Asymmetric Binary Coding for Image Search

Fumin Shen, Yang Yang, Li Liu, Wei Liu, Dacheng Tao and Heng Tao Shen

Abstract—Learning to hash has attracted broad research interests in recent computer vision and machine learning studies, due to its ability to accomplish efficient approximate nearest neighbor (ANN) search. However, the closely related task, Maximum Inner Product Search (MIPS), has rarely been studied in this literature. To facilitate the MIPS study, in this work, we introduce a general binary coding framework based on asymmetric hash functions, named Asymmetric Inner-product Binary Coding (AIBC). In particular, AIBC learns two different hash functions which can reveal the inner products between original data vectors by the generated binary vectors. Although conceptually simple, the associated optimization is very challenging due to the highly nonsmooth nature of the objective that involves sign functions. We tackle the nonsmooth optimization in an alternating manner, by which each single coding function is optimized in an efficient discrete manner. We also simplify the objective by discarding the quadratic regularization term which significantly boosts the learning efficiency. Both problems are optimized in an effective discrete way without continuous relaxations, which produces high-quality hash codes. In addition, we extend the AIBC approach to the supervised hashing scenario, where the inner products of learned binary codes are forced to fit the supervised similarities. Extensive experiments on several benchmark image retrieval databases validate the superiority of the AIBC approaches over many recently proposed hashing algorithms.

I. INTRODUCTION

In recent years, binary coding (also known as hashing) has become a very popular research subject in computer vision [22], [44], [48], [25], [52], [17], [37], [26], [57] and multimedia processing [50], [43], [42], [27], [19], [41], [40]. After encoding high-dimensional feature vectors of documents, images, or videos to compact binary codes, an effective binary coding or hashing method is expected to accomplish efficient similarity search while preserving the similarities among original data to some extent. As a result, using binary codes to represent and search in massive data is a promising solution to handle large-scale vision tasks, because of not only the storage efficiency (typically several hundred binary bits per data item) but also the high time efficiency of pairwise distance computations in the Hamming space.

Almost all the previous binary coding and hashing algorithms were designed to deal with Approximate Nearest Neighbor (ANN) search. Studies have rarely been dedicated to Maximum Inner Product Search (MIPS), which actually plays a critical role in various vision and learning applications [38]. The efficacy of aforementioned methods have not been validated on the MIPS problem yet. Shrivastava and Li [38] proposed an Asymmetric Locality-Sensitive Hashing (ALSH) algorithm for searching with (un-normalized) inner product being the underlying similarity measure. ALSH converts the MIPS problem to a standard LSH ANN problem by performing a simple asymmetric transformations on data pairs. While ALSH inherits all the theoretical guarantees of LSH [38], it can hardly achieve promising search performance using short binary codes due to its independence of data.

In this work, we focus on learning data-dependent binary codes for tackling the MIPS problem. Inspired by the latest advance in asymmetric hashing [38], we propose an asymmetric binary code learning method for MIPS, thus named Asymmetric Inner-product Binary Coding (AIBC). Specifically, two sets of coding functions are learned such that the inner products between their generated binary codes can reveal the inner products between original data vectors. Despite conceptually simple, the associated optimization is very challenging due to the highly nonsmooth nature of the objective that involves sign functions. To this end, we tackle the nonsmooth optimization in an alternating manner, by which a single coding function is solved with the others fixed. Through introducing auxiliary discrete variables to replace the sign functions, the optimization procedure is made efficient. As a simplified version of the proposed binary code learning framework, we propose another objective which maximizes the correlations between the inner products of the produced binary codes and raw data vectors. In both objectives, the binary codes and coding functions are simultaneously learned without continuous relaxations, which is the key to achieve high-quality binary codes.

The main contributions of our work are summarized as follows:

1) We propose a binary code learning framework for addressing the MIPS problem, which has the clear aim of preserving the inner-product similarities among raw data vectors. To this end, we design a tractable discrete optimization method, by which high-quality binary codes are iteratively generated with a closed-form solution for each bit.

2) To further speed up binary code learning for MIPS, we propose an alternative simpler objective which maximizes the correlations between the inner products of the yielded binary codes and raw data vectors. By this objective, the binary codes can be learned in a much more efficient way and, usually, with higher quality. This is mainly because the problem can be easily solved...
with closed-form solutions for the two associated sub-problems.

3) Based on the AIBC framework, we propose an asymmetric supervised hashing algorithm (ASH), which learns two sets of binary codes to fit the supervised similarities instead of the inner products of original data. The effectiveness of asymmetric hashing is further validated by the supervised hashing method.

4) Our AIBC methods (with two unsupervised objectives) together with the supervised ASH are extensively evaluated on several large-scale image datasets. The experimental results demonstrate the superiority of our methods over the state-of-the-arts.

The rest of this paper is organized as follows. Section II briefly reviews the related works. Section IV-B elaborates the details of the proposed AIBC methods. In Section V, we evaluate our approaches on three real-world large-scale datasets, followed by the conclusion of this work in Section VI.

This paper is an extended version of the work previously published in [33]. The major improvements in this work over [33] include the refinement of Abstract, Introduction (Section I) and Related work (Section II); We extend the original work to a general binary coding framework for maximum inner product search, which is studied by both the unsupervised AIBC approach and the newly introduced asymmetric supervised hashing algorithm (Section IV-D); More experimental results with comparison to additional methods are provided (Section V).

