-ij} + \sigma^2_{ij})}\exp\left(\frac{(\mu_{-ij}-\mu_{ij})^2}{2\big(\sigma^2_{-ij}+\sigma^2_{ij}\big)}\right). \tag{4.20d}$$

Finally, the approximate log marginal likelihood can be calculated from (4.2) and (4.13) as

$$\begin{aligned}\log q(\mathcal{D}, \mathbf{Y}) = &\sum_{i=2}^{n-1}\sum_{l=i+1}^{n}\log\Phi(\sigma_y V_{il}) - \frac{nm}{2}\log(2\pi) \\ &+ \sum_{j=1}^{m}\left(\sum_{i=1}^{n}\log\big(Z_{ij}\Phi(\gamma_{ij}Y_{ij})\big) - \frac{1}{2}\log|\mathbf{K}+\Sigma_j| - \frac{1}{2}\mu_j^\top(\mathbf{K}+\Sigma_j)\mu_j\right)\end{aligned} \tag{4.21}$$

The computational complexity of the serial hybrid approach is $\mathcal{O}(n^3)$ for the full GPC model.

4.4 Experiments

We compared the retrieval performance of our hashing method with several state-of-the-art supervised hashing techniques including CCA-ITQ [5], KSH [46], FastHash [49] and SDH [52]. We use the public code provided by the authors with their suggested parameters unless otherwise specified. The experiments were performed on three image datasets, namely, CIFAR-10, MNIST and NUS-WIDE datasets in the MATLAB environment on a machine with a 2.8 GHz Intel Core i7 CPU and 16GB RAM.

4.4.1 Datasets and Experimental Setup

For the NUS-WIDE dataset only, we consider two images as semantically similar if there exists at least one common associated concept as [52]. All datasets were first centered at zero and then each point vector was normalized to unit length. For NUS-WIDE, we constructed a reduced test set by sampling 10,000 images associated with any of the most frequent 21 labels for all methods. For the KSH method, we used reduced datasets of 5,000 images uniformly sampled from the training sets of each dataset because the computational complexity of this approach does not allow it to be trained on the entire datasets as we do with other methods (Table 4.3). Similarly, the tree depth of FastHash was set to 2 due to its longer test time (Table 4.3). FastHash was not trained on that dataset due to its large memory requirements. GPH and KSH were used on pairwise similarities on the NUS-WIDE dataset while SDH and CCA-ITQ were trained on label information on that dataset. The ℓ_2-loss version of SDH was employed in our experiments since there exist no predefined classes for the NUS-WIDE dataset. We uniformly sampled 1000 inducing points (anchor points) from the training set of each dataset. These points were shared by all hashing methods with kernels (GPH, KSH, and SDH) in training. All these methods used an RBF kernel with a kernel width ℓ adjusted specifically for each dataset by cross-validation for each method with a kernel.

Hash functions learned from a subset of a large-scale image dataset can be used for computing the binary codes for the entire dataset. Next, these binary codes can be utilized for efficient large-scale image search in the Hamming space [2].

Therefore, generalization ability of supervised hashing techniques is critical for the retrieval performance.

For GPH, we adjusted the hyperparameters σ_f and σ_y by cross-validation on the training set. However, the performance is usually maximized at $\sigma_y = 2/m$. For all datasets, 5,000 representative images were randomly selected for $\bar{\mathcal{S}}$. The scalable EP algorithm was run in a stochastic fashion with a block size of 1,000 images. The binary codes $\mathbf{Y}$ were initialized randomly. The GPH inference algorithm was executed until convergence or at most 50 sweeps. In learning hash functions by the GPH model, we observe that the Gibbs sampler for $\mathbf{Y}$ rapidly converges after assigning discriminative binary codes to all classes. The rest of the inference algorithm focuses on training GPs with the scalable EP algorithm on these *fixed* binary codes. A pattern of binary codes emerges even for most values of σ_y although some bit assignments might change during the inference algorithm.

After learning hash functions from training data, we computed binary codes for images in the test sets where each dataset contains 10,000 images. Retrieval performance was evaluated by leave-one-out validation on these pictures only. Each test image was used once as a query while the remaining test images were turned into a retrieval set. As overfitting is a common issue for supervised hashing, this methodology was employed to assess generalization performance of the hashing methods on images that were not used in training, analogous to new images being added to a database or a vast dataset from which only a subset of images can be employed for learning hash functions.

4.4.2 Experimental Results

For each query, images were ranked concerning Hamming distances between their binary codes and that of the given query. Ground truth labels were defined by semantic similarity from either class labels or associated tag information. For quantitative analysis, we report retrieval performance in mean average precision (mAP) for all datasets in Figure 4.3. Precision-recall curves of all methods on the CIFAR-10 datasets are demonstrated in Figure 4.4 for a different length of hash codes. The mean of precision and recall (%) at the Hamming radius $r = 2$ are shown for all methods on three different datasets in Tables 4.1 and 4.2, respectively. Precision and recall values for 64 and 128 bits were not evaluated because this evaluation is impractical for longer binary codes. Note that we used a zero score in the calculation of mean precision for queries for which no image exists inside the Hamming ball with radius 2.

Our method GPH outperforms the state-of-the-art supervised hashing methods regarding mAP and mean precision at Hamming distance 2 for smaller binary codes (32-bits or shorter) on all datasets. Some methods perform better than GPH in some cases on the training set (data not shown). However, GPH outperforms on the independent set due to the better generalization as a result of averaging over nonlinear classifiers. Although SDH performs well on the CIFAR-10 and MNIST datasets, its performance on the NUS-WIDE dataset where there are no predefined classes is poor. On the other hand, GPH performs the best on that dataset with a large margin. As the number of bits increases, the approximation error due to our

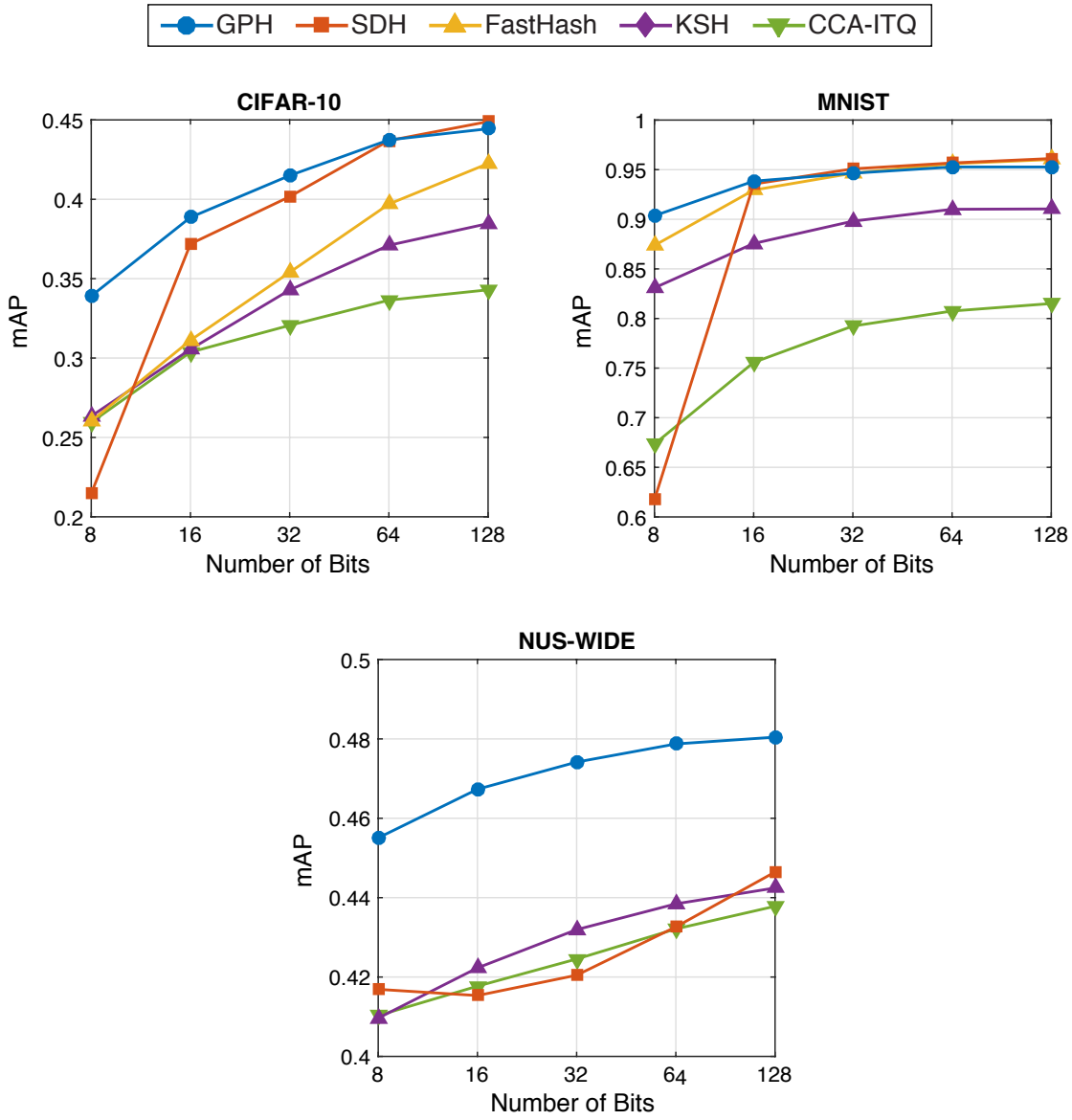


Figure 4.3: Our method (GPH) is compared with the state-of-the-art methods on the CIFAR-10, MNIST and NUS-WIDE datasets in terms of mean average precision (mAP), respectively.

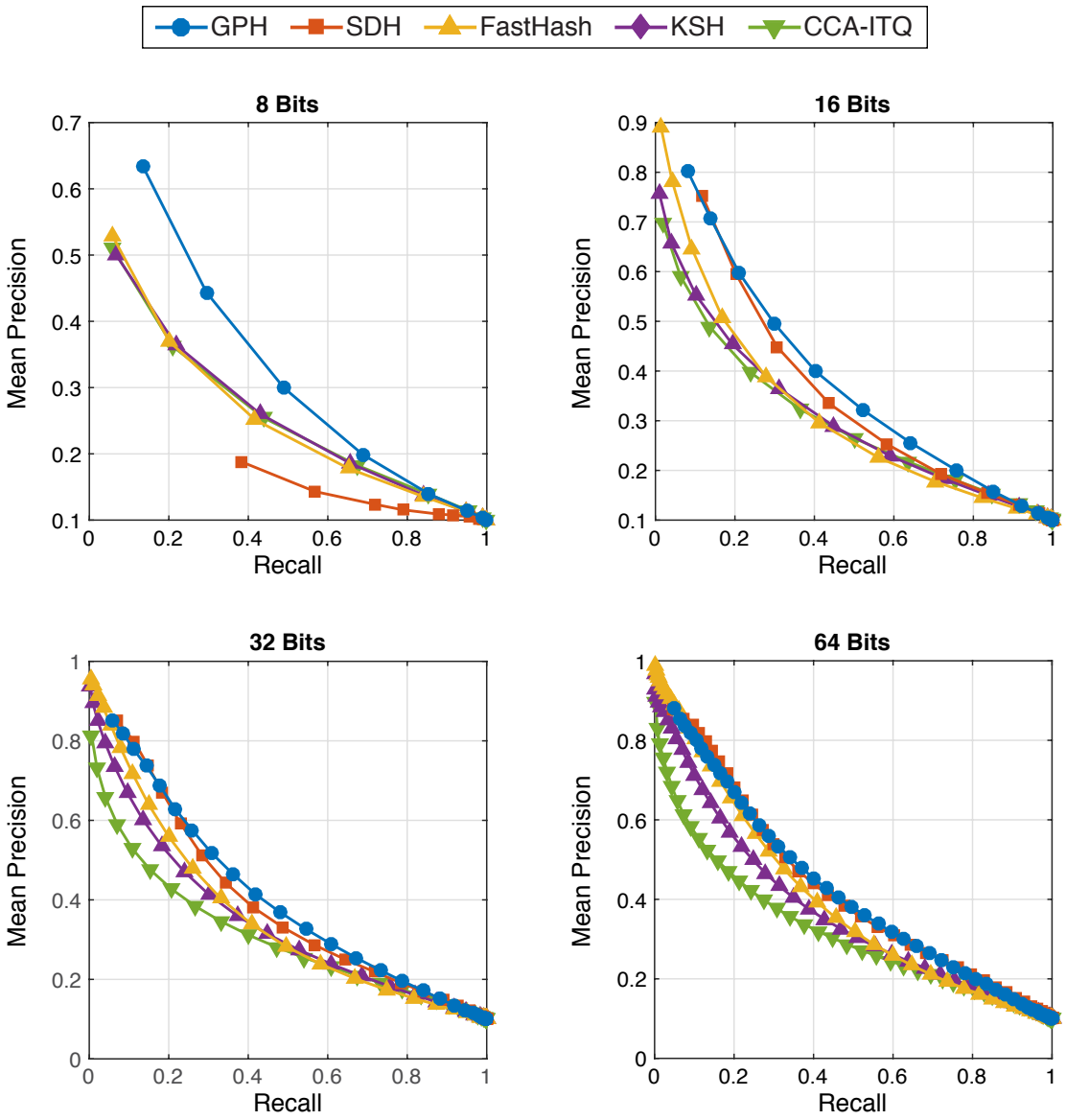


Figure 4.4: Our method (GPH) is compared with the state-of-the-art methods on the CIFAR-10 datasets by precision-recall curves for 8, 16, 32 and 64 bit hash codes.

Table 4.1: Our method (GPH) is compared in terms of the mean of precision (%) at the Hamming radius $r = 2$.

Method	**CIFAR-10**			**MNIST**			**NUS-WIDE**		
	8-bits	16-bits	32-bits	8-bits	16-bits	32-bits	8-bits	16-bits	32-bits
GPH	**30.89**	**46.15**	46.03	**83.49**	93.37	**92.09**	**44.46**	48.26	47.52
SDH	15.35	42.28	**48.09**	28.45	82.67	83.42	40.89	45.95	**48.40**
FastHash	25.44	44.00	26.65	77.42	**93.65**	87.45	-	-	-
KSH	25.99	40.73	31.68	70.68	89.27	87.06	42.99	46.76	26.89
CCA-ITQ	25.20	38.93	41.36	56.36	81.63	81.83	44.07	**48.48**	25.84

Table 4.2: Our method (GPH) is compared in terms of the mean of recall (%) at the Hamming radius $r = 2$.

Method	**CIFAR-10**			**MNIST**			**NUS-WIDE**		
	8-bits	16-bits	32-bits	8-bits	16-bits	32-bits	8-bits	16-bits	32-bits
GPH	48.90	21.01	11.08	**91.36**	**83.80**	**77.62**	47.86	**29.01**	**21.00**
SDH	**71.96**	**30.54**	**14.62**	86.19	78.28	73.10	**57.60**	24.83	4.03
FastHash	41.61	9.11	2.07	90.31	76.13	59.09	-	-	-
KSH	42.98	10.06	2.02	88.48	71.29	55.21	29.21	6.03	0.98
CCA-ITQ	43.71	13.56	4.01	77.52	51.49	34.29	27.50	4.43	0.57

assumption of conditional independence between GPs is growing as well. Therefore, the improvement in retrieval performance by longer codes for the GPH is moderate. For precision and recall at Hamming radius 2, our method and SDH shares the best performances on most cases.

Table 4.3 shows the comparative retrieval performance of our method with different training and anchor set sizes along with the state-of-the-art methods on the CIFAR-10 dataset for 16-bit hash codes regarding mAP and mean precision at Hamming radius 2. We also report the execution time for learning hashing functions as training time and the time needed for computing binary code of a single query using the learned hashing functions as test time. As expected, larger training and inducing sets provide better performance with longer execution time. The number of inducing points has a greater influence on retrieval performance than training size. Note that the scale of the inducing set also affects the test time. Our method outperforms other methods in retrieval performance. Note that Table 4.3 shows the training time on a four-core machine and our method learns its hash function by a parallel algorithm. The training time in Table 4.3 can be improved by a computer with a larger number of nodes. Therefore, GPH is more efficient and effective in retrieval when binary codes are short. The most efficient method in the experiments is CCA-ITQ; however, it performs poorly in retrieval. On the other hand, SDH is the second most efficient method with relatively good retrieval performance. For testing, all kernel methods (GPH, SDH, and KSH) have a similar number of operations: Kernel vector computation, multiplication by weight matrix and kernel vector and finally sign operation. Therefore, they have similar test time per query. ITQ, which

has no kernel computation, has faster test time. On the other hand, FastHash has longer test time due to its computations based on decision trees.

4.5 Conclusion

We proposed a supervised retrieval scheme based on Gaussian processes for classification. We developed an efficient inference algorithm for the proposed model. The experimental results on three real-world image datasets show that our method produces the best retrieval performance for smaller binary codes by preventing overfitting to training data. Also, it provides a predictive probability distribution for each bit.

Table 4.3: Results in mean average precision (mAP), mean of precision at Hamming radius $r = 2$, training and test time for 16-bit hash codes on the CIFAR-10 dataset. For our method (GPH), the number of inducing samples varies from 300 to 3,0000. The experiments were performed on a machine with an Intel quad-core processor.

Method	**Training Set Size**	**Inducing Set Size**	**mAP**	**Precision at** $r = 2$	**Training Time** (s)	**Test Time** (μs)
GPH	5,000	300	0.279	0.365	26.5	7.7
	5,000	500	0.304	0.387	37.6	12.3
	5,000	1,000	0.309	0.395	78.5	21.6
	5,000	3,000	0.338	0.416	456.1	44.7
	50,000	300	0.311	0.403	177.4	7.6
	50,000	1,000	**0.395**	**0.466**	497.9	18.9
SDH	5,000	1,000	0.306	0.373	0.9	17.1
	50,000	1,000	0.372	0.423	7.5	16.5
Fasthash	5,000	-	0.240	0.357	24.6	47.5
	50,000	-	0.311	0.440	355.5	68.0
KSH	5,000	1,000	0.305	0.408	1,461.6	25.0
CCA-ITQ	5,000	-	0.262	0.353	0.1	0.2
	50,000	-	0.304	0.389	0.3	0.2

Chapter 5: Dynamic Hashing

Nothing endures but change.

Heraclitus of Ephesus

5.1 Overview

In this chapter, we propose an incremental supervised hashing method with kernels based on binary and multi-class SVMs, which we refer to as *SVM-based Hashing* (SVM-Hash). We adopt the incremental learning fashion of SVMs in a hashing framework. Consequently, the proposed hashing method can be easily extended to unseen classes incrementally. We identified three main issues that limit the performance of Supervised Discrete Hashing (SDH) [52] – as not being incremental and parallelizable, overfitting to training data and imbalance of $-1/+1$ in learned binary codes. The significance of balanced binary codes has been studied in the hashing literature [7, 3, 30, 29] and its absence leads to ineffective codes, especially when the code length is small and/or the sizes of semantic classes are unbalanced. To overcome these issues, we reformulated the supervised hashing task as a two-stage classification framework. In the first stage of classification, we use a binary SVM for each bit. The feature space is the input space of SVMs while

binary codes are considered as class labels. In the second stage, the binary codes become the input space of a single multi-class SVM, and the semantic space is the output space of the SVM. Besides, we penalize imbalance in binary codes in our optimization formulation. We define an incremental strategy to learn SVMs and a *discrete cyclic coordinate descent* (DCC) algorithm, similar to the one of SDH, to learn binary codes bit by bit as an approximate solution to the NP-hard discrete optimization problem. Finally, we describe an incremental procedure for modifications to the database.

5.2 Incremental Hashing

5.2.1 Learning Hash Functions for Dynamic Databases

Modifications to an image database consist of adding new images to the database and deleting images from the database. Considering a database with class information, we say that there are four types of modifications:

1. Adding new classes,
2. Adding images to existing classes,
3. Deleting existing classes,
4. Deleting images from existing classes.

In this chapter, we focus on the first three types of modifications. From a hashing perspective, the last type of modifications is the least interesting one because

it technically results in reducing the training size. For some cases like biometric databases, in which each class corresponds to a single person, deleting images may aim to change the class structure. For learning hash functions, such cases can be interpreted as deleting that class completely; and then adding the rest of them as a new class. Among the other modification types, deleting existing classes is the easiest case because we have already learned binary codes for all images in the final database. In contrast, adding new classes is the hardest case since we have no prior information about the new classes.

We define three hashing strategies for dynamic datasets:

1. *Passive strategy*: Continuing to use the hash functions learned from an earlier state of the database *i.e.* ignoring the changes in training data.

2. *From-scratch strategy*: Learning hash functions on the final state of the database from scratch whenever a change occurs in the database.

3. *Incremental strategy*: Learning hash functions on the last state of the database incrementally from the previous hash functions.

The incremental strategy can be considered as effective and efficient if its retrieval performance is similar to that of the from-scratch strategy and its training time is shorter than that of the from-scratch strategy. We first describe our supervised hashing method; next, we provide an incremental hashing strategy on that method in the following sections.

5.2.2 Supervised Discrete Hashing

Shen *et al.* [52] recently introduced the SDH method that utilizes binary codes as input data in a linear classification framework. The SDH method searches for optimal set of binary codes $\{\mathbf{b}_i\}_{i=1}^n$ where $\mathbf{b}_i = [b_{i1}, \ldots, b_{im}]^\top$ corresponds to the ith data point in the training set. Let $\mathcal{Y}$ denote the set of ground truth semantic classes and $y_i \in \mathcal{Y}$ denote the class label of the ith data point; then SDH defines a multi-class SVM model that maps binary codes to semantic classes *i.e.* $\mathbf{b}_i \mapsto y_i$ for $i = 1, \ldots, n$. SDH simultaneously approximates $\mathbf{b}_i$ by a nonlinear embedding $F(\mathbf{x}_i) = \mathbf{P}^\top \boldsymbol{\varphi}(\mathbf{x}_i)$ where $\mathbf{P} \in \mathbb{R}^{s \times m}$. Finally, the hash function is defined by $H(\mathbf{x}) = \text{sign}(F(\mathbf{x}))$. The optimization is formulated as a combination of the multi-class linear SVM by Crammer and Singer [66] and least squares for the nonlinear embedding as follows:

$$\begin{aligned} \min_{\mathbf{B},\mathbf{W},F,\boldsymbol{\xi}} \quad & \lambda \sum_{k \in \mathcal{Y}} \|\mathbf{w}_k\|^2 + \sum_{i=1}^{n} \xi_i + \nu \sum_{i=1}^{n} \|\mathbf{b}_i - F(\mathbf{x}_i)\|^2 \\ \text{s.t.} \quad & \forall (i,k) \quad (\mathbf{w}_{y_i} - \mathbf{w}_k)^\top \mathbf{b}_i \geq \mathbb{1}[y_i \neq k] - \xi_i, \\ & \mathbf{b}_i \in \{-1, +1\}^m \end{aligned} \tag{5.1}$$

where λ and ν are regularization parameters, ξ_i is a slack variable corresponding to the hinge loss for data point i and $\mathbb{1}[\cdot]$ is an indicator function. This optimization problem is solved by an iterative algorithm that fixes all but one variable at each step.