II. RELATED WORK

In this section, we briefly review related works in the hashing literature. The binary coding or hashing techniques can be roughly divided into two major categories: unsupervised and supervised methods.

A. Unsupervised hashing

Locality-Sensitive Hashing (LSH) [7] is one of the most popular data-independent unsupervised hashing methods, which generates randomized hash functions via random projections. The LSH family has been continuously developed to accommodate a variety of distance and similarity measures, such as Euclidean distance, $p$-norm distance [3], Mahalanobis distance [15], kernel similarity [14][32]. Although LSH is ensured to have high collision probability for similar data items, in practice LSH usually needs long hash bits and multiple hash tables to achieve both high precision and recall. The huge storage overhead may restrict its applications.

Recent years has witnessed a rapid development of the data-dependent methods (or learning-based methods). The literature was comprehensive reviewed in [46] recently. Its emergence is due to the benefit that learned compact binary codes can effectively and efficiently index and organize massive data. Different from LSH, data-dependent binary coding methods aim to generate short binary codes using available training data. Among the learning based hashing algorithms, linear coding functions formed by a set of learned hyperplanes are mostly adopted, since they are computationally simple and easy to connect to traditional dimensionality reduction techniques. A number of algorithms in this category have been proposed, including unsupervised PCA Hashing [44], Iterative Quantization (ITQ) [9], Isotropic Hashing (IsoHash) [12] and so on. Specially, the works [24][8] introduced a bilinear form of hash functions, which could well handle high-dimensional data. Along this lines, a few other types of hash functions had been proposed, such as Circulant Binary Embedding (CBE) [55] and Binary Projection Bank (BPB) [20].

It has also been shown that harnessing nonlinear manifold structures will help produce neighborhood-preserving compact binary codes. Spectral Hashing (SH) [48][47] is a well-known algorithm in this style. More recently, Anchor Graph Hashing (AGH) [23][21] leverages anchor graphs for making hash code training and out-of-sample extension to novel data both tractable and efficient. Shen et al. [36] proposed a general Inductive Manifold Hashing (IMH) scheme which generates nonlinear coding functions by exploiting the flexibility of available manifold learning approaches.

B. Supervised hashing

By taking advantages of the manually-labeled data, the hashing performance has been significantly improved by the supervised methods. Some representative supervised hashing algorithms are supervised Minimal Loss Hashing (MLH) [31], Semi-Supervised Hashing (SSH)[44], Ranking-Based Supervised Hashing [45], FastH [18] etc. It has also been studied to construct coding functions in a kernel space, for example, Binary Reconstructive Embedding (BRE) [13], Random Maximum Margin Hashing (RMMH) [11], Kernel-Based Supervised Hashing (KSH) [22], the kernel variant of ITQ [9] and Supervised Discrete Hashing (SDH) [34]. The kernel based hashing algorithms are shown to perform better than the linear based hashing ones in some applications.

Recently, a few deep learning based hashing algorithms had been proposed, where the features learned from deep neural networks were shown to achieve better performance than the hand-crafted ones [6][16]. For these deep hashing approaches, there might be concern about the binary coding time, since predicting a novel test sample needs to go through the time-consuming deep neural networks.

III. BACKGROUND: MAXIMUM INNER PRODUCT SEARCH

We give a brief introduction of Maximum Inner Product Search (MIPS). MIPS has been playing a significant role in various applications, such as recommender systems, deformable part model, multi-class classification [38]. Given a new query $q$, MIPS targets at retrieving the datum having the largest inner product with $q$ from the database $A$. Formally, the MIPS problem is formulated as below:

$$p = \arg \max_{a \in A} a^T q.$$  \hspace{1cm} (1)

For large-scale similarity search problems, it is practical to implement MIPS by employing binary coding techniques to achieve both storage and computational efficiencies. A binary coding function $h(x)$ maps an original feature vector $x$ to a...
binary code of \( r \) bits \( \mathbf{b} \in \{1, -1\}^r \) in the Hamming space. Then problem (1) is reformulated as follows

\[
p = \arg \max_{\mathbf{a} \in \mathbb{A}} \ h(\mathbf{a})^T h(\mathbf{q}). \tag{2}\]

It is natural to apply the popular locality sensitive hashing (LSH) for this problem. The LSH algorithm is simply constructed by the random projections generated from the standard normal distribution. Despite the popularity, the direct use of LSH on MIPS does not inherit the high collision probability guarantee of that on near neighbor search problems. To solve this problem, Shrivastava and Li [38] proposed the Asymmetric Locality Sensitive Hashing (ALSH) method, which converted the MIPS problem to the standard \( \ell_2 \) nearest neighbor search problem. ALSH adopts two sets of different hash functions \( h(\cdot) \) and \( z(\cdot) \) to compute the inner product of the query and database point. The MIPS problem is performed as

\[
p = \arg \max_{\mathbf{a} \in \mathbb{A}} \ h(\mathbf{a})^T z(\mathbf{q}), \tag{3}\]

where \( h(\cdot) \) and \( z(\cdot) \) are both constructed directly from LSH, by simply appending a few entries to \( \mathbf{a} \) and \( \mathbf{q} \) with different values. ALSH was proved to share the similar collision guarantee of similar points on MIPS as LSH on nearest neighbor search problem, which is mainly benefited from the flexibility of asymmetric hash functions. An asymmetric LSH algorithm with sketches [5] was previously proposed at a early stage. Recently, [30][29][28] discussed the effectiveness of asymmetric hashes.