The SDH approach outperforms the state-of-the-art hashing methods in most cases [52]. However, it has two main drawbacks. Many unsupervised hashing techniques [5, 7] are designed to yield balanced bits such that the number of +1 and

-1 is equal or closer, as this results in better retrieval performance. Unfortunately, SDH lacks a control mechanism for the distribution of -1 and $+1$ over the data points of each bit. We observe that SDH performs poorly due to imbalanced bits when binary codes are shorter *e.g.* smaller than 32-bits or some classes are enormous in size while some classes are tiny. The other drawback is overfitting to the optimal binary codes $\{\mathbf{b}_i\}_{i=1}^n$ in the nonlinear embedding $F(\mathbf{x})$. Also, each bit of $F(\mathbf{x})$ cannot be distributed among different cores in a machine. Furthermore, no strategy is presented for updating $F(\mathbf{x})$ to unseen classes.

5.2.3 SVM-based Hashing

To overcome the issues described in the previous section, we define a new optimization task that replaces the approximation loss in (5.1) with m independent binary SVMs as [47]. Let $\mathbf{B} \in \{-1, +1\}^{n \times m}$ be a matrix that collects binary codes where the ith row of $\mathbf{B}$ is equal to $\mathbf{b}_i^\top$. We can then define a binary SVM model treating $\{\boldsymbol{\varphi}(\mathbf{x}_i)\}_{i=1}^n$ as input data and column j of $\mathbf{B}$ as binary class labels for $j = 1, \ldots, m$. In addition, we add a new term that penalizes imbalanced assignments

of $-1/+1$ in binary codes. Our formulation for supervised hashing is given by

$$\begin{aligned}
\min_{\mathbf{B},\mathcal{W},\Xi} \quad & \lambda\left(\frac{1}{2}\sum_{k\in\mathcal{Y}}\|\mathbf{w}_k^b\|^2 + C^b\sum_{i=1}^{n}\xi_i^b\right) + \sum_{j=1}^{m}\left(\frac{1}{2}\|\mathbf{w}_j^x\|^2 + C^x\sum_{i=1}^{n}\xi_{ij}^x\right) + \gamma\sum_{j=1}^{m}\left|\sum_{i=1}^{n}b_{ij}\right| \\
\text{s.t.} \quad & \forall(i,k) \quad (\mathbf{w}_{y_i}^b - \mathbf{w}_k^b)^\top\mathbf{b}_i \geq \mathbb{1}[y_i \neq k] - \xi_i^b, \\
& \forall(i,j) \quad b_{ij}\,(\mathbf{w}_j^x)^\top\boldsymbol{\varphi}(\mathbf{x}_i) \geq 1 - \xi_{ij}^x, \\
& \forall(i,j) \quad \xi_{ij}^x \geq 0, \\
& \mathbf{b}_i \in \{-1,+1\}^m
\end{aligned} \tag{5.2}$$

where λ and γ are scaling parameters, C^x and C^b are soft margin parameters of the SVMs. Note that the sum of column j in $\mathbf{B}$ is equal to zero when bit j is balanced. There will be no penalty for bit j in this case. This imbalance penalty is also important for the performance of the binary SVMs with the fixed parameter C^x since they are sensitive to imbalance in the datasets [67]. We can add an extra element 1 to the vectors of all data points and binary codes in order to add a bias term to the loss function of SVMs. Finally, our hash function is defined as $H(\mathbf{x}) = \operatorname{sign}\big((\mathbf{W}^x)^\top\boldsymbol{\varphi}(\mathbf{x})\big)$ where the jth column of $\mathbf{W}^x \in \mathbb{R}^{d\times m}$ is equal to $\mathbf{w}_j^x$ for $j = 1,\ldots,m$. Similarly, we define another weight matrix $\mathbf{W}^b \in \mathbb{R}^{d\times m}$ such that its jth column is equal to $\mathbf{w}_k^b$ for $k = 1,\ldots,|\mathcal{Y}|$.

This optimization task can be solved by an iterative algorithm that consists of two main steps: Training SVMs and learning binary codes as follows.

5.2.4 Training SVMs

If we fix all binary codes $\mathbf{B}$, the optimization problem (5.2) will be reduced to m separate binary SVM and one multi-class linear SVM problems. These $(m+1)$ SVMs can be trained in parallel since they are conditionally independent on binary codes. Our formulation has a disadvantage of computational cost in comparison with the SDH because we need to train $(m+1)$ SVMs at each iteration until convergence. Therefore, we employ an efficient large-scale SVM learning technique called the *Optimized Cutting Plane Algorithm for SVMs* (OCAS) [68]. The OCAS algorithm is based on approximating the original convex loss function of an SVM by a piecewise linear function defined as the maximum over a set of linear under-estimators called cutting planes. At each step, a new cutting plane is added to the set with a cost of $\mathcal{O}(nd)$ complexity. Each under-estimator is a rank-one Taylor approximation to the loss function of the SVM at a weight point $\mathbf{w}'$. The method also has a multi-class counterpart called OCAM.

During the execution of our algorithm, changes in binary codes $\mathbf{B}$ from one iteration to the next one are usually small. Therefore, we take advantage of the pre-computed SVMs from the previous iteration in an incremental approach. We employ a warm start strategy for training SVMs similar to the one in [69]. We initialized the *best-so-far solution* $\mathbf{w}^b$ in the OCAS algorithm with the solution to the SVM in the previous iteration instead of a zero vector for each bit. Hence, the number of iterations in the OCAS computation is reduced with fewer cutting planes. If there exists no change in column j of $\mathbf{B}$ between iteration $(t-1)$ and iteration t,

then we do not train the corresponding SVM again. Similarly, the same approach is used for the multi-class linear SVM using the OCAM algorithm. Our hashing method converges when the entire matrix $\mathbf{B}$ remains unchanged.

5.2.5 Learning Binary Codes

In the process of learning binary codes $\mathbf{B}$, we fix all the SVM weights $\{\mathbf{w}_j^x\}_{j=1}^m$ and $\{\mathbf{w}_k^b\}_{k\in\mathcal{Y}}$ in (5.2). This results in an integer programming problem and is therefore NP-hard. As a result, we take an approach akin to the discrete cyclic coordinate descent algorithm in [52]. We learn binary codes column by column in $\mathbf{B}$ iteratively based on the rest of the binary codes until convergence. For updating bit j in $\mathbf{b}_i$ for $i = 1, \ldots, n$, we reduce (5.2) to the following optimization problem:

$$\begin{aligned} \min_{\{b_{ij}\}_{i=1}^n} \quad & \sum_{i=1}^n L(b_{ij}, i, j) + \gamma \left| \sum_{i=1}^n b_{ij} \right| \\ \text{s.t.} \quad & \forall i \quad b_{ij} \in \{-1, +1\} \end{aligned} \tag{5.3}$$

where $L(b, i, j)$ represents the total hinge losses depending specifically on data point i when the corresponding bit j is equal to $b \in \{-1, +1\}$, specifically:

$$\begin{aligned} L(b, i, j) = \beta\, C^b \max_{k\in\mathcal{Y}} \big(\mathbb{1}[y_i \neq k] + b\,(w_{k,j}^b - w_{y_i,j}^b) + \theta_{ijk} \big) \\ + C^x \max\big(0, 1 - b\,(\mathbf{w}_j^x)^\top \boldsymbol{\varphi}(\mathbf{x}_i)\big) \end{aligned}$$

where $\theta_{ijk} = \sum_{u\neq j} b_{iu}(w_{k,u}^b - w_{y_i,u}^b)$ is a bias term that depends on the binary codes excluding bit j. In case of no penalty for imbalance ($\gamma = 0$), the closed-form solution is $b_{ij} = \arg\min_{b\in\{-1,+1\}} L(b, i, j)$ for $i = 1, \ldots, n$. In our case, we need to consider the trade-off between the hinge losses and the imbalance penalty. Therefore, we sort the data points with respect to the difference $\delta_{ij} = L(-1, i, j) - L(+1, i, j)$ and find

a cutting location between $-1/+1$ assignments as follows. Let I contain the indices of data points in ascending order, then the cutting location is determined by

$$\mathsf{cut} = \arg\min_{l \in \{0,\ldots,n\}} \left(\gamma |2l - n| + \sum_{i=1}^{l} L\big(-1, I[i], j\big) + \sum_{i=l+1}^{n} L\big(+1, I[i], j\big) \right). \tag{5.4}$$

Next, we assign -1 to bit j of data points from $I[1]$ to $I[\mathsf{cut}]$ and $+1$ for data points from $I[\mathsf{cut}+1]$ to $I[n]$. Note that the first sum in (5.4) represents the cumulative hinge losses for the assignments of -1 and the second one represents the same for $+1$ assignments. We repeat these column updates for $j = 1, \ldots, m$ until $\mathbf{B}$ converges; that typically requires fewer than 10 full updates of $\mathbf{B}$ after being initialized with all $+1$'s. The proposed SVM-based Hashing (SVM-Hash) method is summarized in Algorithm 5.1.

5.2.6 Incremental Updates on Hash Functions

Our method can be efficiently updated whenever the training set has a modification. We describe the initialization of our incremental learning strategy from the easiest to the hardest type of modifications as follows:

1. *Deleting existing classes*: Let $\mathcal{Y}_d$ be the set of deleted class labels, then we remove $\mathbf{w}_k^y$ for $k \in \mathcal{Y}_d$ from the multi-class SVM $\{\mathbf{w}_k^b\}_{k \in \mathcal{Y}}$ and we remove the rows of $\mathbf{B}$ that correspond to images associated with any class in $\mathcal{Y}_d$.

2. *Adding images to existing classes*: For each class, we find the most frequent binary string pattern from $\mathbf{B}$. Next, each new image is initialized with the corresponding binary code according to its class.

Algorithm 5.1 SVM-based Hashing

Input: Training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, code length m, number of anchor points r, maximum iteration number $\mathsf{max\_iter}$, parameters C^x, C^b, λ and γ.

Output: Binary codes $\mathbf{B}$, hash function $H(\mathbf{x}) = \operatorname{sign}\big((\mathbf{W}^x)^\top \mathbf{x}\big)$.

Randomly select r anchor points $\{\mathbf{a}_i\}_{i=1}^s$ from the training data and compute the kernel function $\boldsymbol{\varphi}(\mathbf{x}_i)$ for $i = 1, \ldots, n$.
Initialize binary codes $\mathbf{B}^{(0)}$ with a random string in $\{-1, +1\}^m$ for each class.
for $t \leftarrow 1$ to $\mathsf{max\_iter}$ **or until** convergence **do**
  for $j \leftarrow 1$ to m **in parallel do**
    if column j of $\mathbf{B}^{(t-1)} \neq$ column j of $\mathbf{B}^{(t)}$, **then**
      Train binary SVM for bit j using the OCAS algorithm where the data is $\big\{\big(\boldsymbol{\varphi}(\mathbf{x}_i), b_{ij}^{(t-1)}\big)\big\}_{i=1}^n$.
    end if
  end for
  Train linear multiclass SVM on the data $\big\{\big(\mathbf{b}_i^{(t-1)}, y_i\big)\big\}_{i=1}^n$.
  Initialize $\mathbf{B}^{(t)}$ with $\{+1\}^{n \times m}$.
  repeat
    for j from 1 to m **do**
      Compute cut in (5.4)
      Update column j in $\mathbf{B}^{(t)}$ according to cut and I.
    end for
  until $\mathbf{B}^{(t)}$ convergences
end for

3. *Adding new classes*: We add new rows to $\mathbf{B}$ corresponding to new images, and we initialize each new class with a random binary string. Since we do not have multi-class SVM trained on the new classes, we only train the multi-class SVM $\{\mathbf{w}_k^b\}_{k\in\mathcal{Y}_f}$ on $\mathbf{B}_f$ from scratch where the subscript f denotes the final version.

Next, we repeat the loop in Algorithm 5.1 with one of these initializations and with the previous SVMs $\{\mathbf{w}_j^x\}_{j=1}^m$ and $\{\mathbf{w}_k^b\}_{k\in\mathcal{Y}_f}$ in an incremental fashion.

5.3 Experiments

We conducted extensive experiments to assess the effectiveness and efficiency of the proposed method, SVM-Hash, compared to the state-of-the-art supervised hashing techniques including CCA-ITQ [5], KSH [46], FastHash [49] and SDH [52]. The performance of the methods was analyzed regarding search accuracy and training/testing time on three large-scale datasets - CIFAR-10, MNIST, and NUS-WIDE. All experiments were performed in the MATLAB environment on a machine with a 2.8 GHz Intel Core i7 CPU and 16GB RAM using the public code provided by the authors with their suggested parameters unless otherwise specified.

5.3.1 Datasets and Experimental Setup

For the NUS-WIDE dataset only, we used the training images that are associated with exactly one of the most frequent 21 tags. The resultant dataset has a training set of 40,141 images with 21 classes. Similarly, we constructed a query set

of 26,585 images among the test images associated with only one of those tags. All datasets were first centered at zero and then each point vector was normalized to unit length.

For the KSH method, we used reduced datasets of 5,000 images uniformly sampled from the training sets of each dataset because the computational complexity of this approach does not allow it to be trained on the entire datasets as we do with other methods (Table 5.1). The tree depth parameter of FastHash is set to 2 due to its higher computational complexity. 1,000 images uniformly sampled from the training sets were used as anchor points $\{\mathbf{a}_l\}_{l=1}^{s}$. All the hashing methods with kernels (SVM-Hash, KSH, and SDH) share the same anchor points in training. All these kernel methods used an RBF kernel $\varphi(\mathbf{x}, \mathbf{a}) = \exp\big(-\|\mathbf{x} - \mathbf{a}\|^2/2\sigma^2\big)$ with a kernel width σ which is adjusted for each dataset.

For SVM-Hash, we adjusted the soft margin parameters of the SVMs and the kernel width based on cross-validation on some binary codes obtained from other hashing methods. Next, we empirically adjusted the other parameters λ and γ for each dataset. Similar to SDH, we prefer the multi-class SVM dominate the process of learning binary codes by setting λ to a large number. For example, for the CIFAR-10 dataset the parameter values are $C^x = 16$, $C^b = 10^{-3}$, $\lambda = m \times 10^8$ and $\gamma = 10^5$. The maximum number of iterations $\mathsf{max\_iter}$ in Algorithm 5.1 is set to 5 like SDH. We repeated the same evaluation methodology as in Chapter 4.

5.3.2 Effects of Training and Anchor Set Size

We start our experiments by analyzing the effects of training and anchor set size on retrieval. The SVM-Hash method is analyzed in details with a different number of anchor points and training points for comparison. Table 5.1 shows the comparative retrieval performance of our method with different training and anchor set sizes along with the state-of-the-art methods on the CIFAR-10 dataset for 32-bit hash codes regarding mAP and mean precision at Hamming radius 2. We also report the execution time for learning hashing functions as training time and the time needed for computing binary code of a single query using the learned hashing functions as test time. As expected, larger training and anchor sets provide better performance with longer execution time. The number of anchor points has a greater influence on retrieval performance than training size. Note that the scale of the anchor set also affects the test time. Our method outperforms other methods in retrieval performance while it has competitive execution time. Note that Table 5.1 shows the training time on a four-core machine, and our method learns its hash function by a parallel algorithm. The training time in Table 5.1 can be enhanced by a computer with a larger number of nodes. The most efficient method in the experiments is CCA-ITQ; however, it performs poorly in retrieval.

5.3.3 Retrieval Performance Analysis

For quantitative analysis, we report retrieval performance as mean average precision (mAP) for all methods on the three datasets in Figure 5.1. Precision-

Table 5.1: Results in mean average precision (mAP), mean of precision at Hamming radius $r = 2$, training and test time for 32-bit hash codes on CIFAR-10 dataset. For our method (SVM-Hash), the number of training samples varies from 300 to 1,0000. The experiments were performed on a machine with an Intel quad-core processor.

Method	**Training Set Size**	**Anchor Set Size**	**mAP**	**Precision at** $r = 2$	**Training Time** (s)	**Test Time** (μs)
SVM-Hash	5,000	300	0.322	0.383	2.7	6.1
	5,000	500	0.342	0.397	3.4	11.6
	5,000	1,000	0.366	0.420	7.5	15.9
	5,000	3,000	0.393	0.438	47.3	45.7
	50,000	300	0.360	0.417	61.683	6.3
	50,000	1,000	**0.429**	0.475	171.1	17.7
SDH	5,000	1,000	0.107	0.003	4.6	20.6
	50,000	1,000	0.415	**0.476**	25.0	24.9
Fasthash	5,000	-	0.292	0.202	47.5	86.8
	50,000	-	0.354	0.267	628.4	102.6
KSH	5,000	1,000	0.339	0.408	2,895.6	26.8
CCA-ITQ	5,000	-	0.290	0.364	0.2	3.8
	50,000	-	0.321	0.407	1.5	3.8

recall curves of all methods on the same datasets are displayed in Figure 5.2 for 32-bit length codes. Note that these precision and recall values are computed by combining the return set of all queries in a single set. The mean of precision and recall at Hamming radius $r = 2$ are shown in Tables 5.2 and 5.3, respectively. The mean of precision and recall values for 64-bits and longer are not presented because this evaluation is impractical for longer binary codes.

As seen in the figures and tables, the proposed hashing method clearly outperforms the state-of-the-art supervised hashing methods on the CIFAR-10 and MNIST datasets, in which each image belongs to a single class. Our method provides the best performance on the independent set with the aid of better generalization and bit balancing. Furthermore, the imbalance penalty improves the retrieval performance slightly for almost all cases. Note that we initialized $\mathbf{B}$ with uniformly random strings from the SVM-Hash without imbalance penalty *i.e.* $\gamma = 0$ while we used a sequential sampling procedure where balanced binary codes have a higher probability for the SVM-Hash version with imbalance penalty *i.e.* $\gamma > 0$. SDH unsurprisingly provides the second best performance. Our method slightly improves its performance by our changes in the classification framework. On the other hand, our method is affected by the imbalance in the class sizes of reduced NUS-WIDE dataset. About half of the images belong to a single class while the rest of them are distributed among 20 categories. Despite this fact, our method still has competitive results along with SDH and FastHash on this dataset.

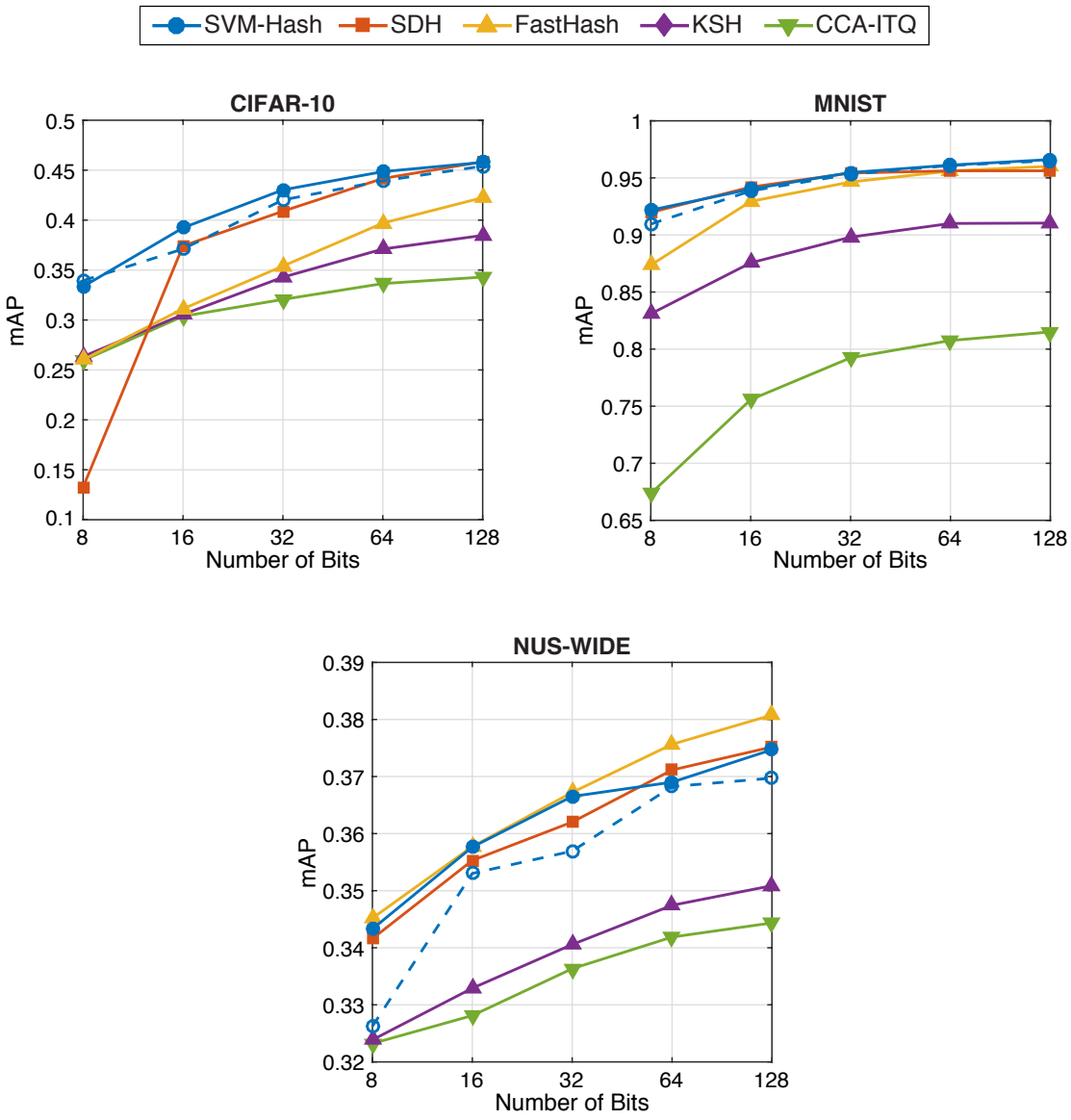


Figure 5.1: Our method (SVM-Hash) is compared with the state-of-the-art methods on the CIFAR-10, MNIST and NUS-WIDE datasets in terms of mean average precision (mAP). Dashed line represents SVM-Hash without imbalance penalty.