IV. A GENERAL BINARY CODING FRAMEWORK FOR MAXIMUM INNER PRODUCT SEARCH

In this section, we present our binary code learning framework for maximum inner product search (MIPS), and then propose an inner-product fitting model to learn two asymmetric coding functions. To speed up the optimization, we practically simplify the original objective by maximizing inner-product correlation.

A. The binary coding framework

While the LSH and ALSH methods guarantee search accuracy to some extent, promising performance can hardly be achieved with compact binary codes due to the independence of data. To overcome the above limitation, we propose to learn hash functions rather than random projection for coping with the MIPS task.

Suppose that we have two sets of points: \( \mathbb{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n] \) and \( \mathbb{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m] \), where \( \mathbf{a}_i \in \mathbb{R}^{d \times 1} \) and \( \mathbf{x}_i \in \mathbb{R}^{d \times 1} \). Denote the similarity matrix of \( \mathbb{A} \) and \( \mathbb{X} \) as \( \mathbb{S} \in \mathbb{R}^{n \times m} \), of which the element \( S_{ij} \) defines the similarity of \( \mathbf{a}_i \) and \( \mathbf{x}_j \). We aim to learn two sets of binary codes for \( \mathbb{A} \) and \( \mathbb{X} \) respectively, the inner product of which can well approximate \( \mathbb{S} \). Inspired by [38], we adopt the asymmetric form of hash functions in our binary code learning formulation. That is, we consider the following problem:

\[
\min_{h, z} \sum_{i=1}^{n} \sum_{j=1}^{m} \ell(h(\mathbf{a}_i)^T z(\mathbf{x}_j), S_{ij}). \tag{4}\]

Here \( \ell(\cdot, \cdot) \) can be any proper loss function that forces the two terms to be close, which we will discuss later.

**Hash function**

The hash functions \( h(\cdot) \) and \( z(\cdot) \) can be a linear, non-linear (i.e., kernel-based) or even a deep neural networks based model. In this work, we focus on the simple linear hash function, i.e., the hash functions write

\[
h(\mathbf{a}) = \text{sgn}(\mathbf{W}^T \mathbf{a}) \tag{5}\]

\[
z(\mathbf{x}) = \text{sgn}(\mathbf{R}^T \mathbf{x}) \tag{6}\]

where \( \mathbf{W} \in \mathbb{R}^{d \times r} \), \( \mathbf{R} \in \mathbb{R}^{d \times r} \) is the projection matrix.

We name the proposed asymmetric binary coding methods for maximum inner-product search as Asymmetric Inner-product Binary Coding (AIBC).

B. Unsupervised: Inner product fitting

A natural approach to model the similarity of two points is through their inner product, i.e., \( S_{ij} = \mathbf{a}_i^\top \mathbf{x}_j \). In practice, to better fit with binary codes, we normalize \( S_{ij} \) to be \( r \) (or \( 0 \)) by a threshold.

We consider the following problem with hash functions \( h(\cdot) \) and \( z(\cdot) \),

\[
\min_{h, z} \| h(\mathbf{A})^\top z(\mathbf{X}) - \mathbb{S} \|^2. \tag{7}\]

Here \( \| \cdot \| \) is the Frobenius norm. Problem (7) turns out to be a problem of inner product fitting with binary codes. We will show next that the utilization of asymmetric hash functions can significantly facilitate the optimization of the above discrete optimization problem.

We then have

\[
\min_{\mathbf{W}, \mathbf{R}} \| \text{sgn}(\mathbf{W}^\top \mathbf{A})^\top \text{sgn}(\mathbf{R}^\top \mathbf{X}) - \mathbb{S} \|^2. \tag{8}\]

It is easy to see that the above problem is highly non-convex and difficult (usually NP hard) to solve due to the discrete sign functions. A feasible solution is to relax the discrete constraint by omitting the sign function. The continuous relaxation methodology is widely applied for hash code learning. However, this approximation method may accumulate considerable quantization errors, which makes the final binary codes less effective.

In order to obtain high-quality hash functions, we keep the sign function in our formulation. By taking the advantage of the flexibility of asymmetric hash functions, for this discrete optimization problem, we can naturally choose to solve \( \mathbf{W} \) and \( \mathbf{R} \) in an alternating fashion, that is, solve for one variable each time while keeping the other one fixed. We thus first consider the following sub-problem with variable \( \mathbf{W} \) by fixing \( z(\mathbf{x}) = \mathbf{Z} \) in (7),

\[
\min_{\mathbf{W}} \| \text{sgn}(\mathbf{W}^\top \mathbf{A})^\top \mathbf{Z} - \mathbb{S} \|^2. \tag{9}\]
In practice, we initialize $R$ by PCA projections. Similarly, by fixing $h(A) = H$, we obtain the following sub-problem with variable $R$,

$$\min_{R} \|H^\top \text{sgn}(R^\top X) - S\|^2, \quad (10)$$

Problem (9) or (10) is still with the sign function and cannot be solved trivially by an off-the-shelf solver. We now present a scalable and tractable method to (9), and (10) can be accordingly solved in the same way. To conquer the optimization involving discrete function, we introduce an auxiliary variable $B \in \{-1, 1\}^{r \times n}$ to separate the optimizing variable $W$ and the sign function. We will show next the introduction of the auxiliary variable $B$ (binary codes for $A$) is the key to simplify the optimization. We rewrite problem (9) as

$$\min_{B, W} \|B^\top Z - S\|^2 + \lambda \|B - W^\top A\|^2 \quad (11)$$

s.t. $B \in \{-1, 1\}^{r \times n}$.

The objective of (11) has a clear explanation: the first term minimizes the inner products fitting error by the learned binary codes; while the second term ensures the hash function $h(x)$ can well predict the target binary codes $B$ with minimum quantization loss. The parameter $\lambda$ serve a trade-off between these two loss terms.

In problem (11), given $B$, it is easy to compute $W$

$$W = (AA^\top)^{-1} A B^\top. \quad (12)$$

We then have the following problem w.r.t. $B$,

$$\min_{B} \|B^\top Z\|^2 - 2 \text{trace}(B^\top D) \quad (13)$$

s.t. $B \in \{-1, 1\}^{r \times n}$,

where $D = ZS^\top + A W^\top A$. Note that $\|B\|^2 = nr$.

For this key sub-problem, inspired by the recent progress of binary codes optimization [34], we choose to solve one row of $B$ each time while fixing all other rows, i.e., we compute one-bit for all $n$ samples each time. Let $b$ be the $l$th row of $B$, $l = 1, \cdots, r$, and $B$ the remaining rows of $B$. Then $b$ contains one bit for each of $n$ samples. Similarly, let $d$ be the $l$th row of $D$, $D$ the matrix of $D$ excluding $d$, $z$ the $l$th row of $Z$ and $\bar{Z}$ the matrix of $Z$ excluding $z$. With these notations and a few simple matrix manipulations, problem (13) can be written as w.r.t. $b$

$$\min_{b} b (B^\top \bar{Z} z^\top - d^\top) \quad (14)$$

s.t. $b \in \{-1, 1\}^n$.

Thus, we obtain the optimal solution for the $l$th row of $B$,

$$b = \text{sgn}(z \bar{Z}^\top b - d). \quad (15)$$

By this method, each bit is iteratively updated with the pre-learned $r - 1$ bits till the procedure converges with a set of better codes. In our experiments, the procedure usually converges with less than 10 iterations. With the analytical solution for each bit, the whole optimization is very efficient and thus can easily scale to massive data. Till now, we are ready to solve problem (11) (therefore also (9)) iteratively with the solution of $W$ and $B$ provided above. By iteratively solve (9) and (10), we finally obtain a pair of hash function of $h(\cdot)$ and $z(\cdot)$. Although this method can hardly achieve the globally optimal solution for the discrete optimization problem, at each step, the local optimal solution for each variable (e.g., $W$, $b$) is obtained in a closed form. We will show in the experiments that this optimization strategy works very well. Since the objective (7) of this method involves a quadratic term with the hash functions, we denote this method as AIBC-Q.

### C. Unsupervised: Inner product correlation maximization

In this part, we propose an alternative model to further speed up the learning procedure of AIBC. We optimize the following problem:

$$\max_{h, z} \text{trace} (h(A) S z(X)^\top). \quad (16)$$

Problem (16) has a clear intuition: it maximizes the correlation between the similarity matrix $S$ and the inner products between learned binary codes. Note that the objective of (16) can be easily obtained by discarding the quadratic term $\|h(A)^\top z(X)\|^2$ in (4). However, the quadratic term does not leverage the groundtruth similarity and can be seen as a regularization with the magnitude of the learned inner product. We show next it provides a much more efficient mean of solving the asymmetric hash function learning problem.

Using the same strategy described in the previous section, for (16) we obtain the following two sub-problems (17) and (18) w.r.t. $W$ and $R$, respectively.

$$\max_{W} \text{trace}(\text{sgn}(W^\top A) S Z^\top), \quad (17)$$

$$\max_{R} \text{trace}(H S \text{sgn}(R^\top X)^\top). \quad (18)$$

By introducing the auxiliary variable $B$, problem (17) is formulated as

$$\max_{B, W} \text{trace} (B S Z^\top) - \lambda \|B - W^\top A\|^2 \quad (19)$$

s.t. $B \in \{-1, 1\}^{r \times n}$.