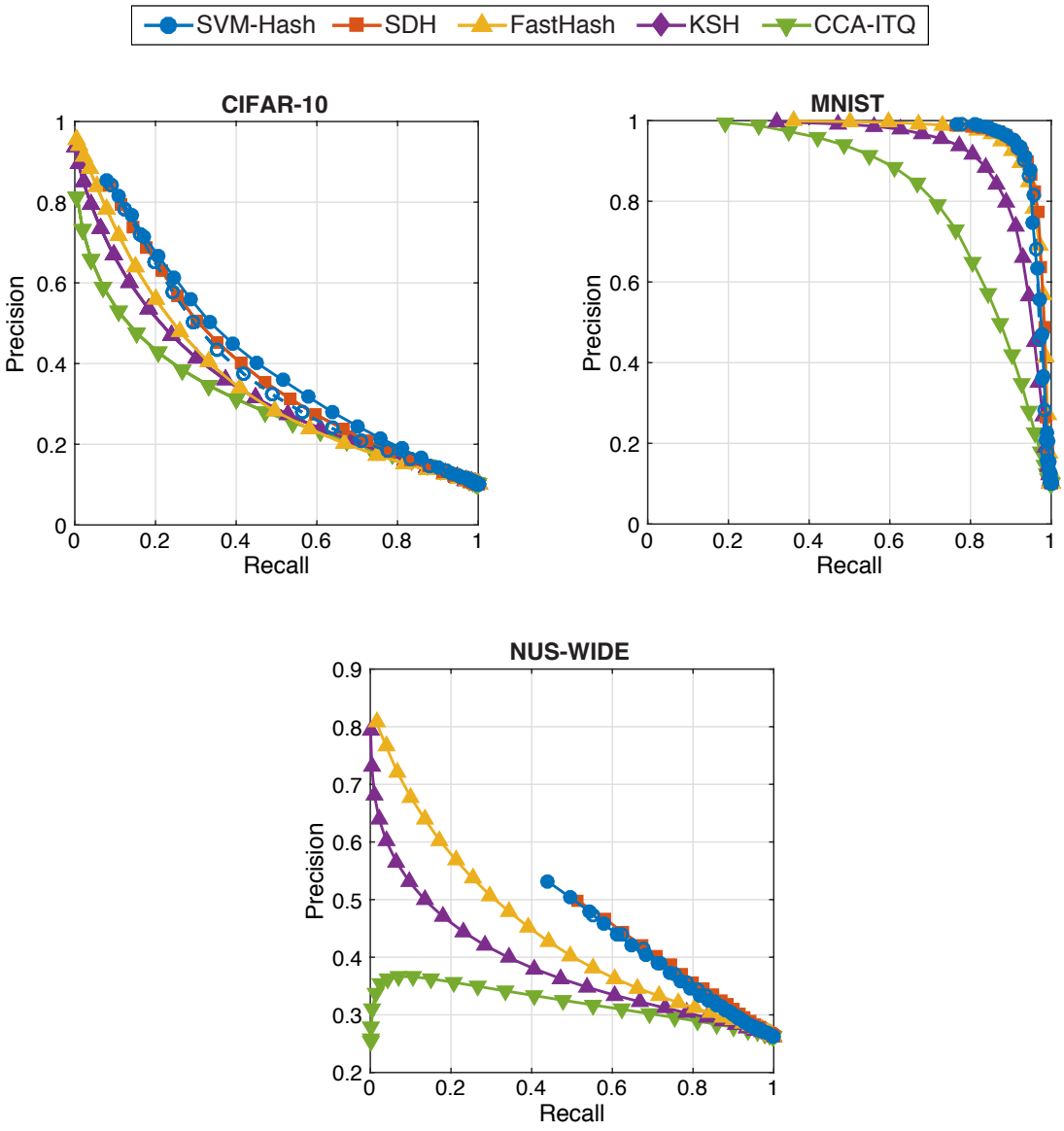


Figure 5.2: Our method (SVM-Hash) is compared with the state-of-the-art methods on the CIFAR-10, MNIST and NUS-WIDE datasets by precision-recall curves for 32-bit length hash codes. Dashed line represents SVM-Hash without imbalance penalty.

Table 5.2: Our method (SVM-Hash) is compared in terms of the mean of precision (%) at the Hamming radius $r = 2$.

Method	**CIFAR-10**			**MNIST**			**NUS-WIDE**		
	8-bits	16-bits	32-bits	8-bits	16-bits	32-bits	8-bits	16-bits	32-bits
SVM-Hash	**27.73**	**45.53**	**47.92**	63.36	93.58	**93.66**	30.81	33.98	**35.88**
SDH	14.59	41.86	47.53	76.62	83.16	83.08	33.15	33.83	34.99
FastHash	25.44	44.00	26.65	**77.42**	**93.65**	87.45	-	-	-
KSH	25.99	40.73	31.68	70.68	89.27	87.06	32.76	35.07	22.68
CCA-ITQ	25.20	38.93	41.36	56.36	81.63	81.83	**33.38**	**35.89**	26.77

Table 5.3: Our method (SVM-Hash) is compared in terms of the mean of recall (%) at the Hamming radius $r = 2$.

Method	**CIFAR-10**			**MNIST**			**NUS-WIDE**		
	8-bits	16-bits	32-bits	8-bits	16-bits	32-bits	8-bits	16-bits	32-bits
SVM-Hash	**51.96**	25.38	13.94	**95.16**	**88.56**	**83.81**	**72.14**	49.59	38.23
SDH	23.42	**35.90**	**14.43**	94.31	87.40	82.54	55.13	**54.15**	**46.70**
FastHash	41.61	9.11	2.07	90.31	76.13	59.09	-	-	-
KSH	42.98	10.06	2.02	88.48	71.29	55.21	28.48	5.59	0.64
CCA-ITQ	43.71	13.56	4.01	77.52	51.49	34.29	29.18	4.57	0.66

5.3.4 Retrieval Performance Analysis for Dynamic Datasets

We evaluated our incremental hashing strategy in comparison to the passive and from-scratch hashing strategies for three types of modifications on the same three datasets with 32-bit binary codes regarding mAP scores and training time. For deleting classes case, we first learned hash functions from the entire data. Next, we removed images of 1, 2 or 4 randomly selected classes and learned hash functions not only from scratch but also incrementally. We repeated this several times and reported mAP scores and training time for all type of hashing strategies. A similar methodology in the reverse order was employed for adding new classes case. For adding new images case, we first learned hash functions from 10%, 50% or 75% of training data selected randomly and then used the entire training set for from-scratch and incremental strategies. For adding new classes and adding new images cases, we used the same query set in the previous section. For deleting classes case, we removed the images associated with those classes and computed mAP scores for the rest of the queries. The mAP scores and training time are shown in Figures 5.3-5.8 for all types of modifications in all three datasets. Note that the training time reported for the passive strategy shows the initial computation only. As shown in the figures, our incremental hashing strategy reaches the same performance as the from-scratch hashing strategy in shorter training time. For the NUS-WIDE dataset, most of the classes are tiny. Therefore, adding or deleting those classes on that dataset has little effect on the retrieval performance. On the other hand, deleting classes from the MNIST dataset did not have any significant impact on the

retrieval performance because there is no room for improvement on that dataset. As shown in Table 5.1, adding new images improves the retrieval performance slightly while adding anchor points enhances considerably. Note that the total computation time for training on 10% of training data and incrementally training on the entire data is less than the training time on the entire from scratch. As a result, it might be sound strategy to use our supervised hashing method as a combination of training on a small set of representative images followed by incremental training on the entire training data.

5.4 Conclusion

We presented a supervised hashing method based on two-stage classification framework. We formulated a joint optimization task for the classification problem and the problem of finding optimal binary codes and developed an efficient algorithm for learning hash functions in a distributed scheme where the sub-problems are solved independently. Our objectives, which are to obtain higher quality codes by better generalization and balanced bits, are met by the proposed approach and validated by experiments on three image datasets. Furthermore, we showed that the incremental hashing strategy for dynamics datasets is capable of updating hash functions efficiently.

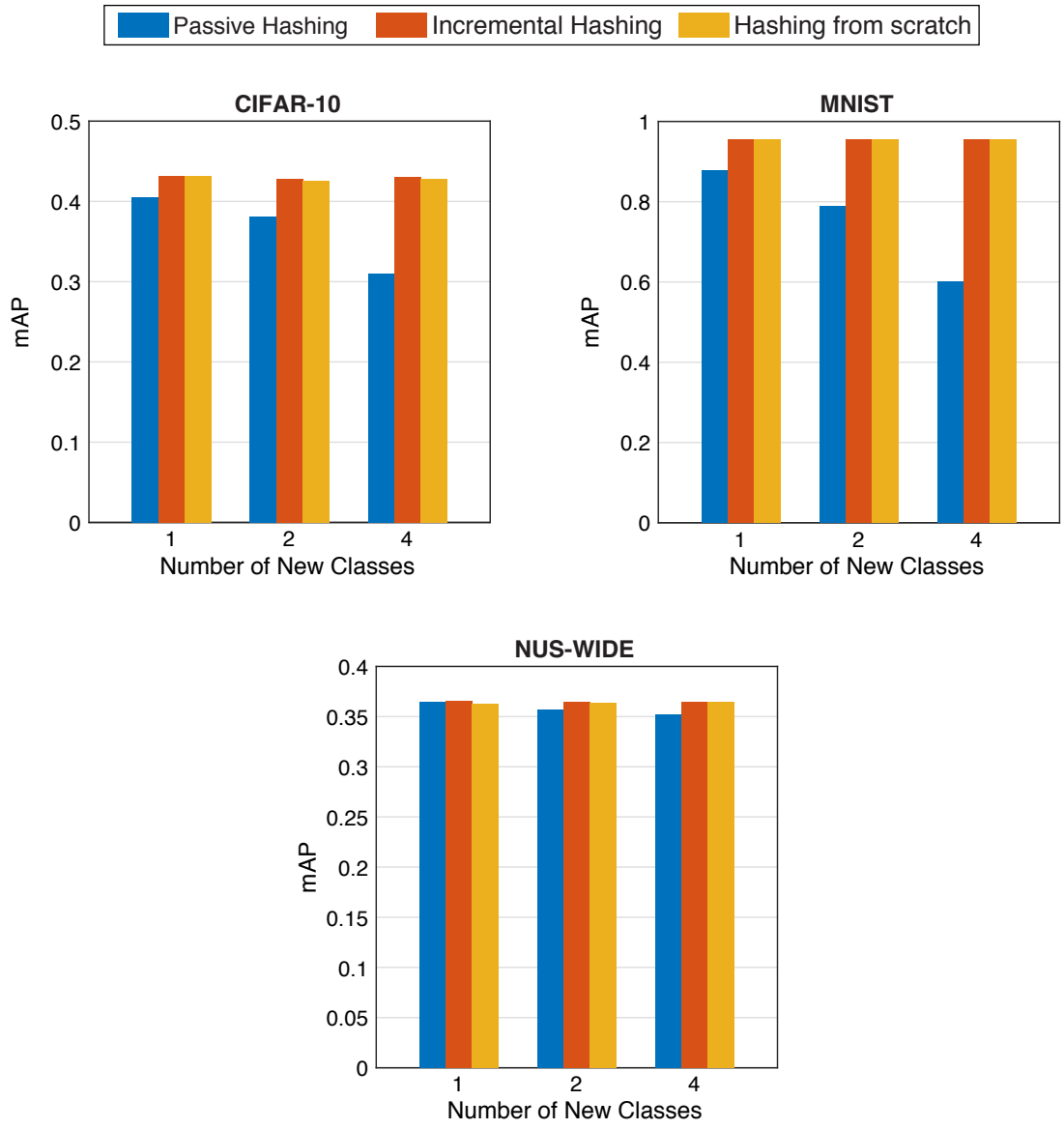


Figure 5.3: *Adding new classes*: Incremental hashing is compared with the from-scratch and passive hashing for adding different number of new classes to the CIFAR-10, MNIST and NUS-WIDE datasets in terms of mean average precision (mAP) at 32-bits.