Same as problem (11), we solve problem (19) iteratively with $B$ and $W$. The difference is, problem (19) benefits from the advantage that it has a optimal analytical solution for $B$ with a given $W$.

$$B = \text{sgn}(ZS^\top + 2 \lambda W^\top A). \quad (20)$$

**Compared to problem (4), optimizing (16) does not only make (19) much more efficient to solve but also provide more accurate solution for the whole optimization.** This explains why this method performs slightly better than AIBC-Q in our experiments. The optimization of (19) can be easily solved by iteratively updating $B$ and $W$ (by (12)). With the above closed-form solutions, the training of the proposed method can be easily performed on large-scale data with very high efficiency. Since the objective (16) of this method only involves a linear term with each hash function, we denote this method as AIBC-L.
Algorithm 1 Asymmetric Inner-product Binary Coding (AIBC)

**Input:** Data $A$ and $X$; code length $r$; maximum iteration number $t$; parameters $\lambda$.

**Output:** Hash function $h(x)$ and $z(x)$.

1. Compute similarity matrix $S$ by $A^T X$ for unsupervised hashing or by Eq. (21) for supervised hashing.
2. Initialize $R$ by PCA projections.
3. Loop until converge or reach maximum $t$ iterations:
   - $h$-step: Compute $W$ by solving problem (9) or (17).
   - $z$-step: Compute $R$ by solving problem (10) or (18).

\[ S_{ij}^l = \begin{cases} r, & \text{if } a_i \text{ and } x_j \text{ share the same label(s)}, \\ 0, & \text{otherwise}. \end{cases} \tag{21} \]

Particularly, for multiple labelled data, $S_{ij} = r$ if $a_i$ and $x_j$ share at least one labels. Similar settings has been used in many supervised hashing methods [22], [44]. With the $\ell_2$ loss as in (7), we write our supervised hashing as

\[ \min_{h,z} ||h(A)^T z(X) - S||^2. \tag{22} \]

Similarly, with the linear loss as in (16), we have

\[ \max_{h,z} \text{trace}(h(A)S^l z(X)^T). \tag{23} \]

Both the above two problems can be solved exactly the same as the unsupervised algorithm as discussed before. For efficiency, in practice we adopt the supervised hashing method in (23). The optimization follows the same procedure in Section IV-C with PCA as initialization. For the supervised hashing method, other supervised dimensionality reduction algorithms (e.g., LDA) can potentially improve the performance, which is however not the focus of this work. We denote this method as asymmetric supervised hashing (ASH) in the following text.

Connection to KSH

The kernel-based supervised hashing (KSH) [22] uses a similar objective as in (22):

\[ \min_h ||h(A)^T h(A) - S||^2. \tag{24} \]

However, our proposed AIBC mainly differs from KSH in the following aspects: 1) KSH learns a single coding function while AIBC learns two coding functions in an asymmetric fashion. 2) KSH applies a greedy optimization procedure to solve a relaxed problem by using a sigmoid function to replace the sign function. In contrast, AIBC directly optimizes the binary codes without resorting to any continuous relaxations. In the experiments, we compare our methods with KSH and its unsupervised version (implemented by ourselves) and the comparative results clearly show the advantage of our AIBC techniques. We summarize the proposed Asymmetric Inner-product Binary Coding (AIBC) algorithm in Algorithm 1.

V. EXPERIMENTS

In this section, we first compare the proposed AIBC to the state-of-the-arts in both supervised and unsupervised scenarios to validate its effectiveness. The AIBC model is then evaluated in terms of time efficiency, training convergence and parameter sensitiveness.

A. Evaluation on unsupervised hashing

In this section, we evaluate the proposed two unsupervised hashing algorithms (i.e., AIBC-Q and AIBC-L) by comparing them to several representative unsupervised hashing methods in the literature. Three large-scale datasets: YouTube Faces [49], SUN397 [51] and ImageNet [4] are used. The compared unsupervised hashing methods including LSH (implemented by signed random projections), ALSH [38], SH [48], ITQ [9], IsoHash [12], AGH [23] and IMH [35]. We also compare the unsupervised version of KSH [22] implemented by ourselves with the similarity computed by inner product (thus named InnerKSH). We use the public codes and suggested parameters of these methods from the authors. For InnerKSH, we use 4,000 training samples to form the pairwise similarity matrix and randomly choose 1,000 samples as anchor points. Since our methods are unsupervised, we do not compare them with the supervised methods. For our AIBC, we empirically set the parameter $\lambda$ to 100 and the maximum iteration number $t = 2$; the data matrix $A$ is set as the whole training data; $X$ is formed by the 10,000 randomly selected points from training data. We binarize each column of $S$ with a threshold computed by the $k_{th}$ largest one among the $n$ inner products, where $k$ is set to 500 for SUN397, 1000 for ImageNet and 2000 for YouTube Faces. For evaluation, database data and queries are compressed by $h(x)$ and $z(x)$, respectively.

We report the compared results in terms of both hash lookup: precision of Hamming distance 2 (HD2 Precision) and Hamming ranking: mean of average precision (mAP) and mean precision of the top 500 retrieved neighbors (Precision@500). We also present the detailed results by precision-recall and the precision of top 2000 curves. Note that we treat a query as a false case if no point is returned when calculating precisions. Ground truths are defined by the category information from the datasets.

1) SUN397: retrieval with scene images: SUN397 [51] contains about 108K images from 397 scene categories, where each image is represented by a 1,600-dimensional feature vector extracted by PCA from 12,288-dimensional Deep Convolutional Activation Features [10]. We use a subset of this dataset including 42 categories with each containing more than 500 images (with total 33K images); 100 images are sampled uniformly randomly from each category to form a test set of 4,200 images.