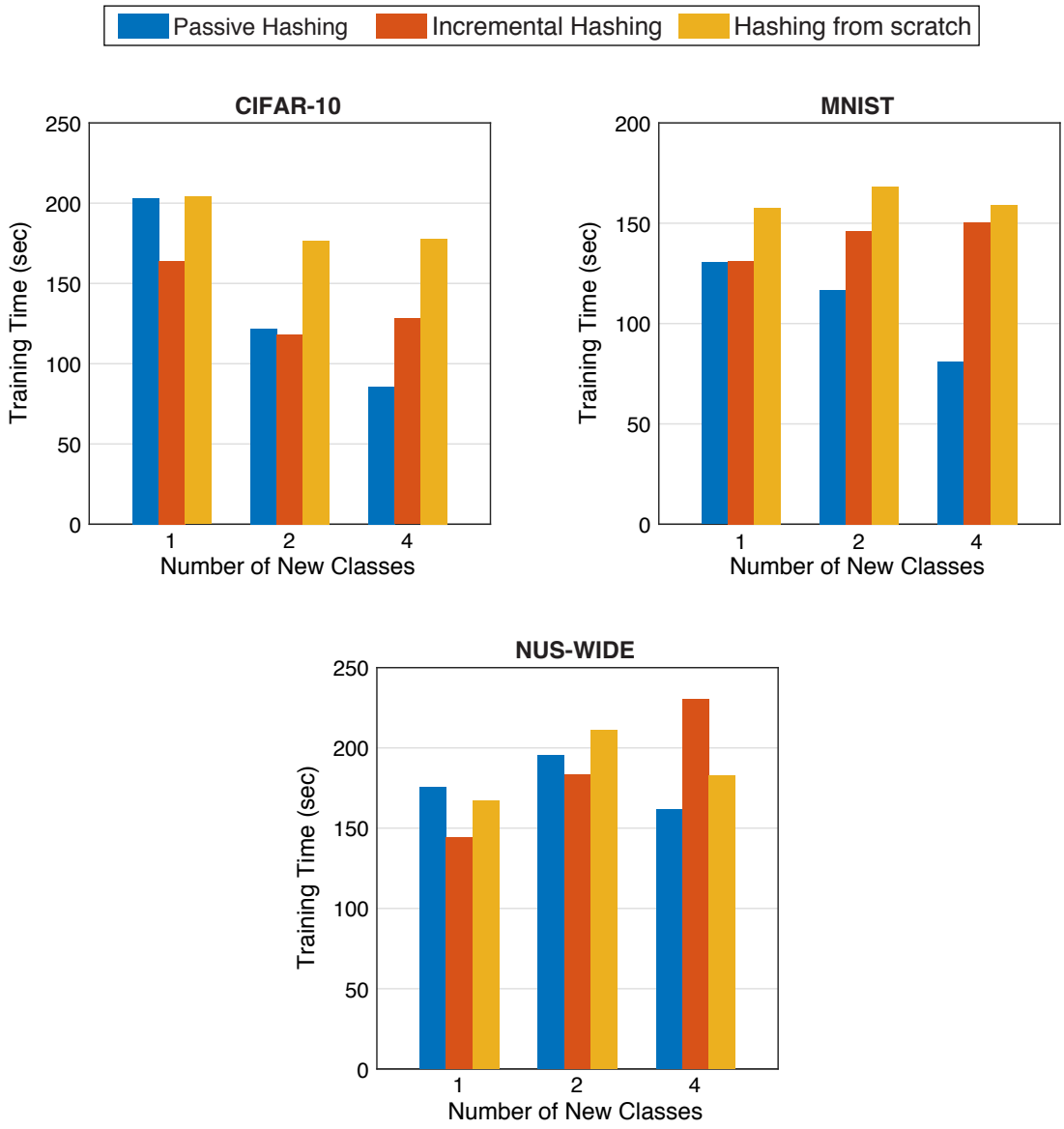


Figure 5.4: *Adding new classes*: Incremental hashing is compared with the from-scratch and passive hashing for adding different number of new classes to the CIFAR-10, MNIST and NUS-WIDE datasets in terms of training time at 32-bits.

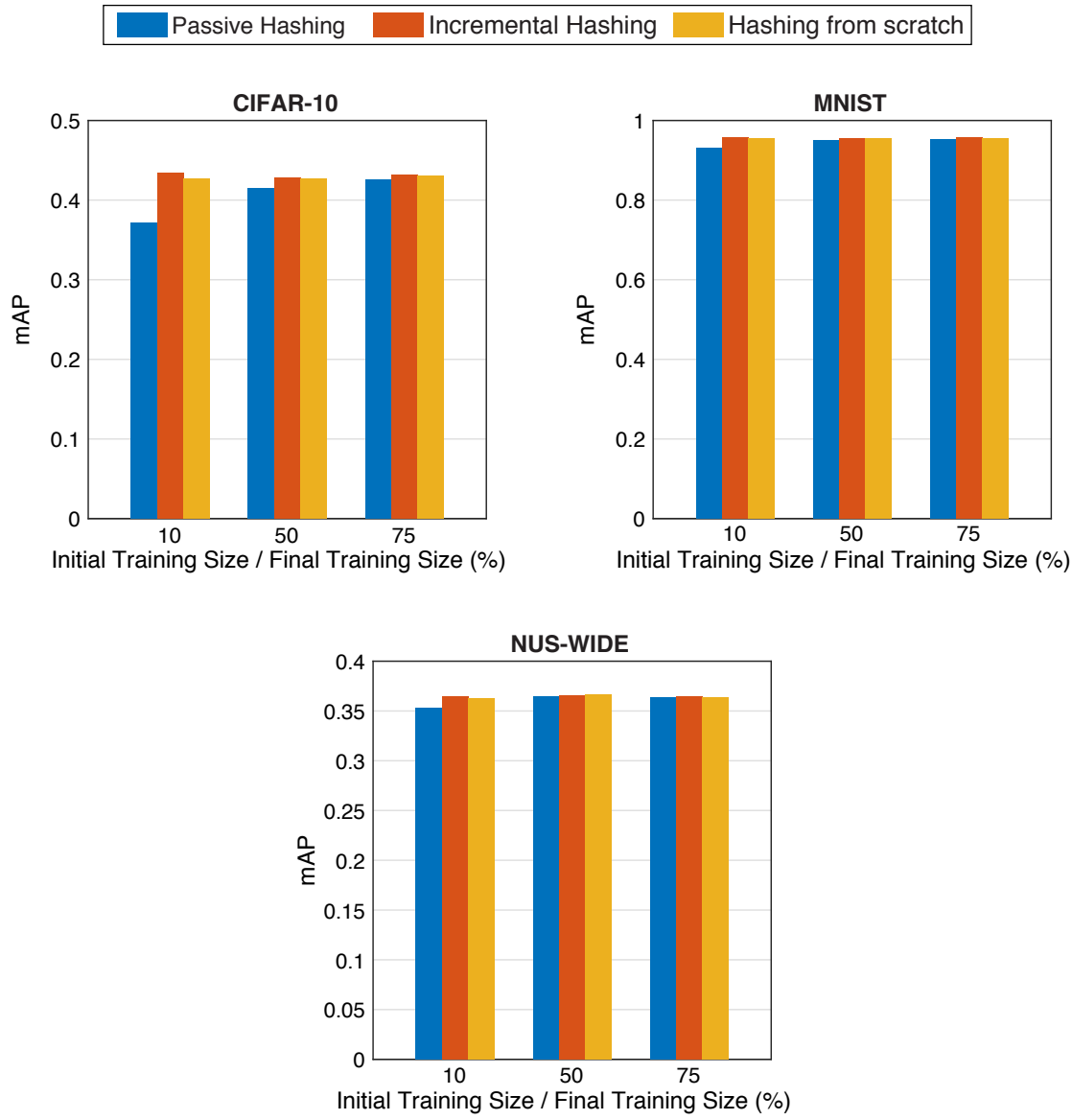


Figure 5.5: *Adding new images*: Incremental hashing is compared with the from-scratch and passive hashing for adding different number of new images to existing classes to the CIFAR-10, MNIST and NUS-WIDE datasets in terms of mean average precision (mAP) at 32-bits.

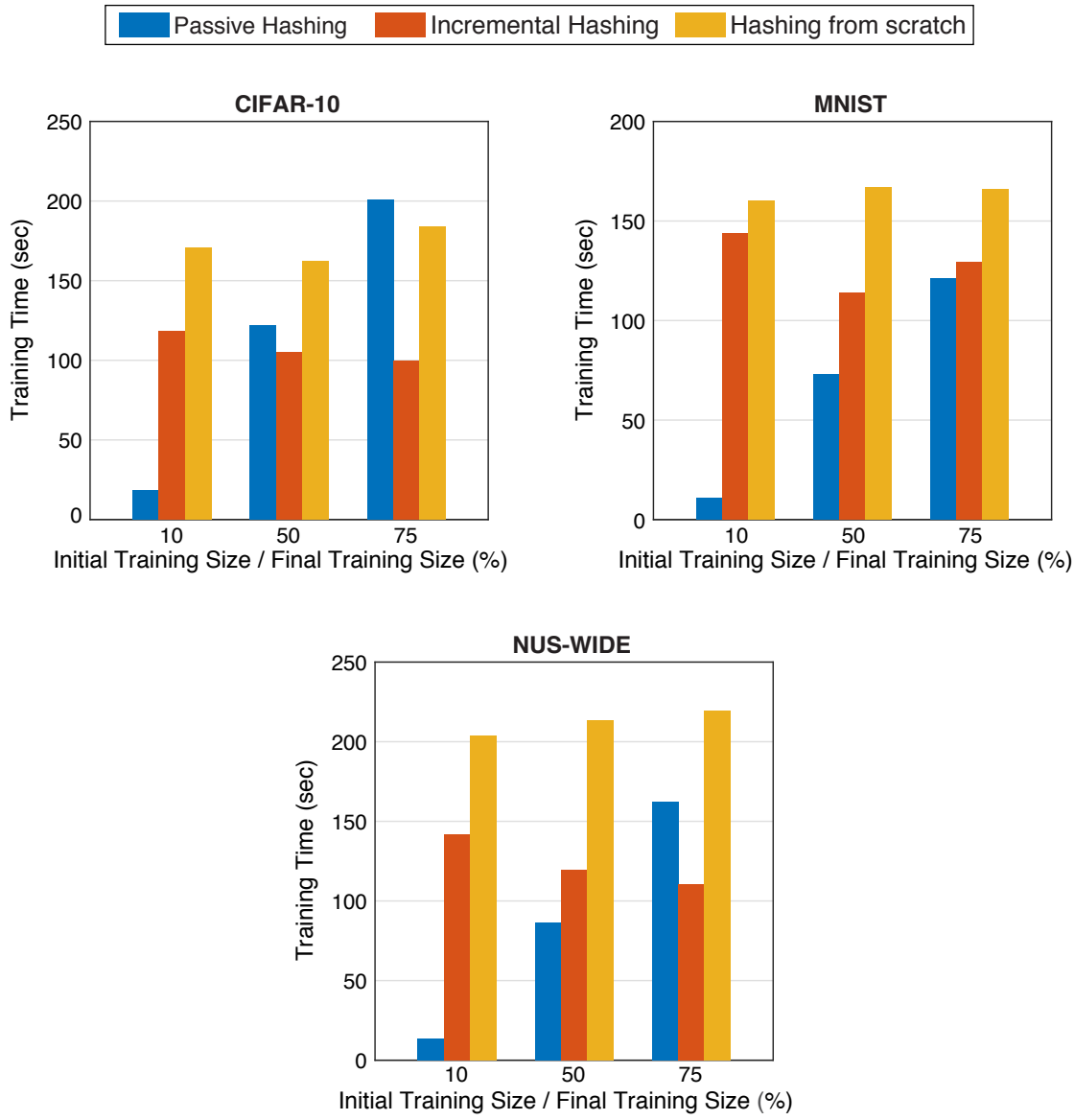


Figure 5.6: *Adding new images*: Incremental hashing is compared with the from-scratch and passive hashing for adding different number of new images to existing classes to the CIFAR-10, MNIST and NUS-WIDE datasets in terms of training time at 32-bits.