The comparative results are shown in Table I. First we can see that the proposed AIBC-L significantly outperforms all other algorithms in mAP with all code lengths. In terms of
precision500, AGH achieves the best result at 32-bit. However, we observe that it suffers from a severe performance drop with long code length for both MAP and precision. For example, AGH obtains much lower results than ITQ, IMH and AIBC at 128-bit. Our AIBC-Q performs slightly worse than AIBC-L on this dataset, although it still outperforms all other methods in most situations. It is not surprising that the data-independent algorithm LSH and ALSH do not perform as well as other learning based methods. Interestingly we also observe that the asymmetric ALSH algorithm achieves close results with the original signed random projection based LSH method on this dataset. The detailed precision curves of up to top 2000 retrieved samples and precision-recall curves using 64 bits are shown in Figure 1. We can easily see that the performance rank of the precision curves of the compared methods is consistent with the above analysis. The proposed AIBC still performs the best among the compared algorithms.

2) YouTube Faces: retrieval with face images: YouTube Faces dataset contains 1,595 different people, from which we choose 340 people such that each one has at least 500 images to form a subset of 370,319 face images, and represent each face image as a 1,770-dimensional LBP feature vector [1]. We use a subset of YouTube Faces with 38 people each containing more than 2,000 faces (about 100K images in total). The test set includes 3,800 face images which are evenly sampled from each of the 38 classes.

We report the results on YouTube Faces in Table II. Again, the proposed AIBC-L and AIBC-Q achieve the best results on this dataset in all situations. For instance, with 64 bits the proposed AIBC-L obtains 82.3% MAP which is higher than the second best 79.16% (by IMH) by 3.17%. SH and IsoHash do not perform as well as other data-dependent methods on this dataset. The proposed AIBC is further compared to these methods in precision of top 2000 samples and precision-recall. As can be clearly observed in Figure 2, AIBC ranks the first on both of the two evaluation curves.

3) ImageNet: retrieval with large dataset: As a subset of ImageNet [4], the large dataset ILSVRC 2012 contains over 1.2 million images of totally 1,000 categories. We form the retrieval database by the 100 largest classes with total 128K images from the provided training set, and 50,000 images from the validation set as the query set. As in [18], we use the 4096-dimensional features extracted by the convolution neural networks (CNN) model.

The results are shown in Table III. The superiority of the AIBC is further demonstrated on this large-scale database. For example, with 64-bit AIBC-L achieves 54.02% MAP while the best results of other methods is 50.95% obtained by AGH. Between the proposed methods, AIBC-L tends to achieve better MAP and precision than AIBC-Q with short binary codes (i.e., 32-bit and 64-bit), while these two methods obtain close results at 128-bit. Figure 3 illustrates the precision and recall of the compared methods on the SUN397.
### B. Evaluation on supervised hashing

We next validate the effectiveness of the proposed asymmetric hashing algorithm ASH by comparing with several recently proposed supervised hashing algorithms. These methods include Minimal Loss Hashing (MLH) [31], Kernel-Based Supervised Hashing (KSH) [22], CCA-ITQ [9] and Supervised Discrete Hashing (SDH) [34]. Due to the large memory overhead and computational costs, we sample 5K and 20K training samples for MLH, KSH and FastH, respectively. For the efficient CCA-ITQ, all available training data is used. As in Section V-A, we use all training data for A and 10,000 samples for X for our method.

1) Retrieval with single-labeled images: In this part, we use the single-labeled datasets SUN397 and ImageNet for evaluation. The comparative results in MAP and Precision@500 are reported in Table V. It can be clearly seen that on these two datasets our ASH significantly outperforms other methods in most cases. On the SUN 397 dataset, CCA-ITQ, KSH, FastH and SDH achieve on par MAPs with short bit length, while FastH perform best at 128-bit (75.26%), which is however still much lower than ASH (82.71%). In terms of precision500, ASH also outperform all other methods by over 4% accuracy. On the ImageNet dataset, SDH performs the second best and ASH also outperform all other methods by over 4% accuracy.

### TABLE IV: Comparison in HD2 precision with code length 32.

<table>
<thead>
<tr>
<th>Method</th>
<th>SUN 397</th>
<th>Youtube Faces</th>
<th>ImageNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSH</td>
<td>0.0352</td>
<td>0.3262</td>
<td>0.0644</td>
</tr>
<tr>
<td>ALSH</td>
<td>0.2479</td>
<td>0.9596</td>
<td>0.4029</td>
</tr>
<tr>
<td>SH</td>
<td>0.3101</td>
<td>0.9561</td>
<td>0.4194</td>
</tr>
<tr>
<td>IsoHash</td>
<td>0.3815</td>
<td>0.9646</td>
<td>0.4619</td>
</tr>
<tr>
<td>ITQ</td>
<td>0.5195</td>
<td>0.8561</td>
<td>0.4903</td>
</tr>
<tr>
<td>AGH</td>
<td>0.3107</td>
<td>0.7467</td>
<td>0.2493</td>
</tr>
<tr>
<td>IMH</td>
<td>0.3735</td>
<td>0.9697</td>
<td>0.4942</td>
</tr>
<tr>
<td>InnerKSH</td>
<td>0.4684</td>
<td>0.9859</td>
<td>0.5280</td>
</tr>
<tr>
<td>AIBC-Q</td>
<td>0.4704</td>
<td>0.9931</td>
<td>0.5595</td>
</tr>
<tr>
<td>AIBC-L</td>
<td>0.4771</td>
<td>0.5402</td>
<td>0.6585</td>
</tr>
</tbody>
</table>

### TABLE V: MAP and Precision500 of the compared supervised hashing methods on SUN 397 and ImageNet.

<table>
<thead>
<tr>
<th>Method</th>
<th>SUN 397</th>
<th>Youtube Faces</th>
<th>ImageNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSH</td>
<td>0.0352</td>
<td>0.3262</td>
<td>0.0644</td>
</tr>
<tr>
<td>ALSH</td>
<td>0.2479</td>
<td>0.9596</td>
<td>0.4029</td>
</tr>
<tr>
<td>SH</td>
<td>0.3101</td>
<td>0.9561</td>
<td>0.4194</td>
</tr>
<tr>
<td>IsoHash</td>
<td>0.3815</td>
<td>0.9646</td>
<td>0.4619</td>
</tr>
<tr>
<td>ITQ</td>
<td>0.5195</td>
<td>0.8561</td>
<td>0.4903</td>
</tr>
<tr>
<td>AGH</td>
<td>0.3107</td>
<td>0.7467</td>
<td>0.2493</td>
</tr>
<tr>
<td>IMH</td>
<td>0.3735</td>
<td>0.9697</td>
<td>0.4942</td>
</tr>
<tr>
<td>InnerKSH</td>
<td>0.4684</td>
<td>0.9859</td>
<td>0.5280</td>
</tr>
<tr>
<td>AIBC-Q</td>
<td>0.4704</td>
<td>0.9931</td>
<td>0.5595</td>
</tr>
<tr>
<td>AIBC-L</td>
<td>0.4771</td>
<td>0.5402</td>
<td>0.6585</td>
</tr>
</tbody>
</table>

Precision-recall curves, which clearly show AIBC performs better than other methods on this large dataset.

At last, we evaluate the proposed AIBC by hash lookup in the metric of Hamming distance 2 precision. The compared results of various methods with 32 bits are shown in Table IV. AGH obtains the best results (51.95%) on the SUN 397 dataset. However, AGH is surpassed by AIBC by large gaps, i.e., 13.7% and 6.9% on Youtube faces and ImageNet, respectively. The proposed AIBC methods (AIBC-Q and AIBC-L) also obtain similar results under this metric.

### Evaluation on supervised hashing

We next validate the effectiveness of the proposed asymmetric hashing algorithm ASH by comparing with several recently proposed supervised hashing algorithms. These methods include Minimal Loss Hashing (MLH) [31], Kernel-Based Supervised Hashing (KSH) [22], CCA-ITQ [9], FastH [18] and Supervised Discrete Hashing (SDH) [34]. Due to the large memory overhead and computational costs, we sample 5K and 20K training samples for MLH, KSH and FastH, respectively. For the efficient CCA-ITQ, all available training data is used. As in Section V-A, we use all training data for A and 10,000 samples for X for our method.

1) Retrieval with single-labeled images: In this part, we use the single-labeled datasets SUN397 and ImageNet for evaluation. The comparative results in MAP and Precision@500 are reported in Table V. It can be clearly seen that on these two datasets our ASH significantly outperforms other methods in most cases. On the SUN 397 dataset, CCA-ITQ, KSH, FastH and SDH achieve on par MAPs with short bit length, while FastH perform best at 128-bit (75.26%), which is however still much lower than ASH (82.71%). In terms of precision500, ASH also outperform all other methods by over 4% accuracy. On the ImageNet dataset, SDH performs the second best and ASH also outperform all other methods by over 4% accuracy. The detailed comparison of these methods are furthered shown in Precision of top 2000 curves Precision-recall curves in Figure 4 and 5 on the two datasets. Our method shows consistent improvements over other state-of-the-art approaches. These results clearly demonstrate the effectiveness of asymmetric hashing for image search in the supervised setting.

2) Retrieval with multi-labeled images: In this part, we compare these supervised hashing methods on the NUS-WIDE dataset containing images annotated by multiple labels. The NUS-WIDE database [2] contains about 270,000 images collected from Flickr. The images in NUS-WIDE are associated with 81 concepts, with each image containing multiple semantic labels. We define the true neighbors of a query as the images sharing at least one labels with the query image. The
Through the above experiments, we conclude that the proposed AIBC can achieve promising retrieval results compared to both supervised and unsupervised state-of-the-art hashing methods. We next analyze the AIBC model by the computational efficiency, parameter sensitiveness and convergence.

C. Algorithm analysis

Efficiency We take the SUN397 dataset as an example to evaluate the computational efficiency of these compared algorithms. The model training time and the testing time (of compressing one query into binary codes with the trained model) are shown in Table VII. First, we are not surprised to see that the proposed AIBC-L has a significant computational advantage over AIBC-Q, due to the simplified objective. Compared to SH, IsoHash and AGH, AIBC-L consumes more time to train the hash functions, which is mainly occupied by the inner product matrix calculation. However, the training of AIBC-L is still sufficiently efficient to scale to large-scale data: it runs only about 15 seconds on a standard PC for training with the whole 33K images of the SUN397 database. As can be seen, InnerKSH suffers from a huge computational overhead with the greedy optimization procedure, while AIBC-L can be trained much more efficiently with the closed-form solution of each sub-problem.