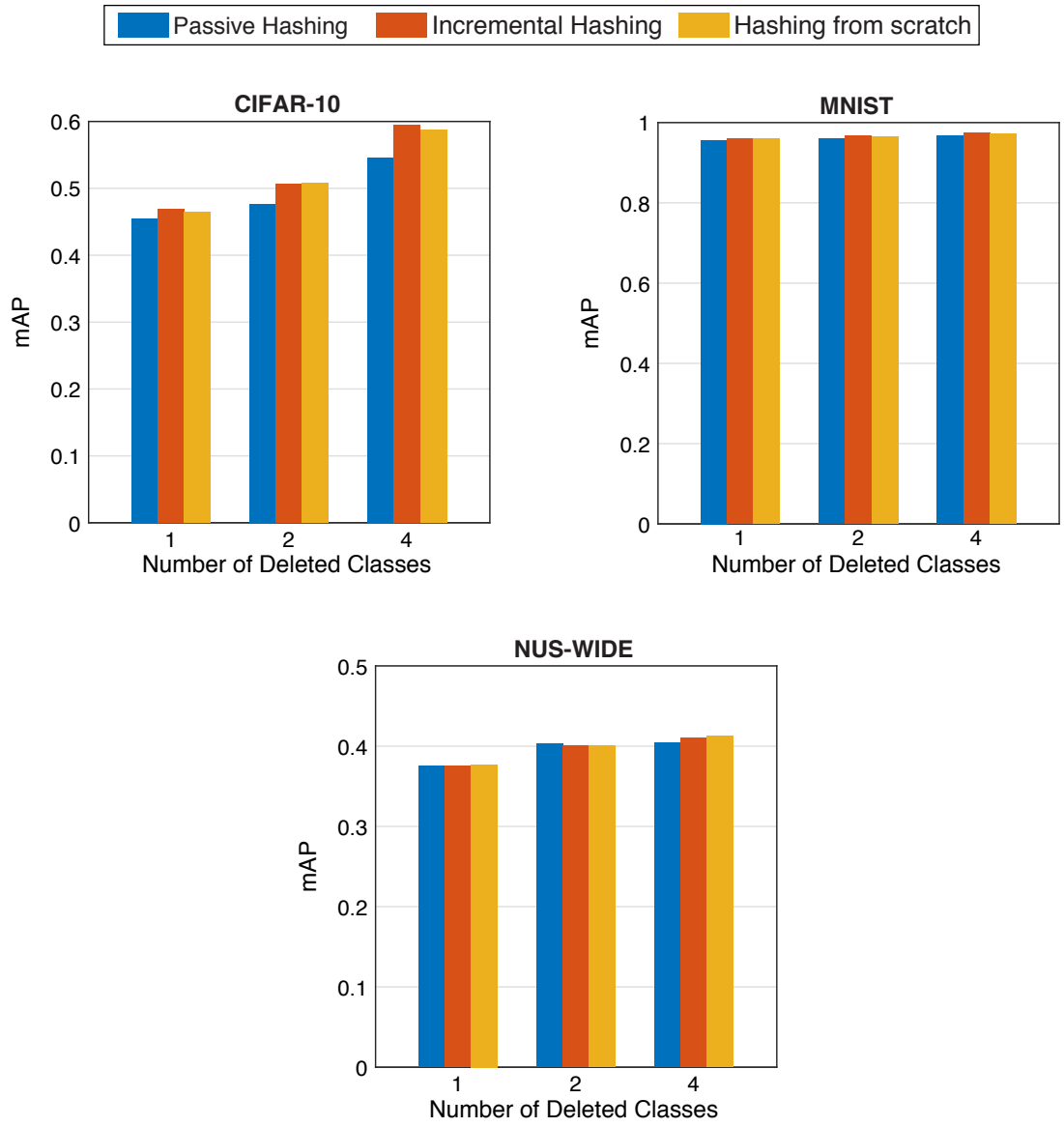


Figure 5.7: *Deleting existing classes*: Incremental hashing is compared with the from-scratch and passive hashing for deleting different number of existing classes from the CIFAR-10, MNIST and NUS-WIDE datasets in terms of mean average precision (mAP) at 32-bits.

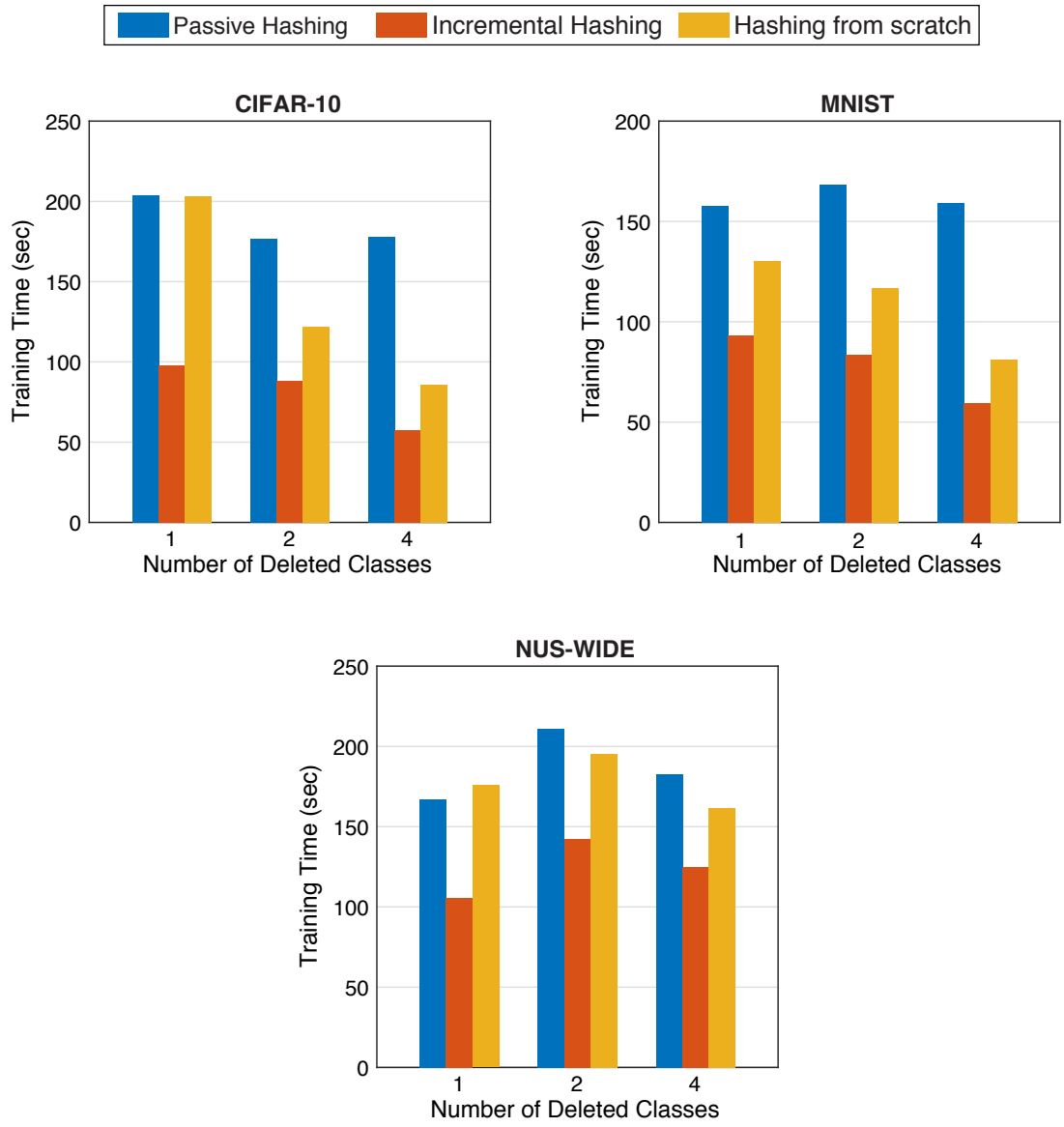


Figure 5.8: *Deleting existing classes*: Incremental hashing is compared with the from-scratch and passive hashing for deleting different number of existing classes from the CIFAR-10, MNIST and NUS-WIDE datasets in terms of training time at 32-bits.

Chapter 6: Conclusions and Future Work

My life would have taken a completely different turn. For better or for worse?

If you're still alive to ask yourself the question, it can't have been for the worse, can it?

"Ports of Call" – Amin Maalouf

6.1 Conclusions

We emphasized the importance of binary code representations for image retrieval. Accordingly, we studied three different types of hashing schemes in this thesis. In the first part, we proposed a retrieval framework based on latent binary features for multimodal data integration. A probabilistic retrieval model was presented for cross-modal queries. User judgments were also included in the framework for better retrieval performance. Experiments on the PASCAL-Sentence and SUN-Attribute datasets demonstrate the effectiveness of the proposed procedure in comparison to the state-of-the-art algorithms for multimedia databases.

In the second part, we introduced a supervised retrieval model based on Gaussian processes for classification. We elegantly constructed a network of latent variables by connecting multiple Gaussian processes with probit models on pairwise similarities. The model is able to learn binary codes for the datasets lacking well-

defined classes where the hashing problem becomes more challenging than for those with class labels. We described a scalable distributed inference algorithm for the proposed model. The experimental results on the CIFAR-10, MNIST, and NUS-WIDE datasets show that our method produces the best retrieval performance by preventing overfitting to training data.

In the last part, we restricted the problem of supervised hashing to image databases with predefined classes. With the help of this restriction, we presented a supervised hashing method based on two-stage classification framework. Also, we define an incremental hashing strategy on the proposed supervised hashing method. We described an efficient distributed and incremental algorithm for learning binary codes. The experimental results demonstrate that the SVM-Hash has the best retrieval performance in comparison to the state-of-the-art supervised hashing methods. Besides, the defined incremental strategy for dynamic datasets gives the same performance like the active strategy while being faster than that method for all types of modifications. We finally suggest a two-step incremental learning of the hash functions from a small set of representative images.