Parameter sensitive study In this part, we empirically evaluate the proposed algorithm (AIBC-L) on ImageNet and SUN397 with parameter $\lambda$ varied from 0.1 to 10,000. We observe from Figure 7 that the performance reaches the peak at $\lambda = 100$, and drops dramatically with larger $\lambda$s. In Eq. 11, the value of $W^T A$ will be exactly equal to $B$ with arbitrary large $\lambda$. This result demonstrate that allowing a certain discrepancy between $B$ and the magnitude $W^T A$ of the sign function is preferred for better performance.

Convergence study We also show in Figure 8 the objective values of our algorithm with increasing number of iterations. As can be seen, the algorithm converges within about 10 iterations. We conclude the fast convergence is due to the optimal analytical solution obtained in each sub-problem. The analytical solution not only provides fast training but also achieves high-quality codes as in the previous experiments.

VI. Conclusion and Future Work

This paper focused on binary code learning for the maximum Inner product search (MIPS) problem through proposing an inner-product fitting framework. In the framework, two asymmetric coding functions are learned such that the inner products between original data pairs are approximated by the produced binary code vectors. Benefiting from the flexibility of the asymmetric coding mechanism, we solved the optimization in an alternating fashion involving two sub-problems. We also...
proposed an alternative objective that maximizes the correlations between the inner products of the pursed binary codes and raw data vectors, which enjoys a closed-form solution for each sub-problem, thus making the whole optimization more efficient. The effectiveness of the proposed Asymmetric Innerproduct Binary Coding (AIBC) technique with both quadratic (AIBC-Q) and linear objectives (AIBC-L) were validated on several large-scale image datasets. As an extension of AIBC, we proposed an asymmetric supervised hashing (ASH) algorithm, which maximized the correlations between the supervised similarity matrix and code inner products. Despite visual retrieval, the proposed AIBC can be potentially used in recommendation systems [56], visual recognition [53][54][39] and related tasks.

REFERENCES


TABLE VII: Comparison of the the computational efficiency (in consumed seconds) on SUN397. Both the training and testing time are compared with two different code lengths.

<table>
<thead>
<tr>
<th>Method</th>
<th>Training time 64-bit</th>
<th>Testing time 64-bit</th>
<th>Training time 128-bit</th>
<th>Testing time 128-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSH</td>
<td>0.0846 0.1459</td>
<td>3.136e-6 3.629e-6</td>
<td>0.1459 0.2121</td>
<td>6.366e-6 7.458e-6</td>
</tr>
<tr>
<td>ALSH</td>
<td>0.5460 0.5976</td>
<td>1.296e-6 1.486e-6</td>
<td>0.5976 0.6457</td>
<td>2.516e-6 2.736e-6</td>
</tr>
<tr>
<td>SH</td>
<td>2.377 5.223</td>
<td>2.750e-5 1.060e-4</td>
<td>5.223 10.446</td>
<td>6.335e-5 2.405e-4</td>
</tr>
<tr>
<td>IsoHash</td>
<td>1.825 2.311</td>
<td>3.664e-6 6.335e-6</td>
<td>2.311 3.657</td>
<td>1.296e-5 2.516e-5</td>
</tr>
<tr>
<td>ITQ</td>
<td>3.326 5.296</td>
<td>3.527e-6 6.409e-6</td>
<td>5.296 10.518</td>
<td>5.037e-5 2.516e-4</td>
</tr>
<tr>
<td>AGH</td>
<td>0.623 0.656</td>
<td>1.905e-6 1.925e-6</td>
<td>0.656 0.680</td>
<td>1.296e-5 1.296e-5</td>
</tr>
<tr>
<td>IMH</td>
<td>2.427 2.964</td>
<td>2.402e-5 2.486e-5</td>
<td>2.964 3.563</td>
<td>1.296e-5 1.296e-5</td>
</tr>
<tr>
<td>InnerKSH</td>
<td>1933.1 4259.1</td>
<td>5.533e-5 6.1706e-5</td>
<td>4259.1 8518.3</td>
<td>1.296e-5 1.296e-5</td>
</tr>
<tr>
<td>AIBC-Q</td>
<td>53.29 173.9</td>
<td>3.179e-6 4.240e-6</td>
<td>173.9 511.8</td>
<td>1.296e-5 1.296e-5</td>
</tr>
<tr>
<td>AIBC-L</td>
<td>15.19 15.61</td>
<td>3.272e-6 3.853e-6</td>
<td>15.61 16.12</td>
<td>1.296e-5 1.296e-5</td>
</tr>
</tbody>
</table>

Fig. 7: MAPs with varying $\lambda$.

Fig. 8: Objective value with iterations.


J. Song, Y. Yang, Y. Yang, Z. Huang, and H. T. Shen. Inter-media hashing for large-scale retrieval from heterogeneous data sources. In SIGMOD, 2013.


Heng Tao Shen is a professor with School of Computer Science and Engineering, University of Electronic Science and Technology of China. He obtained his B.Sc. (with 1st class Honours) and Ph.D. from Department of Computer Science, National University of Singapore in 2000 and 2004 respectively. He then joined the University of Queensland as a Lecturer and became a Professor in 2011. His research interests include multimedia retrieval and mining, machine learning and data mining. He is the winner of Chris Wallace Award for outstanding Research Contribution in 2010 from CORE Australasia. He is an Associate Editor of IEEE TKDE, and will serve as a PC Co-Chair for ACM Multimedia 2015.