6.2 Future Work

Each hashing method that we presented in Chapters 3-5 can be further improved for better retrieval by small extension to the models or by different inference algorithms. For example, our retrieval model based on the integrative Indian buffet process can be extended to a hierarchical model like [70] where the first level clus-

ters images into categories. The Chinese restaurant process, which is a clustering method with infinite categories, will provide a flexible procedure. When similar images are grouped, and the integrative IBP is trained on each group, the training time will be shorter because very diverse data requires a large number of features in the IBP model and this results in longer training time. In addition, the iIBP inference can be made deterministic by a variational inference algorithm based on a truncated stick-breaking approximation. Therefore, the stability of retrieval set returned by iIBP will increase. Furthermore, a web-based user interface for iIBP makes user judgments easier, and it will contribute iIBP towards becoming a useful image retrieval tool for users.

The Gaussian process hashing method provides a predictive probability distribution for each bit. These distributions can be utilized for dynamic databases where changes might result in a formation of a new class or deletion of an existing class. Such modifications to a database require efficient updates to the hash functions and eventually binary code assignments for images in the database. We assume that data points which are closer to hyperplanes *i.e.* having predictive probabilities around 0.5 are more likely to change their binary codes in the Gibbs sampling step. This property can be used in an incremental-decremental learning scheme.

The SVM-Hash has a potential of improvement in two areas: Incremental learning of SVMs and a sophisticated initialization of binary codes. All hashing methods in this thesis are defined based on other models namely, Indian buffet process, Gaussian process, and SVM. Therefore, any future advances in these methods or their inference algorithms will eventually enhance our hashing methods.

Bibliography

[1] Genevieve Patterson, Chen Xu, Hang Su, and James Hays. The sun attribute database: Beyond categories for deeper scene understanding. *International Journal of Computer Vision (IJCV)*, 108(1-2):59–81, 2014.

[2] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *International Conference on Very Large Data Bases (VLDB)*, pages 518–529, 1999.

[3] Brian Kulis and Kristen Grauman. Kernelized Locality-Sensitive Hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(6):1092–1104, 2011.

[4] Maxim Raginsky and Svetlana Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1509–1517, 2009.

[5] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. Iterative quantization: a Procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 35(12):2916–2929, December 2013.

[6] Antonio Torralba, Rob Fergus, and Yair Weiss. Small codes and large image databases for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, June 2008.

[7] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1753–1760, 2009.

[8] Arnold W. M. Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(12):1349–1380, Dec 2000.

[9] Bahadir Ozdemir and Larry S Davis. A Probabilistic Framework for Multimodal Retrieval using Integrative Indian Buffet Process. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2384–2392, 2014.

[10] Abhishek Sharma, Abhishek Kumar, Hal Daumé III, and David W Jacobs. Generalized multiview analysis: A discriminative latent space. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2160–2167, June 2012.

[11] Xiang Sean Zhou and Thomas S. Huang. Relevance feedback in image retrieval: A comprehensive review. *Multimedia Systems*, 8(6):536–544, 2003.

[12] Yi Yang, Feiping Nie, Dong Xu, Jiebo Luo, Yueting Zhuang, and Yunhe Pan. A multimedia retrieval framework based on semi-supervised ranking and relevance feedback. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(4):723–742, April 2012.

[13] Bahadir Ozdemir and Larry S. Davis. Scalable Gaussian Processes for Supervised Hashing. *ArXiv e-prints*, cs.CV/1604.07335, April 2016.

[14] Malte Kuss and Carl Edward Rasmussen. Assessing Approximate Inference for Binary Gaussian Process Classification. *Journal of Machine Learning Research (JMLR)*, 6:1679–1704, October 2005.

[15] Bahadir Ozdemir, Mahyar Najibi, and Larry S. Davis. Incremental Hashing with Kernels. *ArXiv e-prints*, cs.CV/1604.07342, April 2016.

[16] Cyrus Rashtchian, Peter Young, Micah Hodosh, and Julia Hockenmaier. Collecting image annotations using amazon's mechanical turk. In *The NAACL HLT Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, 2010.

[17] Mohammad Rastegari, Jonghyun Choi, Shobeir Fakhraei, Hal Daumé III, and Larry S. Davis. Predictable Dual-View Hashing. *International Conference on Machine Learning (ICML)*, pages 1328–1336, 2013.

[18] Dekang Lin. An information-theoretic definition of similarity. In *International Conference on Machine Learning (ICML)*, pages 296–304, 1998.

[19] Genevieve Patterson and James Hays. Sun attribute database: Discovering, annotating, and recognizing scene attributes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2751–2758, June 2012.

[20] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3485–3492, June 2010.

[21] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

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

[23] Yann Lecun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[24] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yan-Tao Zheng. Nus-wide: A real-world web image database from national university of singapore. In *ACM International Conference on Image and Video Retrieval (CIVR)*, Santorini, Greece, 2009.

[25] Thomas L. Griffiths and Zoubin Ghahramani. Infinite latent feature models and the indian buffet process. In *Advances in Neural Information Processing Systems (NIPS)*, pages 475–482, 2005.

[26] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *ACM Symposium on Computational Geometry*, pages 253–262, 2004.

[27] Brian Kulis, Prateek Jain, and Kristen Grauman. Fast similarity search for learned metrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 31(12):2143–2157, 2009.

[28] Junfeng He, Wei Liu, and Shih-Fu Chang. Scalable similarity search with optimized kernel hashing. In *International ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1129–1138, 2010.

[29] Junfeng He, Regunathan Radhakrishnan, Shih-Fu Chang, and Claus Bauer. Compact hashing with joint optimization of search accuracy and time. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 753–760, 2011.

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

[31] Wei Liu, Cun Mu, Sanjiv Kumar, and Shih-Fu Chang. Discrete graph hashing. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3419–3427, 2014.

[32] Weihao Kong and Wu-Jun Li. Isotropic hashing. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1646–1654, 2012.

[33] Yunchao Gong, Sanjiv Kumar, Vishal Verma, and Svetlana Lazebnik. Angular quantization-based binary codes for fast similarity search. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1196–1204, 2012.

[34] Jae-Pil Heo, Youngwoon Lee, Junfeng He, Shih-Fu Chang, and Sung-Eui Yoon. Spherical hashing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2957–2964, June 2012.

[35] Alexis Joly and Olivier Buisson. Random maximum margin hashing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 873–880, June 2011.

[36] Brian Kulis and Trevor Darrell. Learning to hash with binary reconstructive embeddings. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1042–1050, 2009.

[37] Wei Liu, Jun Wang, Yadong Mu, Sanjiv Kumar, and Shih-Fu Chang. Compact hyperplane hashing with bilinear functions. In *International Conference on Machine Learning (ICML)*, 2012.

[38] Michael M. Bronstein, Alexander M. Bronstein, Fabrice Michel, and Nikos Paragios. Data fusion through cross-modality metric learning using similarity-sensitive hashing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3594–3601, June 2010.

[39] Shaishav Kumar and Raghavendra Udupa. Learning hash functions for cross-view similarity search. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1360–1365, 2011.

[40] Yi Zhen and Dit-Yan Yeung. Co-regularized hashing for multimodal data. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1376–1384, 2012.

[41] Yi Zhen and Dit-Yan Yeung. A probabilistic model for multimodal hash function learning. In *International ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 940–948, 2012.

[42] Nitish Srivastava and Ruslan Salakhutdinov. Multimodal learning with deep boltzmann machines. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2222–2230, 2012.

[43] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Sequential projection learning for hashing with compact codes. In *International Conference on Machine Learning (ICML)*, pages 1127–1134, 2010.

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

[45] Christoph Strecha, Alexander M Bronstein, Michael M Bronstein, and Pascal Fua. Ldahash: Improved matching with smaller descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(1):66–78, 2012.

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

[47] Mohammad Rastegari, Ali Farhadi, and David Forsyth. Attribute discovery via predictable discriminative binary codes. In *European Conference on Computer Vision (ECCV)*, pages 876–889, 2012.

[48] Ran He, Yinghao Cai, Tieniu Tan, and Larry S. Davis. Learning predictable binary codes for face indexing. *Pattern Recognition*, 48(10):3160 – 3168, 2015.

[49] Guosheng Lin, Chunhua Shen, Qinfeng Shi, Anton Van den Hengel, and David Suter. Fast supervised hashing with decision trees for high-dimensional data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1971–1978, June 2014.

[50] Mohammad Norouzi and David J Fleet. Minimal loss hashing for compact binary codes. In *International Conference on Machine Learning (ICML)*, 2011.

[51] Peichao Zhang, Wei Zhang, Wu-Jun Li, and Minyi Guo. Supervised hashing with latent factor models. *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 173–182, 2014.

[52] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. Supervised discrete hashing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[53] Ilker Yildirim and Robert A. Jacobs. A rational analysis of the acquisition of multisensory representations. *Cognitive Science*, 36(2):305–332, 2012.

[54] Finale Doshi-Velez and Zoubin Ghahramani. Accelerated sampling for the indian buffet process. In *International Conference on Machine Learning (ICML)*, pages 273–280, 2009.

[55] Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. Every picture tells a story: Generating sentences from images. In *European Conference on Computer Vision (ECCV)*, pages 15–29, Berlin, Heidelberg, 2010.

[56] Harold Hotelling. Relations Between Two Sets of Variates. *Biometrika*, 28(3/4):321–377, December 1936.

[57] Carl Edward Rasmussen and Christopher K I Williams. *Gaussian processes for machine learning*. MIT Press, Cambridge, Massachusetts, 2006.

[58] Hannes Nickisch and Carl Edward Rasmussen. Approximations for binary Gaussian process classification. *Journal of Machine Learning Research (JMLR)*, 9:2035–2078, October 2008.

[59] Joaquin Quinonero Candela and Carl Edward Rasmussen. A Unifying View of Sparse Approximate Gaussian Process Regression. *Journal of Machine Learning Research (JMLR)*, 6:1939–1959, December 2005.

[60] Edward Snelson and Zoubin Ghahramani. Sparse Gaussian Processes using Pseudo-inputs. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1257–1264, 2005.

[61] Andrew Naish-Guzman and Sean Holden. The generalized FITC approximation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1057–1064, 2007.

[62] Finale Doshi-Velez, David A Knowles, Shakir Mohamed, and Zoubin Ghahramani. Large Scale Nonparametric Bayesian Inference: Data Parallelisation in the Indian Buffet Process. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1294–1302, 2009.

[63] Thomas P Minka. Expectation Propagation for approximate Bayesian inference. In *Conference in Uncertainty in Artificial Intelligence (UAI)*, August 2001.

[64] Andrew Gelman, Aki Vehtari, Pasi Jylanki, Christian Robert, Nicolas Chopin, and John P Cunningham. Expectation propagation as a way of life. *arXiv.org*, December 2014.

[65] Daniel Hernandez-Lobato and Jose Miguel Hernandez-Lobato. Scalable Gaussian Process Classification via Expectation Propagation. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016.

[66] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research (JMLR)*, 2:265–292, March 2002.

[67] Rukshan Batuwita and Vasile Palade. *Class Imbalance Learning Methods for Support Vector Machines*, pages 83–99. John Wiley & Sons, Inc.

[68] Vojtěch Franc and Sören Sonnenburg. Optimized cutting plane algorithm for large-scale risk minimization. *Journal of Machine Learning Research (JMLR)*, 10:2157–2192, December 2009.

[69] Cheng-Hao Tsai, Chieh-Yen Lin, and Chih-Jen Lin. Incremental and decremental training for linear classification. In *International ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 343–352, New York, NY, USA, 2014.

[70] Thomas L. Griffiths, Michael I. Jordan, Joshua B. Tenenbaum, and David M. Blei. Hierarchical topic models and the nested chinese restaurant process. In *Advances in Neural Information Processing Systems (NIPS)*, pages 17–24, 2004.